Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2024 р

Пояснювальна записка
до кваліфікаційної роботи
бакалавра

на тему: **СИСТЕМА РОЗПІЗНАВАННЯ ПЕРЕШКОД ДЛЯ АВТОПЛОТА**

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.06.2024 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
студент 4 курсу, групи КІ–41
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – Комп'ютерна інженерія.
СЕРГЄЄВ Вячеслав Олександрович
_____

Керівник: к.т.н., доцент ЗВО кафедри ТПС
БИКОВА Тетяна Володимирівна _____

Рецензент: к.ф.-м.н., доцент ЗВО, зав. кафедри електроніки і управляючих систем

ХРУСЛОВ Максим Михайлович _____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і двох додатків. Загальний обсяг роботи складає 81 сторінки, із яких 50 сторінок основної частини з 11 рисунками, 1 таблицею, 8 найменуваннями списку використаних джерел та двома додатками.

Об'єкт дослідження – процеси розпізнавання перешкод системами з автопілотом

Предмет дослідження – методи і алгоритми розпізнавання перешкод системами з автопілотом.

Проблема, яку вирішує кваліфікаційна робота: Розробка та оптимізація моделі з функціями автопілоту для розпізнавання та уникнення перешкод. Існуючі моделі часто мають обмежені можливості щодо точності розпізнавання перешкод, недостатню надійність в умовах змінного навколишнього середовища та складнощі в інтеграції з іншими системами. Робота спрямована на підвищення ефективності та надійності автопілотних систем для різних застосувань.

Область застосування – Автоматизація та покращення автономних систем керування транспортними засобами. Розроблена модель з функціями автопілоту може широко використовуватися в різних галузях, включаючи безпекові та рятувальні операції, цивільне та військове використання.

Ключові слова: Система, автопілот, розпізнавання перешкод, дрон розвідник, безпека, технології сенсорів, обробка даних, автономний транспорт, рятувальні операції, цивільне та військове використання.

ABSTRACT

An explanatory note to a bachelor's thesis consists of an introduction, three sections, conclusions, a list of sources used, and two appendices. The total volume of work is 81 pages, of which 50 pages of the main part with 11 figures, 1 tables, 8 names of the list of used sources and two appendices.

The object of the study – is the obstacle recognition processes by autopilot systems.

The subject of the research – is the methods and algorithms of recognition of obstacles by systems with autopilot.

The problem addressed by the qualification work: Development and optimization of a model with autopilot functions for recognition and avoidance of obstacles. Existing models often have limited capabilities in terms of obstacle detection accuracy, insufficient reliability in changing environmental conditions, and difficulties in integration with other systems. The work is aimed at increasing the efficiency and reliability of autopilot systems for various applications.

Scope – Automation and improvement of autonomous vehicle control systems. The developed model with autopilot functions can be widely used in various fields, including security and rescue operations, civil and military use.

Keywords: System, autopilot, obstacle detection, reconnaissance drone, security, sensor technology, data processing, autonomous transport, rescue operations, civil and military use.

ЗМІСТ

ВСТУП 6

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ 8

РОЗДІЛ 1 АНАЛІЗ КОНЦЕПЦІЇ ТА ФУНКЦІОНУВАННЯ КІБЕРФІЗИЧНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ПЕРЕШКОД ДЛЯ АВТОПІЛОТУ 9

1.1. Означення та сутність концепції «Кіберфізична система розпізнавання перешкод для автопілоту» 9

1.2. Технологічні аспекти функціонування системи розпізнавання перешкод для автопілоту 10

1.3. Вплив системи розпізнавання перешкод для автопілоту на різні аспек.. 11

1.4. Аналіз існуючих досліджень з проблематики систем розпізнавання перешкод для автопілоту 12

1.5. Порівняння різних підходів 18

Висновки за розділом 1 22

РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ 24

2.1. Структура системи розпізнавання першкод для автопілоту 24

2.2. Вибір програмного забезпечення для реалізації моделі системи розпізнавання перешкод автопілоту 27

2.3. Вибір програмного забезпечення для реалізації програмного додатку керування 31

2.4. Вибір основи для створення фізичної моделі 34

Висновки за розділом 2 35

РОЗДІЛ 3 РЕАЛІЗАЦІЯ СИСТЕМИ 37

3.1. Моделювання основи для системи 37

3.2. Створення фізичної моделі та програмування алгоритмів аналізу, та прийняття рішень 40

3.2.1. Акумуляторний блок 40
3.2.2. Двигуни 43
3.2.3. Датчики 45
3.2.4. Програмування алгоритмів керування 47
3.2.5. Алгоритми розпізнавання перешкод 51
3.3. Компоненти системи 54
Висновки за розділом 3 54
ВИСНОВКИ 56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57
ДОДАТОКИ 58
Додаток А 58
Додаток Б 60
Додаток В 66
Додаток Г 74

ВСТУП

Актуальність роботи. У сучасному світі міські середовища стають все більш заповненими транспортом, що призводить до збільшення кількості дорожніх інцидентів та ускладнює безпечний та ефективний рух транспортних засобів. Пошкодження, які виникають через дорожні перешкоди, не тільки можуть призвести до людських постраждалих та матеріальних збитків, але й сповільнюють рух транспорту, створюючи затори та перешкоджаючи нормальному функціонуванню міського життя.

У цьому контексті розробка систем з автопілотом, які мають можливість розпізнавання та уникнення перешкод, набуває важливого значення. Ці системи можуть стати ключовими складовими вдосконалення безпеки та ефективності транспорту, дозволяючи транспортним засобам швидко та безпечно реагувати на небезпеки на шляху.

Дрони-розвідники, як частина системи, грають важливу роль у цивільних, рятувальних та військових цілях. Вони забезпечують можливість вчасного виявлення та оцінки небезпек для безпеки та оперативності дій. У цивільному секторі вони допомагають в розпізнаванні ризикованих зон та моніторингу дорожнього руху, у рятувальних операціях вони забезпечують пошук та рятування постраждалих, а військове використання включає розвідку та надання підтримки в оперативних діях.

Крім того, розвиток таких технологій має потенціал змінити уявлення про транспортне майбутнє. Автономні транспортні засоби, які використовують системи з автопілотом, можуть зменшити кількість дорожніх інцидентів, покращити ефективність руху та зменшити негативний вплив на довкілля.

Метою роботи є підвищення ефективності систем розпізнавання перешкод системами з автопілотом. Робота спрямована на створення ефективної системи розпізнавання та уникнення перешкод у реальному часі.

Це включає вивчення алгоритмів, технологій сенсорів та методів обробки даних, що забезпечать надійність і точність функціонування такої системи в різних умовах. Дослідження також має на меті оцінити потенціал застосування цієї моделі в різних сферах, від безпекових і рятувальних операцій до цивільного та військового використання, з акцентом на підвищення рівня безпеки, оперативності та ефективності.

Об'єкт дослідження – процеси розпізнавання перешкод системами з автопілотом.

Предмет дослідження – методи і алгоритми розпізнавання перешкод системами з автопілотом. Особлива увага приділяється оцінці ефективності, надійності та можливостям інтеграції цієї моделі в різні сфери, включаючи безпекові та рятувальні операції, цивільне та військове використання.

У процесі дослідження розв'язувалися наступні завдання:

1. Проаналізувати принципи функціонування систем з автопілотом та їх роль у забезпеченні безпеки і ефективності.
2. Вивчити сучасні технології сенсорів та методи обробки даних для розпізнавання та уникнення перешкод у реальному часі.
3. Розробити модель з функціями автопілоту.
4. Провести тестування та оцінку ефективності запропонованої моделі в різних умовах і сценаріях.
5. Оцінити вплив впровадження систем з автопілотом на розвиток автономного транспорту, зменшення кількості дорожньо-транспортних пригод та зниження навантаження на водіїв.
6. Виявити потенційні виклики та обмеження використання систем з автопілотом і можливості їх подальшого розвитку.

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

Python – високорівнева мова програмування загального призначення, що використовується для розробки програмного забезпечення, аналізу даних, автоматизації задач та створення веб-додатків.

Arduino – платформа для створення електронних проектів, що складається з апаратної частини (контролери) і програмного забезпечення. Використовується для розробки інтерактивних проектів і прототипів.

Arduino IDE – інтегроване середовище розробки для програмування мікроконтролерів Arduino. Дозволяє писати, компілювати та завантажувати код на плату Arduino.

Fusion 360 – програмне забезпечення для 3D-моделювання, CAD (Computer-Aided Design), CAM (Computer-Aided Manufacturing) та CAE (Computer-Aided Engineering). Використовується для проектування та симуляції механічних компонентів та систем.

API – (Application Programming Interface) інтерфейс програмування додатків. Набір функцій та протоколів для створення програмного забезпечення, що дозволяє різним програмам взаємодіяти між собою.

HTTP – (Hypertext Transfer Protocol) протокол передачі гіпертексту. Використовується для передачі даних у всесвітній павутині (World Wide Web) та є основою для обміну інформацією між веб-серверами та клієнтами.

Wi-Fi – (Wireless Fidelity) технологія бездротового зв'язку, що дозволяє електронним пристроям обмінюватися даними або підключатися до інтернету за допомогою радіохвиль.

3D – (Three-Dimensional) тривимірний. Зазвичай використовується для позначення об'єктів або моделей, що мають три виміри: довжину, ширину та висоту.

РОЗДІЛ 1
АНАЛІЗ КОНЦЕПЦІЇ ТА ФУНКЦІОНУВАННЯ КІБЕРФІЗИЧНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ПЕРЕШКОД ДЛЯ АВТОПІЛОТУ

1.1. Означення та сутність концепції «Кіберфізична система розпізнавання перешкод для автопілоту»

Кіберфізична система — це механізм, що контролюється або відстежується комп'ютерними алгоритмами і тісно пов'язаний з Інтернетом та його користувачами. В кіберфізичних системах програмне забезпечення тісно пов'язано з фізичними об'єктами. Компоненти кіберфізичної системи взаємодіють на різних часових та просторових рівнях та можуть мати різні, відмінні одна від одної моделі поведінки та взаємодіяти одна з одною різними шляхами, які можуть змінюватися в залежності від контексту. Прикладами кіберфізичних систем можна вважати розумні енергосистеми, безпілотні автомобільні системи, автоматизовані системи керування, робототехнічні системи. [1]

Система розпізнавання перешкод для автопілоту — це інтегрована система, яка поєднує в собі алгоритми обробки даних, сенсори та інші технологічні засоби для виявлення, аналізу та управління рухом автономних транспортних засобів з автопілотом.

Сутність концепції полягає в забезпеченні безпеки та ефективності руху транспортних засобів, особливо автономних, в умовах непередбачуваних обставин та наявності перешкод на дорозі. Ця система дозволяє автоматично виявляти перешкоди та уникати зіткнень шляхом розрахунку оптимальних шляхів об'їзду чи зупинки.

Однією з основних переваг є підвищення рівня безпеки на дорозі шляхом уникнення потенційно небезпечних ситуацій та аварійних ситуацій. Крім того,

така система сприяє покращенню ефективності руху, зменшує затори та підвищує потенціал для реалізації концепції автономного транспорту.

Важливими аспектами є точність і швидкість реакції системи на перешкоди, надійність виявлення та аналізу даних про оточуюче середовище, а також адаптивність до різних умов дорожнього руху. Для досягнення успіху в цій області необхідно поєднати сенсори та алгоритми обробки даних для створення ефективної та надійної системи розпізнавання перешкод.

1.2. Технологічні аспекти функціонування системи розпізнавання перешкод для автопілоту

Технологічні аспекти функціонування системи розпізнавання перешкод для автопілоту охоплюють використання передових технологій для забезпечення точного та ефективного виявлення та уникнення перешкод на шляху автономних транспортних засобів. Деякі з ключових технологій, які застосовуються в цій системі, включають:

Сенсори та камери: Використання різноманітних сенсорів, таких як радари, лідари, ультразвукові сенсори, камери та інші, для збору даних про оточуюче середовище. Ці сенсори дозволяють системі отримувати інформацію про перешкоди на дорозі, їх розташування та характеристики.

Алгоритми обробки даних: Використання спеціалізованих алгоритмів для аналізу та інтерпретації даних, отриманих від сенсорів. Ці алгоритми дозволяють системі розпізнавати перешкоди, визначати їх тип та величину, а також приймати рішення щодо подальших дій.

Комунікаційні технології: Використання бездротових технологій зв'язку, таких як 5G, для передачі даних між транспортним засобом та системою керування. Це дозволяє системі оперативно реагувати на зміни у середовищі та координувати рух транспортного засобу.

Штучний інтелект та машинне навчання: Використання алгоритмів штучного інтелекту для навчання системи розпізнавати та класифікувати перешкоди, а також для прийняття оптимальних рішень щодо уникнення них. Це дозволяє системі адаптуватися до різних умов дорожнього руху та навчатися на основі власного досвіду.

Інтеграція з іншими системами: Забезпечення взаємодії з іншими системами автомобіля, такими як системи керування, системи безпеки та інші, для координації дій та забезпечення повного керування транспортним засобом в умовах виявлення перешкод.

1.3. Вплив системи розпізнавання перешкод для автопілоту на різні аспекти

Вплив системи розпізнавання перешкод для автопілоту на різні аспекти може бути значним та багатогранним, сприяючи покращенню якості та ефективності транспортних операцій. Ось кілька конкретних способів, які ця система може впливати на різні аспекти:

Безпека: система розпізнавання перешкод для автопілоту дозволяє транспортним засобам виявляти та уникати перешкод на дорозі, що значно знижує ризик дорожньо-транспортних пригод та забезпечує безпеку для пасажирів та оточуючих. Система може аналізувати інформацію про перешкоди та вибирати оптимальні маршрути, що дозволяє уникнути небезпечних зон та забезпечити безпечну доставку вантажу чи пасажирів.

Комфорт: Уникнення перешкод допомагає забезпечити стабільний рух транспортного засобу, що збільшує комфорт пасажирів та знижує ризик виникнення дискомфорту під час поїздки.

Швидкість та ефективність: Уникнення перешкод дозволяє транспортним засобам рухатися без перерв, що призводить до скорочення часу подорожі та підвищення ефективності транспортних операцій.

Загальний вплив системи розпізнавання перешкод для автопілоту полягає у забезпеченні безпеки та ефективності транспортних операцій, що призводить до поліпшення якості перевезень та забезпечення комфорту для користувачів.

1.4. Аналіз існуючих досліджень з проблематики систем розпізнавання перешкод для автопілоту

На сучасному етапі компанії, що спеціалізуються на розробці систем розпізнавання перешкод для автопілоту, проводять активні дослідження та впроваджують нові технології з метою підвищення безпеки та ефективності автономного транспорту. Однією з ключових проблем, з якою вони стикаються, є розпізнавання та уникнення різних типів перешкод у реальному часі.

Провідні компанії у цій галузі включають:

- Tesla, Inc.
- Waymo LLC (підрозділ Alphabet Inc.)
- Cruise Automation
- Uber Technologies Inc. (Advanced Technologies Group)
- Argo AI (підприємство, спільно засноване Ford та Volkswagen)

Ці компанії активно досліджують та розвивають різні аспекти автономного транспорту та систем розпізнавання перешкод. Вони зосереджуються на розпізнаванні різноманітних видів перешкод, таких як інші транспортні засоби, пішоходи, будівлі, дорожні знаки та сигнальні системи, та розвивають відповідні алгоритми та системи розпізнавання для забезпечення безпеки та надійності автономних транспортних засобів.

Ключові аспекти аналізу існуючих досліджень включають:

Технічні аспекти розпізнавання перешкод для автопілоту охоплюють ряд ключових елементів, які включають алгоритми обробки даних, методи машинного навчання та сенсорні технології:

Алгоритми обробки даних: Вони відіграють ключову роль у функціонуванні системи розпізнавання перешкод, оскільки дозволяють обробляти великі обсяги даних, отриманих від різних сенсорів та джерел. Ці алгоритми включають в себе методи фільтрації, сегментації, класифікації та прогнозування, що дозволяють ефективно виявляти та аналізувати перешкоди на дорозі.

Методи машинного навчання: Вони стають все більш популярними в галузі розпізнавання перешкод, оскільки дозволяють системам автопілоту навчитися визначати перешкоди на основі великої кількості навчальних даних. Методи машинного навчання, такі як нейронні мережі, дерева рішень та метод опорних векторів, можуть бути використані для класифікації та визначення ризику зіткнення з перешкодою.

Сенсорні технології: Сенсори відіграють важливу роль у зборі даних про навколишнє середовище, що допомагає системі автопілоту розпізнавати перешкоди. Серед найпоширеніших сенсорів можна виділити радари, лідари, камери та ультразвукові сенсори, які спільно використовуються для створення детальної моделі оточуючого середовища.

Комплексне використання цих технічних рішень дозволяє автономним транспортним засобам ефективно розпізнавати та уникати перешкод на дорозі, забезпечуючи безпеку та надійність руху. Вдосконалення цих технологій в майбутньому може призвести до подальшого покращення функціональності та безпеки автопілотних систем.

Випробування та результати є ключовою частиною аналізу існуючих систем розпізнавання перешкод для автопілоту. Ця частина оцінює

ефективність та надійність систем в умовах реального світу, враховуючи різноманітні фактори, такі як погодні умови, швидкість руху та типи доріг:

Результати випробувань: Під час випробувань оцінюються різні параметри роботи систем розпізнавання перешкод, такі як швидкість та точність реакції на перешкоди. Результати випробувань включають аналіз часу реакції, відстані до перешкоди, точності прогнозування траєкторії та інші параметри, які визначають ефективність системи.

Різні умови: Випробування проводяться в різних умовах, щоб оцінити реакцію системи на різні сценарії. Це включає випробування в різних погодних умовах, таких як дощ, сніг, туман або яскраве сонце, а також на різних типах доріг, таких як швидкісні магістралі, міські вулиці чи гірські серпантини.

Швидкість руху: Важливо враховувати швидкість руху автономного транспорту, оскільки вона впливає на час реакції системи на перешкоди та можливість уникнення небезпечних ситуацій. Випробування проводяться на різних швидкостях руху, включаючи міські перегони з низькою швидкістю та швидкісні траси з високою швидкістю.

Ретельний аналіз результатів випробувань у різних умовах допомагає визначити сильні та слабкі сторони існуючих систем розпізнавання перешкод для автопілоту. Це дозволяє розробникам вдосконалювати та оптимізувати системи з метою забезпечення найвищого рівня безпеки та ефективності в різних умовах експлуатації.

Недоліки та виклики існуючих систем розпізнавання перешкод для автопілотів грають значну роль у вдосконаленні цих технологій. Детальний аналіз цих аспектів дозволяє зрозуміти поточні обмеження та визначити напрямки подальшого розвитку:

Точність розпізнавання: Одним з найважливіших недоліків є точність розпізнавання перешкод. Це означає, наскільки ефективно система визначає та

класифікує перешкоди на шляху. Наприклад, іноді системи можуть неправильно інтерпретувати неперешкоди як перешкоди або навпаки, що може призвести до неправильної реакції або навіть до аварійної ситуації.

Складність реалізації: Іншою проблемою є складність реалізації систем розпізнавання перешкод. Деякі підходи можуть бути технічно складними або вимагати великої кількості обчислень, що ускладнює їхнє впровадження та ефективне функціонування в реальному часі.

Управління в складних дорожніх сценаріях: Комплексні дорожні сценарії, такі як рух на перехрестях, будівництво або дорожні роботи, можуть бути викликом для існуючих систем розпізнавання перешкод. У таких сценаріях системам необхідно враховувати багато факторів, таких як рух інших транспортних засобів, реакції пішоходів та робочі зони, що потребує високої точності та швидкості реакції.

Недоліки зв'язку: Недолік зв'язку може призвести до затримок у передачі даних про перешкоди або навіть до втрати зв'язку з системою управління. Це може ускладнити реакцію системи на потенційні небезпечні ситуації та знизити загальний рівень безпеки.

Аналіз цих недоліків та викликів допомагає визначити ключові аспекти, які потребують удосконалення та удосконалення. Розробники можуть використовувати цю інформацію для поліпшення точності, ефективності та безпеки систем розпізнавання перешкод для автопілоту.

Переваги та досягнення існуючих систем розпізнавання перешкод для автопілоту відіграють критичну роль у визначенні їхньої ефективності та цінності для сучасного транспортного середовища. Детальний аналіз цих переваг допомагає розуміти вплив цих систем на безпеку та ефективність руху:

Зменшення аварійності: Однією з основних переваг є зменшення ризику аварій за рахунок розпізнавання та уникнення перешкод. Системи можуть

ефективно виявляти небезпечні ситуації та реагувати на них швидко та точно, запобігаючи потенційним аваріям.

Підвищення безпеки: Іншою важливою перевагою є підвищення загального рівня безпеки на дорозі для всіх учасників руху. За допомогою систем розпізнавання перешкод можна забезпечити швидку реакцію на небезпечні ситуації та запобігти потенційним аваріям або травмам.

Покращення ефективності руху: Системи розпізнавання перешкод можуть сприяти покращенню потоку транспорту шляхом забезпечення більш ефективного управління рухом. Вони можуть допомагати уникати заторів та оптимізувати маршрути з метою зменшення часу в дорозі.

Підтримка автономного транспорту: Існуючі системи розпізнавання перешкод грають важливу роль у розвитку автономного транспорту, допомагаючи забезпечити безпеку та ефективність автопілотних систем. Вони створюють фундаментальні технологічні блоки, необхідні для реалізації повноцінного автономного руху.

Покращення досвіду користувача: За допомогою систем розпізнавання перешкод можна покращити досвід користувача та зробити подорожі більш зручними та безпечними. Це може включати зменшення стресу від водіння та забезпечення більш комфортних умов для пасажирів.

Аналіз цих переваг та досягнень є важливим для оцінки впливу існуючих систем розпізнавання перешкод та визначення їхньої цінності для подальшого розвитку та вдосконалення.

Напрямки подальшого розвитку систем розпізнавання перешкод для автопілоту становлять важливу складову в удосконаленні ефективності та безпеки автономного руху. Аналіз цих напрямків дозволяє визначити перспективні технології та стратегії, які сприятимуть подальшому розвитку цих систем:

Використання штучного інтелекту (AI) та машинного навчання (ML): Подальший розвиток систем розпізнавання перешкод включає в себе збільшення використання штучного інтелекту та методів машинного навчання для покращення їхньої точності та надійності. Розробка більш складних алгоритмів, які здатні адаптуватися до різних умов дорожнього руху, є ключовим напрямком розвитку.

Розширення сенсорного спектру: Розвиток систем розпізнавання перешкод також передбачає розширення сенсорного спектру для покращення їхньої здатності виявляти та класифікувати перешкоди. Це може включати використання нових типів сенсорів, таких як лідари високої роздільної здатності, радари та камери зі штучним інтелектом.

Розробка спеціалізованих алгоритмів: Для покращення точності та швидкості розпізнавання перешкод важливо розробляти спеціалізовані алгоритми, які враховують особливості дорожнього середовища та типів перешкод. Це може включати адаптацію алгоритмів до різних швидкостей руху, типів доріг та погодних умов.

Інтеграція з іншими системами: Подальший розвиток систем розпізнавання перешкод також передбачає їхню інтеграцію з іншими системами транспортного середовища, такими як системи навігації та комунікації. Це дозволить створити більш комплексні та розумні системи, які здатні ефективно виявляти та уникати перешкод.

Стандартизація та регулювання: Важливим аспектом подальшого розвитку є створення стандартів та регулятивного середовища, яке сприятиме впровадженню та розвитку систем розпізнавання перешкод. Це дозволить стандартизувати процеси розробки та випробування, а також забезпечить відповідність цих систем вимогам безпеки та ефективності.

Автоматизація та самооптимізація: Одним з перспективних напрямків розвитку є створення систем, які здатні автоматично оптимізувати свою

роботу на основі навчання та досвіду. Це може включати в себе автоматичне вдосконалення алгоритмів та підтримку навчання в реальному часі.

Ці напрямки подальшого розвитку визначають ключові технологічні та наукові виклики, які стоять перед дослідниками та розробниками в галузі розпізнавання перешкод для автопілоту.

Аналіз існуючих досліджень у цій області допомагає визначити найбільш ефективні підходи та напрямки для подальшого розвитку систем розпізнавання перешкод для автопілоту, сприяючи подальшому покращенню безпеки та ефективності автономного транспорту.

1.5. Порівняння різних підходів

Порівняння різних підходів до розпізнавання перешкод для автопілоту є важливим етапом у визначенні оптимального рішення для покращення безпеки та ефективності автономного руху. Цей процес включає оцінку переваг та недоліків кожного підходу з урахуванням їхніх технічних особливостей, ефективності та вартості реалізації. Основні аспекти порівняння включають:

Технічні особливості: Цей аспект включає вивчення різних технічних характеристик кожного підходу до розпізнавання перешкод. Наприклад, системи можуть використовувати різні типи сенсорів, такі як лідари, радари, камери або комбіновані сенсорні системи. Аналізується також використання різних алгоритмів розпізнавання, таких як машинне навчання, глибокі нейронні мережі, класифікатори та фільтри. Кожен підхід має свої переваги та обмеження залежно від технічних характеристик, тому важливо ретельно проаналізувати їх для вибору найбільш підходящого рішення.

Ефективність: Під час оцінки ефективності порівнюються різні підходи щодо їхньої здатності ефективно виявляти та уникати перешкод у різних

умовах. Враховуються такі фактори, як точність розпізнавання, швидкість реакції на перешкоди, здатність працювати в різних погодних умовах та на різних типах доріг. Ефективність системи може бути оцінена шляхом проведення випробувань у реальних умовах або за допомогою симуляційних моделей.

Вартість реалізації: Вартість реалізації включає витрати на обладнання, розробку програмного забезпечення, випробування, підтримку та інші витрати, пов'язані з впровадженням системи. Різні підходи можуть мати різну вартість реалізації залежно від використаної технології, складності реалізації та інших факторів. Порівняння вартості реалізації допомагає вибрати економічно доцільне рішення.

Переваги та недоліки: Кожен підхід має свої переваги та недоліки, які слід враховувати при порівнянні. Наприклад, деякі системи можуть бути більш точними в розпізнаванні перешкод, але мати вищі витрати на реалізацію та більшу складність управління. Інші системи можуть бути менш точними, але більш доступними за ціною та менш енергоефективними. Ретельний аналіз переваг та недоліків кожного підходу допомагає зрозуміти їхню придатність для конкретних завдань.

Приклади порівняння різних підходів до розпізнавання перешкод для автопілоту, враховуючи технічні особливості, ефективність та вартість реалізації:

Використання лідарів проти камер:

- Технічні особливості:

  Лідари: Висока точність, але висока вартість і складність інтеграції.

  Камери: Відносно низька вартість, але менш точне в різних умовах освітлення.

- Ефективність:

Лідари: Висока точність у всіх умовах, але обмеження у визначенні кольорів та текстур.

Камери: Можуть виявляти перешкоди, але менш точно, особливо в умовах обмеженого освітлення.

- Вартість реалізації:

Лідари: Висока вартість обладнання та високі витрати на обробку даних.

Камери: Відносно низька вартість, особливо для систем з використанням штучного інтелекту для обробки зображень.

Використання радарів проти камер:

- Технічні особливості:

Радари: Висока надійність у різних погодних умовах, але обмежені можливості визначення форми та текстури об'єктів.

Камери: Здатні розпізнавати форму та текстуру об'єктів, але менш надійні в умовах поганої видимості.

- Ефективність:

Радари: Висока ефективність в умовах обмеженого видимості, таких як туман або дощ.

Камери: Ефективні у добрих умовах видимості, але можуть мається проблеми з розпізнаванням у складних погодних умовах.

- Вартість реалізації:

Радари: Зазвичай дорожчі у встановленні та підтримці, але можуть бути ефективні у відношенні до точності та надійності.

Камери: Загалом менш витратні на встановлення, але можуть вимагати додаткових витрат на програмне забезпечення для обробки зображень.

Використання машинного навчання проти класичних алгоритмів:

- Технічні особливості:

Машинне навчання: Здатність до автоматичного визначення шаблонів та адаптації до нових умов.

Класичні алгоритми: Більш прості у реалізації та розумінні, але можуть бути менш ефективними у складних сценаріях.

- Ефективність:

Машинне навчання: Здатність до адаптації та навчання на основі даних, але може вимагати більшого обсягу даних для тренування.

Класичні алгоритми: Можуть пропонувати більш точні результати у визначених умовах, але можуть бути менш гнучкими у вирішенні нових завдань.

- Вартість реалізації:

Машинне навчання: Зазвичай потребує більшого обсягу обчислювальних ресурсів для тренування та використання навчених моделей, що може підвищити вартість проекту.

Класичні алгоритми: Зазвичай менш вимогливі до обчислювальних ресурсів, що може зменшити загальні витрати на розробку та впровадження.

Висновки за розділом 1

У першому розділі було здійснено всебічний аналіз концепції та функціонування кіберфізичної системи розпізнавання перешкод для автопілоту, що є важливою складовою сучасних автономних транспортних засобів.

Означення та сутність концепції «Кіберфізична система розпізнавання перешкод для автопілоту». У цьому підрозділі було визначено основні характеристики та компоненти кіберфізичної системи розпізнавання перешкод. Встановлено, що такі системи інтегрують фізичні об'єкти з цифровими технологіями, забезпечуючи взаємодію між сенсорами, алгоритмами обробки даних та виконавчими механізмами для забезпечення автономного руху транспортного засобу. Особлива увага приділена важливості точної та швидкої обробки даних для забезпечення безпеки та ефективності роботи системи.

Технологічні аспекти функціонування системи розпізнавання перешкод для автопілоту. У цьому підрозділі розглянуто ключові технології, що забезпечують функціонування систем розпізнавання перешкод. Зокрема, проаналізовано різні типи сенсорів (камери, лідари, радари, ультразвукові датчики) та їх роль у формуванні вхідних даних. Описано методи обробки інформації, включаючи алгоритми машинного навчання та штучного інтелекту, які використовуються для аналізу та інтерпретації даних, що надходять від сенсорів.

Вплив системи розпізнавання перешкод для автопілоту на різні аспекти. Визначено вплив використання систем розпізнавання перешкод на безпеку, комфорт та енергоефективність транспортних засобів. Показано, що впровадження таких систем значно підвищує рівень безпеки на дорогах, зменшує кількість аварій та інцидентів. Крім того, системи автопілоту сприяють підвищенню комфорту пасажирів за рахунок плавного та

передбачуваного руху транспортного засобу. Також було відзначено потенціал зменшення споживання енергії завдяки оптимізації маршрутів та керування швидкістю.

Аналіз існуючих досліджень з проблематики систем розпізнавання перешкод для автопілоту. Проведено огляд сучасних досліджень у сфері систем розпізнавання перешкод для автопілотів. Виявлено основні досягнення та проблеми, що стоять перед розробниками таких систем. Описано приклади успішного впровадження технологій у різних проектах та проаналізовано недоліки та обмеження, що виникають під час їх використання.

Порівняння різних підходів. Проведено порівняння різних підходів до розробки та впровадження систем розпізнавання перешкод. Розглянуто переваги та недоліки використання різних типів сенсорів та алгоритмів обробки даних. Визначено, що комбінування кількох типів сенсорів та використання гібридних методів обробки даних забезпечує найвищу ефективність та надійність функціонування системи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Структура системи розпізнавання першкод для автопілоту

Система розпізнавання перешкод для автопілоту включає в себе комплекс компонентів, які забезпечують взаємодію між фізичними датчиками та обчислювальними алгоритмами. Структура цієї системи складається з наступних елементів:

Ультразвукові датчики відстані HC-SR04:

Рисунок 2.1 – Ультразвукові датчики відстані HC-SR04

Система використовує два ультразвукові датчики відстані HC-SR04 для виявлення перешкод. Ці датчики генерують ультразвукові сигнали та вимірюють час їх повернення після відбиття від об'єкта, дозволяючи точно визначати відстань до перешкод. Розміщення датчиків дозволяє охопити різні напрями та забезпечити комплексний огляд навколишнього середовища.

Драйвер двигунів L9110S:

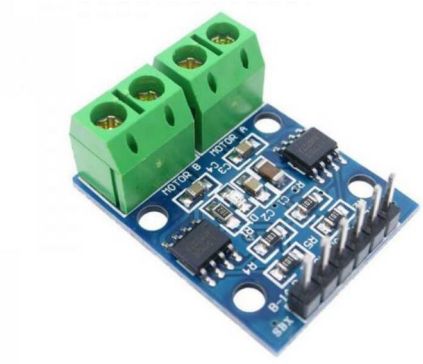


Рисунок 2.2 – Драйвер двигунів L9110S

Драйвер двигунів L9110S використовується для керування двома двигунами постійного струму. Цей компонент приймає сигнали від головної плати та забезпечує необхідну потужність для двигунів, дозволяючи регулювати швидкість та напрямок руху транспортного засобу.

Двигуни 5V:

Система включає два двигуни 5V, які забезпечують рух транспортного засобу. Вони відповідають за привід коліс та дозволяють здійснювати маневри для уникнення перешкод, керовані сигналами від драйвера двигунів.

Акумуляторні батареї:

Система використовує п'ять акумуляторних батарей ємністю 650 міліампер та напругою 3,7V, з'єднаних паралельно. Паралельне з’єднання утворює теоретичну ємність поєднаних акумуляторів у 3250 міліампер. Це забезпечує стабільне живлення всіх компонентів системи, дозволяючи їм працювати тривалий час без необхідності частої підзарядки.

Плата заряду акумуляторних батарей:

Для забезпечення безперервного живлення система включає плату заряду акумуляторних батарей. Ця плата дозволяє безпечно заряджати акумулятори, контролюючи рівень заряду та запобігаючи перенапруженням або перегріву.

Головна плата ESP8266:

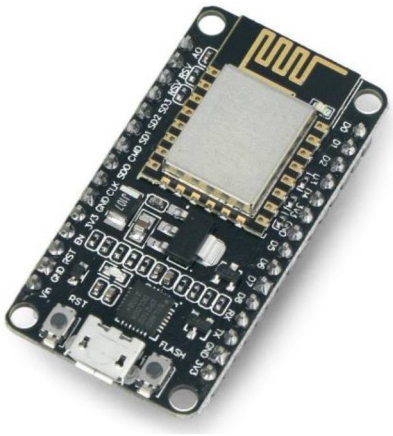


Рисунок 2.3 – Головна плата ESP8266

Основний компонент системи - головна плата ESP8266, яка відповідає за обробку даних, комунікацію та виконання алгоритмів. Ця плата приймає дані від ультразвукових датчиків, обробляє їх за допомогою вбудованих алгоритмів розпізнавання перешкод та керує драйвером двигунів для відповідної реакції на виявлені перешкоди. ESP8266 також забезпечує можливість віддаленого управління та координації дій через Wi-Fi.

Загальна структура системи:

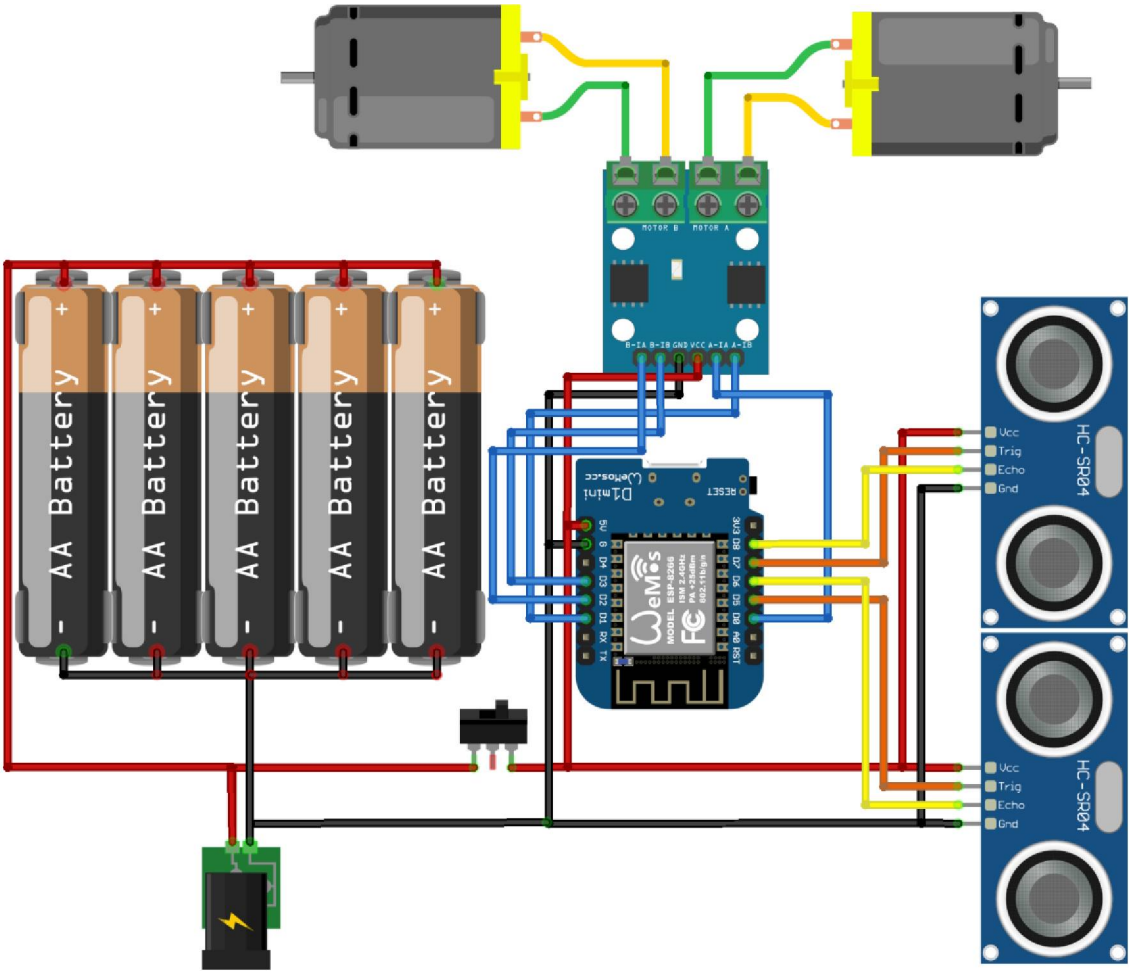


Рисунок 2.4 – Загальна структура системи

Всі вищезазначені компоненти інтегровані в єдину систему, яка забезпечує ефективне розпізнавання та уникнення перешкод у реальному часі. Ультразвукові датчики виявляють перешкоди, дані передаються на головну плату, яка обробляє їх та приймає рішення щодо керування двигунами через драйвер. Акумуляторні батареї забезпечують необхідну енергію для всіх компонентів, а плата заряду дозволяє підтримувати їх в робочому стані.

2.2. Вибір програмного забезпечення для реалізації моделі системи розпізнавання перешкод автопілоту

Для успішної реалізації моделі системи розпізнавання перешкод автопілоту важливо правильно підібрати програмне забезпечення, яке забезпечить ефективне оброблення даних з датчиків, виконання алгоритмів розпізнавання та керування транспортним засобом. Вибір програмного забезпечення включає кілька ключових аспектів:

Простота та зручність використання:

Arduino IDE було обрано завдяки його інтуїтивно зрозумілому інтерфейсу, який дозволяє легко писати, компілювати та завантажувати код на мікроконтролери. Це забезпечує швидкий та зручний процес розробки, який підходить як для початківців, так і для досвідчених розробників.

- Інтуїтивно зрозумілий інтерфейс: Arduino IDE має простий і зрозумілий користувацький інтерфейс, який дозволяє швидко знайти необхідні інструменти та функції.
- Швидкий процес розробки: Завдяки зручним функціям компіляції та завантаження коду, розробники можуть швидко тестувати та налагоджувати свої програми.
- Підтримка новачків та досвідчених користувачів: Arduino IDE є доступним для новачків завдяки своїй простоті, але також пропонує достатньо можливостей для просунутих користувачів.

Широка підтримка апаратного забезпечення:

Arduino IDE підтримує різні мікроконтролери, включаючи ESP8266, що є центральним елементом цієї системи. Це дозволяє використовувати всі можливості мікроконтролера ESP8266, зокрема Wi-Fi з'єднання для віддаленого управління та збору даних.

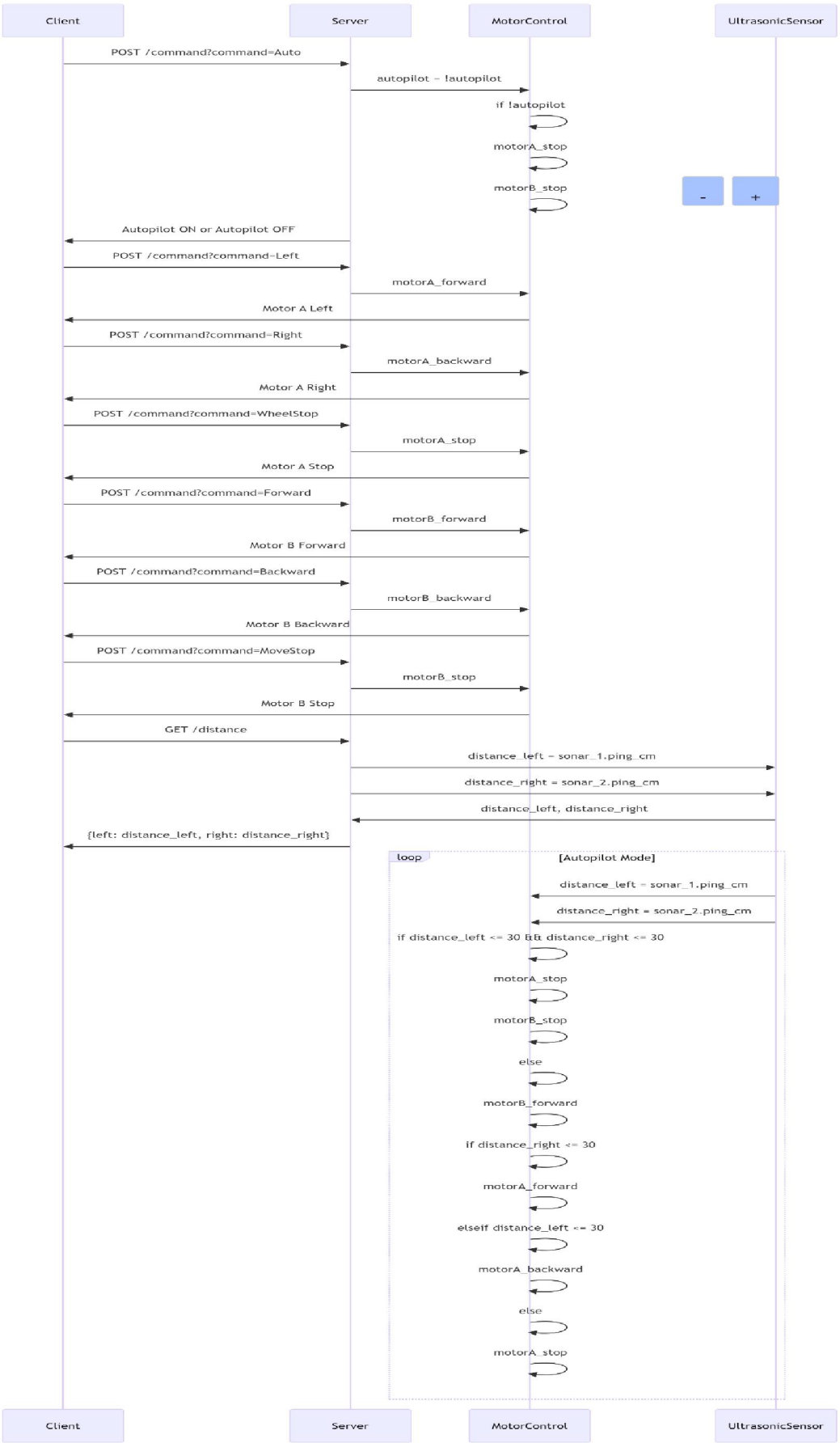
- Підтримка ESP8266: ESP8266 є важливим елементом системи, оскільки він забезпечує бездротове з'єднання і можливість інтеграції з іншими компонентами.
- Різноманітність платформ: Arduino IDE підтримує багато різних мікроконтролерів і апаратних платформ, що робить його універсальним інструментом для розробки різних проектів.
- Гнучкість: Широка підтримка апаратного забезпечення дозволяє розробникам легко адаптувати свої проекти під різні вимоги і умови експлуатації.

Розширена бібліотека функцій:

Arduino IDE надає великий набір бібліотек для роботи з різноманітними сенсорами та модулями. У цій системі використовуються наступні бібліотеки:

- ESP8266WiFi.h: Для роботи з Wi-Fi, дозволяючи мікроконтролеру підключатися до мережі, налаштовувати точку доступу та обмінюватися даними з іншими пристроями.
- ESP8266WebServer.h: Для створення веб-сервера, що дозволяє керувати системою та отримувати дані через веб-інтерфейс.
- NewPing.h: Для управління ультразвуковими сенсорами, забезпечуючи точне вимірювання відстані до об'єктів.

Ці бібліотеки значно спрощують розробку, надаючи готові рішення для складних задач, таких як налаштування бездротового зв'язку або обробка сигналів з сенсорів.


Client
Server
MotorControl
UltrasonicSensor
POST /command?command=Auto
autopilot = !autopilot
if !autopilot
motorA_stop
motorB_stop
Autopilot ON or Autopilot OFF
POST /command?command=Left
motorA_forward
Motor A Left
POST /command?command=Right
motorA_backward
Motor A Right
POST /command?command=WheelStop
motorA_stop
Motor A Stop
POST /command?command=Forward
motorB_forward
Motor B Forward
POST /command?command=Backward
motorB_backward
Motor B Backward
POST /command?command=MoveStop
motorB_stop
Motor B Stop
GET /distance
distance_left = sonar_1.ping_cm
distance_right = sonar_2.ping_cm
distance_left, distance_right
{left: distance_left, right: distance_right}
loop
[Autopilot Mode]
distance_left = sonar_1.ping_cm
distance_right = sonar_2.ping_cm
if distance_left <= 30 && distance_right <= 30
motorA_stop
motorB_stop
else
motorB_forward
if distance_right <= 30
motorA_forward
elseif distance_left <= 30
motorA_backward
else
motorA_stop
-
+
Client
Server
MotorControl
UltrasonicSensor

Програмне забезпечення системи реалізовано наступним чином:

Підключення бібліотек та оголошення змінних: На початку коду підключаються необхідні бібліотеки для роботи з Wi-Fi, веб-сервером та ультразвуковими сенсорами. Також оголошуються глобальні змінні для зберігання налаштувань Wi-Fi, керування двигунами та даних з сенсорів.

Налаштування функцій обробки команд та вимірювань: Функції обробки команд («handleCommand()») та вимірювань («handleDistance()») відповідають за обробку вхідних команд від клієнтів та запити на вимірювання відстані. Ці функції забезпечують інтерактивність системи, дозволяючи користувачам керувати роботою автопілоту та отримувати актуальні дані про відстань до перешкод.

Налаштування та запуск сервера: У функції «setup()» виконується налаштування пін-кодів для керування двигунами, конфігурація Wi-Fi у режимі точки доступу та запуск веб-сервера. Цей етап є критично важливим для забезпечення віддаленого доступу до системи та інтерактивного керування.

Основний цикл роботи: У функції «loop()» обробляються запити клієнтів та реалізується логіка автопілоту. Використовуючи дані з ультразвукових сенсорів, автопілот приймає рішення про напрямок руху, коригуючи швидкість та напрямок руху транспортного засобу для уникнення перешкод.

Цей вибір програмного забезпечення та структура коду дозволяють ефективно обробляти дані з датчиків, виконувати алгоритми розпізнавання та керувати транспортним засобом, забезпечуючи високу ефективність та надійність роботи системи. Arduino IDE, завдяки своїй простоті, підтримці широкого спектру апаратного забезпечення та великому набору бібліотек, є оптимальним вибором для реалізації цієї системи.

2.3. Вибір програмного забезпечення для реалізації програмного додатку керування

Вибір мови програмування та фреймворку:

Python був обраний як мова програмування для розробки програмного додатку керування з кількох причин. По-перше, Python відомий своєю простотою в навчанні та використанні, що робить його ідеальним вибором для широкого кола розробників, включаючи початківців. Він має зрозумілий синтаксис та велику кількість вбудованих функцій і бібліотек, що полегшує розробку та прискорює процес програмування.

Друга причина - широкі можливості Python. Він підтримує різноманітні структури даних та має велику кількість сторонніх бібліотек, які можуть бути використані для різних задач, включаючи створення графічних інтерфейсів, роботу з мережами та обробку даних.

Щодо вибору фреймворку, Kivy був обраний через свою крос-платформенність і зручний спосіб опису інтерфейсу користувача за допомогою мови розмітки Kivy. Крос-платформенність означає, що один і той же код може бути використаний для створення програм для різних операційних систем, що є важливим фактором для забезпечення універсальності додатку. Крім того, Kivy надає розширені можливості для створення графічного інтерфейсу, такі як анімація та реагування на дотик, що робить його привабливим вибором для створення додатків зі складним інтерфейсом.

Інтерфейс користувача:

Головна мета інтерфейсу користувача - забезпечити зручний та інтуїтивно зрозумілий спосіб керування транспортним засобом. Для досягнення цієї мети на головному вікні додатку розміщені кнопки керування, такі як "вперед", "назад", "ліворуч", "праворуч", а також кнопка для активації

режиму автопілота. Це дозволяє користувачу швидко та ефективно взаємодіяти з програмою та керувати рухом транспортного засобу.

Крім того, для візуального відображення відстані до перешкод використовується графічне зображення піраміди. Кожен рівень піраміди представляє певний діапазон відстані, а його колір вказує на ступінь небезпеки: від зеленого (безпечно) до червоного (небезпечно). Це дозволяє користувачеві швидко оцінити ситуацію та прийняти відповідні дії.

Відправка команд на мікроконтролер:

Коли користувач натискає кнопку керування на інтерфейсі, програма відправляє відповідну команду на мікроконтролер ESP8266 через WiFi-з'єднання. Ці команди передаються як HTTP-запити, які мікроконтролер може інтерпретувати та виконати відповідні дії, такі як рух вперед, назад, ліворуч, праворуч або активація режиму автопілота.

Цей механізм керування дозволяє користувачеві взаємодіяти з транспортним засобом безпосередньо з програмного додатку, що забезпечує зручність та ефективність використання.

Оновлення відображення даних:

Оновлення відображення даних, таких як відстань до перешкод, відбувається періодично через HTTP-запити до мікроконтролера. Після отримання цих даних програма оновлює відповідні елементи інтерфейсу, щоб користувач міг завжди отримувати актуальну інформацію про стан транспортного засобу.

Цей процес автоматично повторюється через певний інтервал часу, забезпечуючи постійне оновлення інформації та реагування на зміни у середовищі. Такий підхід дозволяє програмі оперативно адаптуватися до змін у відстані до перешкод та надає користувачеві достовірну та зрозумілу інформацію для прийняття рішень.

Безпека з'єднання:

Однією з ключових аспектів розробки програмного додатку є забезпечення безпеки з'єднання між програмою та мікроконтролером ESP8266. Для цього використовується механізм автентифікації з логіном та паролем, що дозволяє перевіряти ідентичність користувача перед передачею команд на керування транспортним засобом.

Це механізм додає додатковий рівень безпеки до системи, запобігаючи несанкціонованому доступу до керування транспортним засобом. Такий підхід забезпечує захист від потенційних кібератак і зберігає дані в безпеці.

Можливості розширення:

Один з головних переваг програмного додатку полягає в його модульній структурі, що дозволяє легко розширювати його функціональні можливості. Наприклад, розробник може додати нові режими керування, розширити графічний інтерфейс, покращити алгоритми обробки даних або інтегрувати додаток з іншими пристроями через різні протоколи зв'язку.

Такий підхід дозволяє забезпечити максимальну гнучкість та адаптивність програмного додатку до змінних умов та вимог користувачів. Крім того, він сприяє постійному розвитку та вдосконаленню системи, забезпечуючи її актуальність у довгостроковій перспективі.

Ці аспекти говорять про виважений та комплексний підхід до проектування та реалізації програмного додатку керування, який спрямований на забезпечення якості, ефективності та безпеки в умовах реального використання.

2.4. Вибір основи для створення фізичної моделі

Для створення фізичної моделі системи керування транспортним засобом було обрано основу від іграшкової машинки з коліщатками. Цей вибір був зроблений з огляду на кілька ключових переваг, таких як компактність, простота модифікації та доступність компонентів. Основу іграшкової машинки легко адаптувати під вимоги проекту, що дозволяє зосередитися на розробці та тестуванні програмного забезпечення і систем керування, мінімізуючи витрати та складність процесу.

Використання іграшкової машинки як основи

Іграшкова машинка забезпечує всі необхідні механічні компоненти для створення функціональної моделі транспортного засобу. Зокрема, вона має:

Колеса: Наявність чотирьох коліс забезпечує стабільність і можливість руху в різних напрямках. Колеса мають достатній розмір та якість, щоб забезпечити плавний рух по різних типах поверхонь.

Шасі: Шасі іграшкової машинки виготовлено з міцного пластику та металу, що витримує навантаження від встановлених електронних компонентів. Воно забезпечує основу для кріплення всіх необхідних частин, таких як мотори, сенсори та мікроконтролер.

Мотори: Для забезпечення руху використовуються електромотори, які можуть обертати колеса, дозволяючи машинці рухатися вперед, назад, а також здійснювати повороти.

Редуктори з шестернями: Використання редукторів з шестернями дозволяє керувати рухом задньої осі для прямолінійного руху і передньої осі для здійснення поворотів. Ці механізми забезпечують необхідний момент обертання та швидкість, що важливо для керування транспортним засобом.

Висновки за розділом 2

У другому розділі було розглянуто процес проектування системи розпізнавання перешкод для автопілоту, що включає вибір структури системи, програмного забезпечення для реалізації моделі та програмного додатку керування, а також основи для створення фізичної моделі.

Структура системи розпізнавання перешкод для автопілоту. У цьому підрозділі було визначено ключові компоненти системи, включаючи сенсори, обчислювальні блоки та виконавчі механізми. Було розроблено загальну архітектуру системи, яка забезпечує ефективне розпізнавання та уникнення перешкод. Ця структура включає інтеграцію різних типів сенсорів, таких як камери, лідари та радари, які забезпечують всебічне виявлення об'єктів у навколишньому середовищі.

Вибір програмного забезпечення для реалізації моделі системи розпізнавання перешкод автопілоту. У цьому підрозділі проведено аналіз різних програмних платформ для реалізації моделі системи. Було вибрано найбільш підходящі інструменти, зокрема, Python для розробки алгоритмів машинного навчання та обробки даних, а також Arduino IDE для програмування контролерів. Пояснено, чому ці програмні платформи є оптимальними для реалізації даної системи, враховуючи їхню гнучкість, підтримку спільноти та наявність бібліотек.

Вибір програмного забезпечення для реалізації програмного додатку керування. У цьому підрозділі розглянуто різні програмні засоби для створення інтерфейсу керування системою. Було обрано середовище розробки, яке дозволяє створити інтуїтивно зрозумілий та функціональний додаток для користувачів. Вибір програмного забезпечення базувався на критеріях зручності використання, можливостей інтеграції з іншими системами та забезпечення надійної роботи.

Вибір основи для створення фізичної моделі. У цьому підрозділі розглянуто можливі варіанти платформи для фізичної реалізації моделі. Було визначено, що використання Arduino як основи для створення фізичної моделі є оптимальним рішенням, зважаючи на її широкі можливості, доступність та підтримку різних модулів і сенсорів. Пояснено, як саме ця платформа забезпечить ефективне функціонування всієї системи, включаючи збір даних від сенсорів, обробку інформації та управління виконавчими механізмами.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Моделювання основи для системи

Для інтеграції електронних компонентів та забезпечення їх стабільного кріплення було вирішено використати технологію 3D-друку. Зокрема, були змодельовані та надруковані наступні компоненти:

Кріплення для мікроконтролера:

Проектування: Спеціальні кріплення були спроектовані з урахуванням розмірів мікроконтролера ESP8266 та місця його розташування. Кріплення забезпечують легкий доступ до портів мікроконтролера для підключення сенсорів та живлення.

Друк: Після проектування кріплення було надруковано на 3D-принтері з використанням міцного пластику. Це забезпечує надійність кріплення та захист мікроконтролера від механічних пошкоджень.

Тримачі для сенсорів:

Проектування: Тримачі були спроектовані таким чином, щоб сенсори могли бути встановлені на оптимальних місцях для виявлення перешкод. Це включає бічні розташування сенсорів таким чином, щоб фронтальні перешкоди також були виявлені, та були вчасно прийняті дії алгоритмами керування.

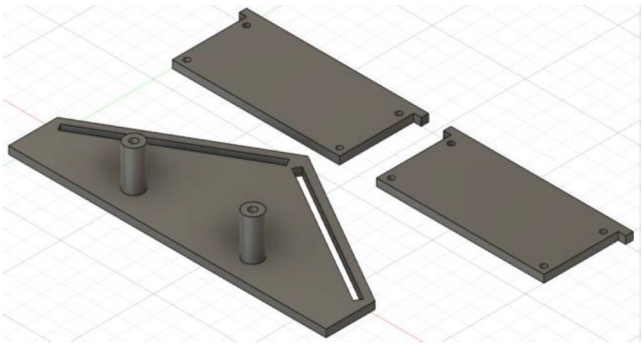


Рисунок 3.1 – Спроектована модель тримача для сенсорів

Друк: Тримачі були надруковані на 3D-принтері, що дозволяє легко їх замінювати або модифікувати при необхідності. Вони забезпечують стабільне кріплення сенсорів та мінімізацію вібрацій під час руху.

Корпус для акумуляторів:

Проектування: Корпуси для акумуляторів були спроектовані для захисту акумуляторів від механічних пошкоджень та забезпечення легкості їх підзарядки.

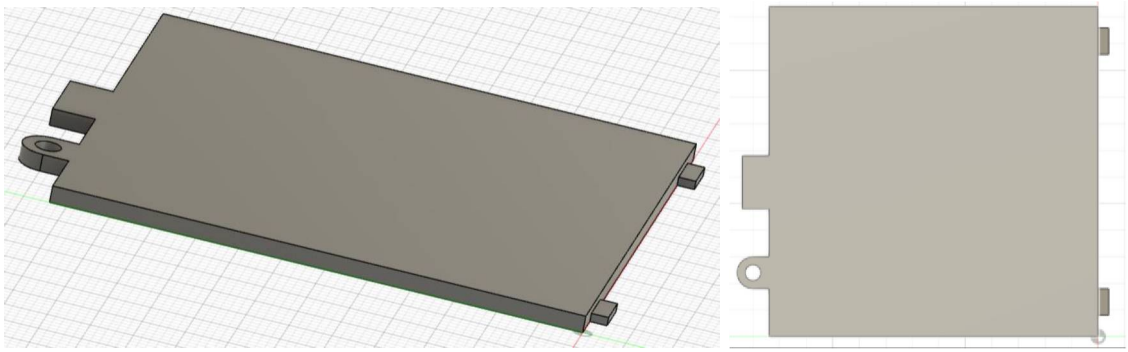


Рисунок 3.2 – Спроектована модель корпусу для акумуляторів

Друк: Корпуси були надруковані з використанням пластику, що забезпечує їх довговічність та надійність.

Технологія 3D-друку:

Використання 3D-друку дозволило швидко та ефективно виготовити необхідні компоненти для модифікації іграшкової машинки. Основні переваги цієї технології включають:

Гнучкість у дизайні:

Індивідуальне налаштування: 3D-друк дозволяє створювати деталі, точно відповідні до потреб проекту, з урахуванням всіх особливостей і вимог. Це особливо важливо для компонентів, які потребують точного кріплення та специфічних форм.

Швидка модифікація: У випадку необхідності зміни дизайну або адаптації під нові умови, моделі деталей можуть бути швидко скориговані в програмному забезпеченні і знову надруковані.

Для проектування деталей використовувалася безкоштовна програма Fusion 360 від Autodesk. Fusion 360 забезпечує потужні інструменти для моделювання, що дозволяє створювати складні деталі та адаптувати їх відповідно до потреб проекту.

Швидкість виготовлення:

Миттєва реалізація: Процес виготовлення деталей за допомогою 3D-друку значно скорочує час між проектуванням і отриманням готового виробу. Це дозволяє швидко проводити тестування та вносити необхідні зміни.

Зниження витрат часу на доставку: Виготовлення деталей безпосередньо на місці зменшує залежність від постачальників і зменшує час на логістику.

Економічність:

Низькі витрати на матеріали: Використання пластику для 3D-друку є економічно вигідним, оскільки матеріали мають відносно низьку вартість у порівнянні з традиційними методами виробництва.

Зменшення витрат на виробництво: Виготовлення деталей за допомогою 3D-друку не потребує великих вкладень у виробниче обладнання, що значно знижує загальні витрати на проект.

Вибір основи для створення фізичної моделі транспортного засобу з використанням іграшкової машинки з коліщатками, редукторів та технології 3D-друку виявився вдалою стратегією. Це забезпечило необхідну механічну стабільність та гнучкість у модифікації, що дозволило створити ефективну та надійну модель для подальших експериментів та розробки програмного забезпечення. Завдяки такому підходу стало можливим швидко адаптувати та

вдосконалювати систему, що є критичним для успішного впровадження та тестування системи керування транспортним засобом.

Використання Fusion 360 від Autodesk додатково сприяло успіху проекту, забезпечивши зручний та потужний інструмент для проектування необхідних компонентів, що значно скоротило час та зусилля, необхідні для створення та модифікації деталей.

3.2. Створення фізичної моделі та програмування алгоритмів аналізу, та прийняття рішень

Цей розділ охоплює процес створення фізичної моделі транспортного засобу та розробку алгоритмів програмного забезпечення для його автоматичного керування. Для кращої структури і розуміння, даний пункт розділено на підпункти, де кожен підпункт присвячений конкретному аспекту моделі або алгоритму. Кожен елемент моделі та кожен алгоритм програмування розглядаються окремо, що дозволяє детально описати процес їх інтеграції в загальну систему.

3.2.1. Акумуляторний блок

Розробка фізичної моделі починається з вибору і інтеграції акумуляторного блоку. Акумуляторний блок є основним джерелом живлення для всіх компонентів моделі, включаючи двигуни, датчики та мікроконтролер.

Для створення ефективного та надійного джерела живлення фізичної моделі транспортного засобу було прийнято рішення використовувати акумуляторні батареї. Після ретельного аналізу різних типів акумуляторів, таких як нікель-кадмієві (NiCd), нікель-металгідридні (NiMH) та літій-іонні (Li-ion), було обрано останній тип. Літій-іонні акумулятори були вибрані через їх високу енергетичну щільність, тривалий термін служби та низький рівень саморозряду, що робить їх ідеальними для даного проекту.

Для забезпечення стабільної роботи моделі було вибрано п’ять акумуляторів ємністю 650 міліампер-годин (мА·год) кожний та номінальною напругою 3,7 Вольт. Вибір саме такої ємності був обумовлений необхідністю забезпечення достатньої потужності для всіх компонентів моделі, включаючи двигуни, датчики та мікроконтролер. Літій-іонні акумулятори відомі своєю високою продуктивністю та надійністю, що гарантує стабільне живлення протягом тривалого часу.

Для збільшення загальної ємності акумуляторного блоку та, відповідно, збільшення часу працездатності моделі, акумулятори були з’єднані паралельно. Паралельне з’єднання акумуляторів дозволяє підсумовувати їх ємності, залишаючи напругу незмінною. Це означає, що п’ять акумуляторів ємністю 650 мА·год кожний, з’єднані паралельно, утворюють акумуляторний блок із загальною ємністю 3250 мА·год при напрузі 3,7 Вольт. Таке рішення дозволяє значно збільшити час роботи моделі без необхідності частого підзаряджання акумуляторів.

Інтеграція акумуляторного блоку у фізичну модель також включає розробку відповідної системи кріплення, яка забезпечує надійне фіксування акумуляторів у корпусі моделі. Для цього використовуються спеціально розроблені кріплення, надруковані на 3D-принтері, які точно відповідають розмірам акумуляторів та забезпечують їхню стабільність під час руху моделі. Крім того, конструкція кріплень враховує необхідність зручного доступу до акумуляторів для їх заміни або обслуговування.

З метою оптимізації процесу зарядки та забезпечення безпеки використання акумуляторів, у фізичну модель також було включено плату заряду з відповідними захистами перезаряду. Ця плата забезпечує контроль та захист від перезаряду, що є критичним для довготривалої та безпечної роботи акумуляторів. Також, на цій платі присутній індикатор заряду, який надає візуальну інформацію про стан заряду акумуляторів. Цей індикатор відображає, чи заряджаються акумулятори, чи вже зарядились, що дозволяє

зручно відслідковувати поточний статус заряду та вчасно вживати заходів, якщо потрібно. Таке комплексне підхід до управління акумуляторним блоком дозволяє підтримувати його ефективну роботу та максимально продовжувати тривалість роботи моделі без перерви.

Таким чином, вибір та інтеграція акумуляторного блоку є критичним етапом у створенні фізичної моделі, оскільки від цього залежить стабільність та тривалість роботи всієї системи. Використання літій-іонних акумуляторів, їх паралельне з’єднання та розробка спеціалізованих кріплень забезпечують надійне та ефективне живлення моделі, що є ключовим для успішної реалізації проекту.

Для створеного акумуляторного блоку було розроблено схематику, яка відображає підключення п'яти акумуляторів ємністю 650 міліампер-годин (мА·год) кожен паралельно. На схемі чітко видно, як кожен акумулятор з'єднаний з іншими паралельно, що дозволяє підсумувати їхні ємності для отримання загальної потужності. Крім того, на схемі показано підключення плати заряду, що забезпечує контроль та захист акумуляторів. Ця схематика стала основою для подальшого виготовлення фізичного акумуляторного блоку та його інтеграції у модель.

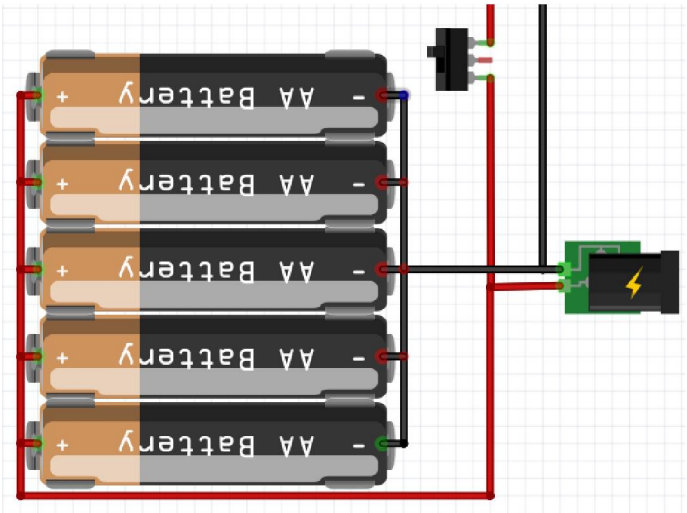


Рисунок 3.3 – Схематика блоку акумуляторів

Забезпечення належного кріплення та безпеки акумуляторів у моделі передбачає розробку спеціалізованих кріплень, які надійно фіксують акумуляторний блок у корпусі моделі. Ці кріплення ретельно розробляються таким чином, щоб забезпечити стабільність під час руху транспортного засобу та забезпечити зручний доступ для заміни акумуляторів або обслуговування. Для цього використовуються спеціально розроблені деталі, які виготовляються за допомогою 3D-принтера з врахуванням розмірів та формату акумуляторів. Крім того, враховується підключення плати заряду з відповідними захистами перезаряду та індикатором заряду, що дозволяє контролювати та забезпечувати безпеку під час заряджання та використання акумуляторів у моделі.

3.2.2. Двигуни

Після інтеграції акумуляторного блоку в фізичну модель наступним важливим кроком є додавання двигунів, які забезпечують рух транспортного засобу. У даному проекті було обрано двигуни постійного струму з напругою 5 Вольт для цієї ролі. Ці двигуни володіють достатньою потужністю для забезпечення руху моделі та є ефективними у використанні.

Для керування цими двигунами та їхньої координації з рухом транспортного засобу був обраний драйвер моторів L9110S. Цей драйвер забезпечує стабільну та ефективну роботу двигунів, а також дозволяє реалізувати необхідні функції управління.

Однак просте підключення двигунів не є достатнім для забезпечення повного функціонування транспортного засобу. Тому через простий редуктор з шестерень було вирішено приєднати двигуни до коліщаток. Ця конструкція дозволяє передавати обертальний рух від двигунів до коліс, що забезпечує рух та керованість транспортного засобу.

Поворот передніх коліс реалізовано через використання механізму, який передає обертальний рух від двигунів до передніх коліс через редуктор. Це дозволяє моделі маневрувати на дорозі.

Для двигунів було розроблено схематику, яка включає в себе відображення підключення двигунів постійного струму до драйвера моторів L9110S. На схемі також показано підключення живлення та керування кожного двигуна. Основною метою є візуалізація правильного підключення та взаємодії всіх компонентів для забезпечення ефективної роботи системи керування моделлю.

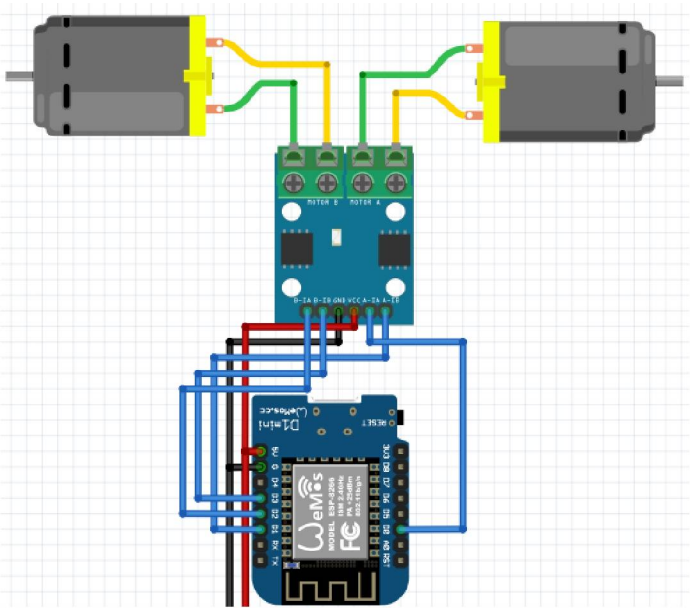


Рисунок 3.4 – Схематика блоку двигунів

Для забезпечення належного функціонування двигунів у фізичній моделі транспортного засобу вони були підключені до контролера за допомогою драйвера моторів L9110S. Цей драйвер забезпечував потрібний рівень потужності та керував напрямком обертання двигунів. Підключення було здійснене за допомогою відповідних кабелів, що забезпечували надійний контакт та передачу сигналів. Це дозволило налаштувати та керувати рухом транспортного засобу з використанням програмного забезпечення, що

взаємодіяло з контролером, забезпечуючи оптимальну працездатність та ефективність руху.

3.2.3. Датчики

Після успішного підключення двигунів до контролера та їх належного функціонування, наступним важливим кроком у створенні фізичної моделі транспортного засобу є додавання датчиків розпізнавання перешкод. Для цієї цілі було обрано ультразвукові датчики відстані HC-SR04, які відомі своєю надійністю та високою точністю вимірювань. Робочий діапазон напруги цих датчиків становить від 3,8 до 5,5 вольт, що робить їх сумісними з широким спектром контролерів, та цілком відповідає робочій напрузі акумуляторів.

Характеристики датчиків включають частоту 40 кілогерц, максимальну робочу відстань до 400 сантиметрів, мінімальну робочу відстань від 0 сантиметрів, роздільну здатність 3 міліметри, ширину імпульсів 10 мікросекунд та кут огляду 15 градусів. Ці параметри забезпечують точне визначення відстані до перешкод та надійну роботу датчиків у різних умовах.

Зібрані дані від датчиків подаються на вхід алгоритмів, які аналізують їх та видають відповідні команди для прийняття рішень. У реалізованому алгоритмі передбачено логіку, яка реагує на наближення моделі до перешкод. Наприклад, якщо відстань до перешкоди становить 50, або менше, сантиметрів, приймається рішення про поворот в протилежний бік від перешкоди. Крім того, якщо сигнал про наближення спостерігається одночасно на обох датчиках, модель автоматично зупиняється, що дозволяє уникнути зіткнень та забезпечує безпеку руху.

Для ультразвукових датчиків наближення було розроблено детальну схематику, яка включає в себе кілька ключових елементів. Схематика відображає розташування датчиків HC-SR04 на моделі.

Крім того, схематика показує, як датчики підключені до мікроконтролера. Кожен датчик HC-SR04 має чотири контакти: VCC, GND, Trigger та Echo. Контакти VCC підключені до джерела живлення (робоча напруга 3.8 – 5.5В), а GND – до заземлення. Контакти Trigger і Echo з'єднані з відповідними піном мікроконтролера для передачі та прийому ультразвукових сигналів.

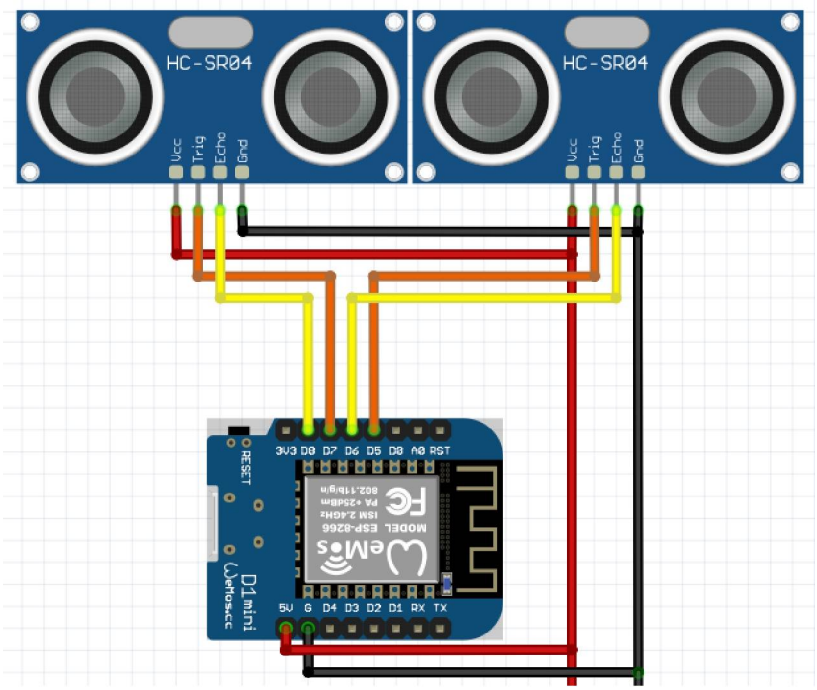


Рисунок 3.5 – Схематика блоку сенсорів

Підключення датчиків до мікроконтролера та налаштування їх для коректної роботи включає кілька важливих кроків. Спочатку ультразвукові датчики HC-SR04 розміщуються на моделі транспортного засобу в оптимальних позиціях для максимального покриття передньої області.

Далі проводиться підключення датчиків до мікроконтролера. Контакт VCC кожного датчика підключається до джерела живлення, а контакт GND - до заземлення. Контакти Trigger та Echo підключаються до відповідних пінів мікроконтролера для забезпечення передачі та прийому ультразвукових сигналів.

Після фізичного підключення мікроконтролер налаштовується на відправку сигналу через контакт Trigger та вимірювання часу повернення

відбитого сигналу через контакт Echo. Отримані дані конвертуються у відстань до перешкоди.

Ці кроки забезпечують точну та надійну роботу датчиків, що є критично важливим для ефективного керування транспортним засобом та уникнення зіткнень з перешкодами.

3.2.4. Програмування алгоритмів керування

Цей підпункт охоплює розробку програмного забезпечення для керування моделлю. Створення основних алгоритмів для базового керування включає програмування функцій для руху вперед, назад та поворотів. У цьому підпункті детально розглядаються ключові функції, які забезпечують ефективне керування транспортним засобом.

Розробка базових алгоритмів керування. Основні алгоритми керування транспортним засобом включають наступні функції:

Forward (Рух вперед): Ця функція відповідає за керування рухом моделі вперед. Вона активує двигуни, щоб транспортний засіб рухався в заданому напрямку. Функція реалізована шляхом надсилання відповідної команди на мікроконтролер, який керує двигунами. Ця команда активує двигуни таким чином, щоб модель рухалася прямо вперед, забезпечуючи стабільний та рівномірний рух.

Мета цієї функції - забезпечити основний рух моделі вперед, що є базовою необхідністю для будь-якого транспортного засобу.

Backward (Рух назад): Функція для руху назад. Активує двигуни в зворотному напрямку для забезпечення руху моделі назад. Ця команда особливо корисна при необхідності уникнути перешкод або для маневрування в обмеженому просторі. Надсилання команди на мікроконтролер дозволяє

двигунам змінити напрямок обертання, забезпечуючи зворотний рух транспортного засобу.

Рух назад важливий для додаткової маневровості та здатності адаптуватися до змінних умов навколишнього середовища.

Left (Поворот вліво): Ця функція забезпечує поворот транспортного засобу вліво, активуючи відповідний двигун для зміни напрямку руху. Поворот вліво реалізується шляхом надсилання команди, яка змушує один із двигунів зупинитися або сповільнитися, тоді як інший двигун продовжує обертання. Це створює асиметрію в обертанні коліс, що призводить до повороту транспортного засобу.

Поворот вліво важливий для забезпечення маневреності моделі, що дозволяє їй обходити перешкоди та змінювати напрямок руху.

Right (Поворот вправо): Функція, що відповідає за поворот вправо. Вона активує двигун для зміни напрямку руху транспортного засобу вправо. Як і у випадку з поворотом вліво, поворот вправо досягається за рахунок різниці в швидкості обертання коліс. Команда для повороту вправо змушує один двигун зупинитися або сповільнитися, що призводить до повороту транспортного засобу вправо.

Ця функція забезпечує повний контроль над напрямком руху транспортного засобу, що є критично важливим для успішного маневрування.

Stop (Зупинка руху): Ця функція використовується для зупинки руху транспортного засобу, вимикаючи всі двигуни. Команда зупинки надсилається на мікроконтролер, який припиняє подачу електроенергії на двигуни, забезпечуючи миттєву зупинку моделі. Це важливо для безпеки та точного контролю транспортного засобу.

Функція зупинки є важливою для запобігання зіткнень та швидкої реакції на непередбачені ситуації.

Auto (Автоматичний режим): Ця функція відповідає за вмикання та вимикання автоматичного керування моделлю на основі отриманих даних з датчиків. Алгоритм автоматичного управління реалізований на основі аналізу відстаней до об'єктів, виміряних ультразвуковими датчиками. Наприклад, якщо датчик виявляє перешкоду на шляху моделі, то функція активує поворот для уникнення зіткнення. Автоматичний режим дозволяє моделі самостійно реагувати на зміни у навколишньому середовищі та уникати перешкод без втручання користувача.

Структура програмного забезпечення:

Програмне забезпечення побудовано на базі бібліотеки Kivy для створення графічного інтерфейсу користувача (GUI), що дозволяє легко керувати моделлю. Основні компоненти програмного забезпечення включають:

Інтерфейс користувача: Інтерфейс складається з кнопок для керування рухом транспортного засобу (вперед, назад, повороти та автоматичний режим) та дисплеїв для відображення даних з датчиків відстані. Інтерфейс користувача спроектований таким чином, щоб бути інтуїтивно зрозумілим і легко керованим, забезпечуючи користувачу можливість швидко реагувати на зміну умов руху моделі.

Клас MyApp: Головний клас програми, який відповідає за створення інтерфейсу користувача та обробку подій. Він містить основну логіку для взаємодії з користувачем і обробки команд керування.

Цей клас відповідає за ініціалізацію всіх компонентів програми, включаючи створення макетів, кнопок керування та обробку подій, таких як натискання кнопок та отримання даних з датчиків.

Функція send_command: Функція для надсилання команд на мікроконтролер через HTTP-запити. Вона приймає команду як параметр і

відправляє її на відповідний URL. Це забезпечує безперервну взаємодію між програмою та апаратним забезпеченням моделі.

Функція send_command є ключовою для реалізації будь-яких дій транспортного засобу, оскільки вона перетворює команди користувача на конкретні дії мікроконтролера.

Функція update_distances: Ця функція періодично оновлює відстані, отримані з датчиків, і відображає їх у графічному інтерфейсі користувача. Вона відповідає за збір даних від датчиків і їх візуалізацію в реальному часі.

Функція update_distances забезпечує постійний моніторинг навколишнього середовища, що дозволяє транспортному засобу швидко реагувати на перешкоди та змінювати маршрут у разі необхідності.

Програмне забезпечення забезпечує інтеграцію з мікроконтролером, використовуючи HTTP-запити для обміну командами та даними. Це дозволяє легко керувати моделлю та відстежувати її стан в режимі реального часу. Завдяки добре структурованим функціям керування, забезпечується надійне та ефективне управління транспортним засобом.

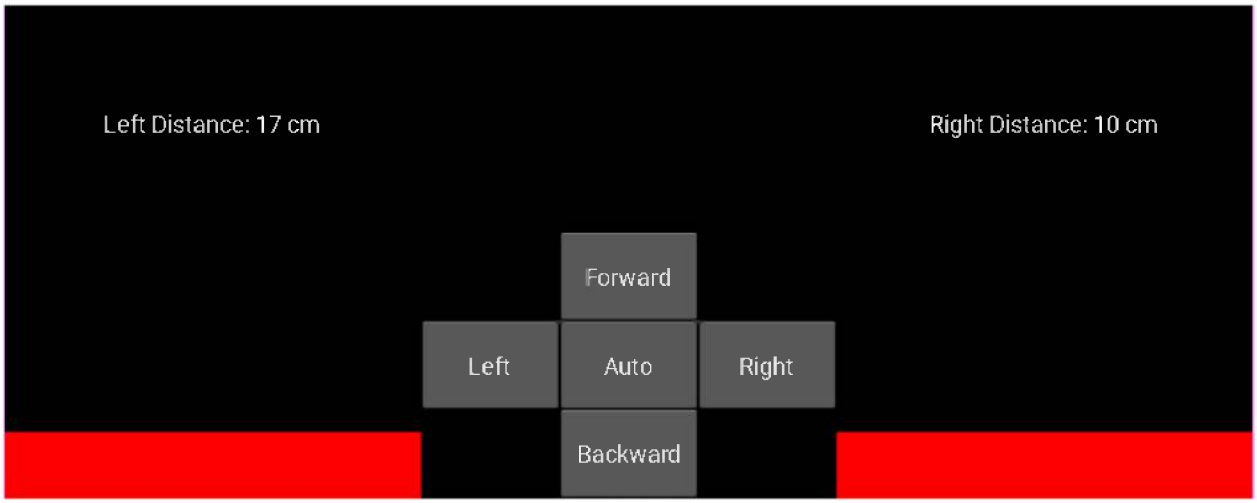


Рисунок 3.6 – Інтерфейс програми керування

3.2.5. Алгоритми розпізнавання перешкод

У цьому підпункті детально описуються алгоритми, які дозволяють моделі розпізнавати та реагувати на перешкоди на своєму шляху. Основою роботи цих алгоритмів є використання даних з ультразвукових датчиків HC-SR04, які встановлені на моделі. Ці датчики дозволяють вимірювати відстань до об'єктів перед моделлю, створюючи картину навколишнього середовища.

Підключення датчиків: Датчики підключені до мікроконтролера, використовуючи тригерні та ехо піни. Для кожного з датчиків створюється об'єкт класу «NewPing», який забезпечує ефективну роботу з ними. Датчики розташовані на передній частині моделі, що дозволяє забезпечити широкий кут огляду. Такий спосіб розташування дозволяє охопити більшу область перед моделлю, що підвищує точність виявлення перешкод.

Ультразвукові датчики працюють за принципом ехолокації: вони випромінюють звукові хвилі, які відбиваються від об'єктів і повертаються до датчика. Мікроконтролер вимірює час, за який звукова хвиля повертається, і на основі цього розраховує відстань до об'єкта.

Збір даних: Дані про відстань до об'єктів збираються за допомогою викликів методів об'єктів «NewPing». Кожен з датчиків періодично випромінює ультразвуковий сигнал і вимірює час, необхідний для повернення відбитого сигналу. На основі цього часу обчислюється відстань до об'єкта.

Для кожного вимірювання створюється окрема змінна, яка зберігає отриману відстань. Ці дані оновлюються в реальному часі, що дозволяє моделі постійно адаптувати свій рух відповідно до змін у навколишньому середовищі.

Мікроконтролер обробляє сигнали з датчиків і визначає, на якій відстані знаходяться об'єкти зліва та справа від моделі. Це дозволяє створити просторову карту перед моделлю, яка оновлюється з кожним циклом вимірювань. На основі цієї карти модель приймає рішення про подальший рух.

Інтерпретація даних: Алгоритми розпізнавання перешкод використовують отримані відстані для визначення наявності перешкод на шляху моделі. Якщо відстань до об'єкта менша за заданий поріг (50 см), це розцінюється як перешкода. Ці дані використовуються для прийняття рішень про подальші дії моделі.

Модель постійно збирає дані про відстані до об'єктів з обох датчиків. Якщо обидва датчики виявляють перешкоди на відстані меншій за встановлений поріг, модель приймає рішення зупинитися, щоб уникнути зіткнення. Якщо перешкода виявляється лише з одного боку, модель здійснює маневр для обходу перешкоди, змінюючи напрямок руху.

Таким чином, алгоритми розпізнавання перешкод дозволяють моделі адаптуватися до змін у навколишньому середовищі в реальному часі, забезпечуючи безпечний і ефективний рух. Інтеграція даних з ультразвукових датчиків та їх аналіз є основою для створення інтелектуальної системи керування моделлю, яка може самостійно орієнтуватися в просторі та уникати перешкод.

Інтеграція в основний цикл: В основному циклі програми («loop») перевіряється, чи активований автопілот. Якщо так, виконується збір даних з датчиків та приймаються відповідні рішення щодо руху. Алгоритм працює в реальному часі, постійно оцінюючи відстані до перешкод і коригуючи рух моделі.

Коли активовано автопілот, модель постійно вимірює відстані до об'єктів перед собою. У випадку виявлення перешкод алгоритм вирішує, чи зупинити модель, чи змінити напрямок її руху для обходу перешкод. Це забезпечує безпечний і плавний рух моделі, навіть у складних умовах.

Функції для керування моторами: Для забезпечення руху моделі використовуються функції керування моторами, які виконують різні команди:

рух вперед, назад, повороти та зупинки. Ці функції викликаються в залежності від результатів аналізу даних з датчиків.

Основні функції керування включають:

- «motorA_forward()»: функція для руху мотору А вперед – поворот передніх коліс ліворуч.
- «motorA_backward()»: функція для руху мотору А назад – поворот передніх коліс прворуч.
- «motorA_stop()»: функція для зупинки мотору А – зупинка дії поворотного механізму.
- «motorB_forward()»: функція для руху мотору В вперед.
- «motorB_backward()»: функція для руху мотору В назад.
- «motorB_stop()»: функція для зупинки мотору В.

Ці функції забезпечують контроль над рухом моделі, дозволяючи їй адаптуватися до умов навколишнього середовища і уникати перешкод на своєму шляху. Інтеграція алгоритмів розпізнавання перешкод з функціями керування моторами є критично важливою для забезпечення безпеки та ефективності руху моделі.

В цілому, розділ «Створення фізичної моделі та програмування алгоритмів аналізу, та прийняття рішень» охоплює всі аспекти створення інтегрованої системи, яка включає апаратну та програмну частини. Це забезпечує повне розуміння процесу розробки та дозволяє створити ефективну модель для подальших експериментів і впровадження.

3.3. Компоненти системи

Для реалізації системи використовується 13 модулів:

Таблиця 3.1

Компоненти системи:

Товар	Кількість	Сума(грн)
Мікроконтролер ESP8266 V3	1 шт.	100
Драйвер моторів L9110S	1 шт.	30
Модуль заряду	1 шт.	10
Акумулятори	5 шт.	75
Ультразвуковий датчик відстані HC-SR04	2 шт.	60
Двигун постійного струму	2 шт.	50
Дроти з'єднання	1 уп.	25
Витрати на друк додаткових деталей	4 шт.	45
Основа коліщаток для кріплення	1 шт.	100
		Всього: 495

Висновки за розділом 3

Третій розділ роботи присвячено реалізації системи розпізнавання перешкод для автопілоту, охоплюючи процеси моделювання, створення фізичної моделі та програмування алгоритмів керування і аналізу.

Моделювання основи для системи. У цьому підрозділі було розглянуто процес моделювання основи для системи розпізнавання перешкод. Визначено основні параметри та конструкційні особливості, які забезпечують стабільну роботу системи. Було створено віртуальну модель, що включає всі необхідні компоненти, та проведено симуляції для перевірки її функціональності.

Створення фізичної моделі та програмування алгоритмів аналізу, та прийняття рішень. Підрозділ деталізує етапи створення фізичної моделі, включаючи інтеграцію акумуляторного блоку, двигунів, датчиків та інших компонентів. Було проведено ретельний відбір та тестування кожного компонента для забезпечення їхньої сумісності та ефективної роботи в складі системи.

Акумуляторний блок: Розглянуто вибір і підключення акумуляторного блоку, який забезпечує енергією всі компоненти системи. Визначено оптимальні характеристики акумулятора для досягнення максимальної продуктивності та тривалості роботи.

Двигуни: Проведено аналіз типів двигунів, які найбільше підходять для реалізації системи. Обрано двигуни, що забезпечують необхідну швидкість та маневреність моделі.

Датчики: Обрано та інтегровано датчики, що дозволяють системі розпізнавати перешкоди. Було протестовано різні типи сенсорів, включаючи ультразвукові, інфрачервоні та лазерні датчики, для визначення найефективніших з них.

Програмування алгоритмів керування: Розроблено та реалізовано алгоритми керування, що забезпечують стабільне функціонування системи. Алгоритми були написані на базі Python та Arduino IDE, забезпечуючи необхідний рівень гнучкості та адаптивності.

Алгоритми розпізнавання перешкод: Впроваджено алгоритми розпізнавання перешкод, що дозволяють системі швидко та точно виявляти об'єкти на своєму шляху. Алгоритми базуються на обробці даних з сенсорів та використанні методів машинного навчання для покращення точності розпізнавання.

Компоненти системи. У підрозділі розглянуто всі основні компоненти системи, включаючи апаратну та програмну частини. Визначено їхні функції та взаємодію між собою. Проведено тестування кожного компоненту окремо та у складі системи для забезпечення їхньої надійної роботи.

ВИСНОВКИ

У даній роботі була розроблена система розпізнавання перешкод для автопілоту, яка використовує поєднання мікроконтролера ESP8266 та програмного забезпечення, написане мовою Python, для дистанційного керування.

У першому розділі ретельно проаналізовано концепцію «Кіберфізична система розпізнавання перешкод для автопілоту». Досліджено ключові аспекти та сутність цієї концепції, що дозволило отримати глибше розуміння принципів функціонування системи.

Другий розділ включав у себе процес проектування системи, вибір компонентів та програмного забезпечення для реалізації моделі автопілоту. Крім того, було розглянуто варіанти 3D-моделювання для створення додаткових компонентів та необхідних кріплень.

У третьому розділі докладно описано процес реалізації системи. Це включало моделювання основи та створення фізичної моделі з використанням 3D-моделювання, програмування алгоритмів керування для мікроконтролера ESP8266, а також розробку програмного забезпечення на мові Python для дистанційного керування.

Робота підкреслила успішне поєднання інженерних знань та навичок програмування для розробки та реалізації інтелектуальної системи автопілоту. Потенційні перспективи застосування подібних систем у розвитку автономних транспортних засобів можуть бути значними, враховуючи їх ефективність та можливості в управлінні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Визначення «Кіберфізична система» [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D1%96%D0%B1%D0%B5%D1%80%D1%84%D1%96%D0%B7%D0%B8%D1%87%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0
2. Статистика розробників автопілоту для транспортних засобів [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://www.imena.ua/blog/18-companies-to-get-driverless-cars/
3. Система автопілот [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: http://www.1gai.ru/publ/516653-avtopilot-tesla-samaya-prodvinutaya-iz-sovremennyh-sistemna-rynke.html
4. Лідари в автомобілях [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://vc.ru/transport/61028-lidary-v-bespilotnyh-avtomobilyah
5. Ультразвукові датчики [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://electrosam.ru/glavnaja/slabotochnye-seti/oborudovanie/ultrazvukovye-pribory/
6. Система автоматичного управління автомобілем [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: http://ua.nauchebe.net/2015/04/elektronni-sistemi-keruvannyaavtomobilyami-napivprovidnikova-silova-elektronika/
7. Лисенко В. П., Шворов С. А., Комарчук Д. С., Чирченко Д. В. Метод розпізнавання перешкод на шляху руху роботизованої збиральної техніки / Техніка та енергетика №256, 2016.
8. S. Semenov, O. Lipchanska, M. Lipchanskyi. Adapted neural network of information support subsystem. / Наука і техніка Повітряних Сил Збройних Сил України №1 (34), 2019.

ДОДАТОКИ

Додаток А

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

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 – Комп'ютерна інженерія.**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної
та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.
«21» грудня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

СЕРГЄЄВ ВЯЧЕСЛАВ ОЛЕКСАНДРОВИЧ
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«Система розпізнавання перешкод для автопілота»**

керівник роботи Бикова Тетяна Володимирівна, к.т.н., доцент кафедри ТПС (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» травня 2024року № 4101-5/909

2. Строк подання студентом роботи 31 травня 2024 року

3. Перелік питань, які потрібно розробити
1) Визначення технічних параметрів системи розпізнавання перешкод.
2) Розробка алгоритму обробки даних з сенсорів для виявлення перешкод.
3) Аналіз ефективності різних методів розпізнавання перешкод.
4) Визначення метрик оцінки точності та швидкодії системи.
5) Проектування інтерфейсу взаємодії з автопілотом для управління системою розпізнавання перешкод.
6) Аналіз можливостей використання системи в реальних умовах та визначення потенційних областей застосування.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз научної літератури	30.10.2023 – 14.11.2023
2	Визначення технічних параметрів системи розпізнавання перешкод	14.11.2024 – 31.12.2024
3	Розробка алгоритму обробки даних з сенсорів для виявлення перешкод системи	02.01.2024 – 31.01.2024
4	Проектування інтерфейсу взаємодії з автопілотом для управління системою розпізнавання перешкод	01.02.2024 – 29.02.2024
5	Визначення метрик оцінки точності та швидкодії	01.03.2024 – 14.03.2024
6	Аналіз можливостей використання системи в реальних умовах та визначення потенційних областей застосування.	15.03.2024 – 31.03.2024
7	Підготовка технічного звіту	01.04.2024 – 14.05.2024
8	Розробка пояснювальної записки.	15.05.2024 – 31.05.2024
9	Представлення кваліфікаційної роботи керівнику та рецензенту.	31.05.2024
10	Оформлення пояснювальної записки та підготовка презентації.	31.05.2024

5. Дата видачі завдання 21.12.2023

Студент В. О. Сергєєв _____
ініціали, прізвище підпис

Керівник роботи Т. В. Бикова _____
ініціали, прізвище підпис

Додаток Б

Додаток Б

Затверджую

_____

«_____»_____2024 р.

Технічне завдання

на розробку системи

«Система розпізнавання перешкод для автопілоту».

1.	Введення	1.1 Назва роботи – Система розпізнавання перешкод для автопілоту. 1.2. Галузь застосування: Інформаційні технології
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп'ютерна інженерія 2.2. Завдання на кваліфікаційну роботу бакалавра №_____від «__»_____2024 року (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: Підвищення ефективності систем розпізнавання перешкод системами з автопілотом 3.2. Призначення розробки: Створення технологічного рішення, яке допоможе підвищити рівень безпеки автомобільних перевезень шляхом автоматичного виявлення та уникнення перешкод на шляху автопілотованого транспортного засобу.

		3.3. Вхідні дані: вхідні дані програмного забезпечення включають: вимірювання, отримані від датчиків наближення, інформація, що надходить через WiFi мережу з смартфона, наприклад, команди керування. 3.4. Вихідні дані розробки: У якості вихідних даних система надає інформацію про поточний стан, та команди керування. Система може виробляти команди щодо руху - наприклад, зміна швидкості, напрямку або режиму руху, які дозволяють моделі транспортного засобу уникнути перешкод.
4.	Технічні вимоги до програмного виробу	4.1. Функціональні вимоги: - Можливість дистанційного керування зі смартфону, через WiFi мережу для отримання команд керування. - Аналіз неперервних даних з датчиків, прийняття рішень на основі аналізу та автоматичне керування на основі отриманих команд для прийнятих рішень щодо уникнення перешкод. 4.2. Нефункціональні вимоги: - Система повинна бути надійною і функціональною у різних умовах експлуатації, зокрема в різних погодних умовах та на різних типах поверхонь. - Система повинна точно визначати розмір та відстань до перешкод для забезпечення ефективного уникнення.

		- Система повинна працювати швидко та ефективно, забезпечуючи миттєву відповідь на зміни у середовищі. - Система повинна ефективно використовувати енергію, забезпечуючи тривалий час роботи без необхідності частого заряджання або заміни батарей. - Система повинна мати захист від несанкціонованого доступу та вмістити заходи безпеки для запобігання втраті чи викраденню даних. - Система повинна бути здатною масштабуватися для використання в різних масштабах, включаючи різні розміри транспортних засобів та умови руху. 4.3. Вимоги до інтеграції: - Система повинна бути інтегрована з мікроконтролером ESP8266, забезпечуючи можливість зчитування даних з датчиків наближення та керування моделлю через цей пристрій. - Система повинна бути інтегрована з мобільним додатком на смартфоні, забезпечуючи взаємодію через WiFi мережу для передачі команд керування та отримання звітів про стан роботи. 4.4. Вимоги до безпеки: Автоматизація процесу оновлення ПЗ для усунення вразливостей.

5.	Вимоги до програмної документації	Документацією до виробу «Система розпізнавання перешкод для автопілоту» вважати: 1) Опис основних вимог та функціональності системи (представити у вигляді Додатку Б пояснювальної записки до кваліфікаційної роботи). 2) Програму і методику випробувань розробленої програми (представити як додаток В до пояснювальної записки до кваліфікаційної роботи). 3) Опис розробленої моделі (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи).
6.	Вимоги до техніко-економічних показників	Документацією до виробу «Система розпізнавання перешкод для автопілоту» вважати: 1) Справжнє Технічне завдання на розробку системи управління теплицею (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Опис розробленої системи управління теплицею (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи). 3) Джерела базової інформації.

7.	Стадії і етапи розробки	Дата	Назва етапу
		21.12.2023 – 25.01.2024	Аналіз наукової літератури та існуючих технологій, пов'язаних з системою розпізнавання перешкод для автопілоту.
		19.12.2023 – 02.01.2024	Формування вимог до технічних характеристик.
		02.01.2024 – 02.02.2024	Розробка технічного завдання.
		02.01.2024 – 02.02.2024	Розробка структурної схеми «моделі транспортного засобу на керуванні» та вибір необхідного обладнання.
		03.02.2024 – 30.03.2024	Розробка програмного забезпечення та «моделі транспортного засобу на керуванні»
		03.03.2024 – 30.04.2024	Тестування та валідація роботи системи розпізнавання перешкод для автопілоту
		31.03.2024 – 27.05.2024	Підготовка технічного звіту
		31.03.2024 – 27.05.2024	Розробка пояснювальної записки.
		15.05.2024 – 31.05.2024	Представлення кваліфікаційної роботи керівнику та рецензенту.
		31.05.2024	Оформлення пояснювальної записки та підготовка презентації.

8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки комп'ютерної моделі виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді.

Виконавець

Студент групи КІ-41

СЕРГЄЄВ В.О.

Замовник

д. т. н., проф.

БИКОВА Т. В.

Додаток В

Програма і методика випробувань програмного виробу

«Система розпізнавання перешкод для автопілоту»

1. Об'єкт випробувань

1.1. Назва розробленого прототипу: «Система розпізнавання перешкод для автопілоту».

1.2. Галузь застосування: Інформаційні технології

1.3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональні можливості системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

3.1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3.3. Обсяг випробувань

Приймальні випробування програмного виробу проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавці та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

4.1. Функціональні вимоги:

- Можливість дистанційного керування зі смартфону, через WiFi мережу для отримання команд керування.
- Аналіз неперервних даних з датчиків, прийняття рішень на основі аналізу та автоматичне керування на основі отриманих команд для прийнятих рішень щодо уникнення перешкод.

4.2. Нефункціональні вимоги:

- Система повинна бути надійною і функціональною у різних умовах експлуатації, зокрема в різних погодних умовах та на різних типах поверхонь.
- Система повинна точно визначати розмір перешкод для забезпечення ефективного уникнення.
- Система повинна працювати швидко та ефективно, забезпечуючи миттєву відповідь на зміни у середовищі.
- Система повинна ефективно використовувати енергію, забезпечуючи тривалий час роботи без необхідності частого заряджання.
- Система повинна мати захист від несанкціонованого доступу та вмістити заходи безпеки для запобігання втраті чи викраденню даних.

- Система повинна бути здатною масштабуватися для використання в різних масштабах, включаючи різні розміри транспортних засобів та умови руху.

4.3. Вимоги до інтеграції:

- Система повинна бути інтегрована з мікроконтролером ESP8266, забезпечуючи можливість зчитування даних з датчиків наближення та керування моделлю через цей пристрій.
- Система повинна бути інтегрована з мобільним додатком на смартфоні, забезпечуючи взаємодію через WiFi мережу для передачі команд керування та отримання звітів про стан роботи.

4.4. Вимоги до безпеки:

- Автоматизація процесу оновлення ПЗ для усунення вразливостей.

5. Вимоги до програмної документації

Документацією до виробу «Система розпізнавання перешкод для автопілоту» вважати:

1) Опис основних вимог та функціональності системи (представити у вигляді Додатку Б пояснювальної записки до кваліфікаційної роботи).

2) Програму і методику випробувань розробленої програми (представити як додаток В до пояснювальної записки до кваліфікаційної роботи).

3) Опис розробленої комп'ютерної моделі (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1. Засоби випробувань

Випробування проводяться на технічних засобах, серед яких персональний комп'ютер та(або) смартфон, та побудована модель, яка містить датчики для відповідної працездатності алгоритму системи «Система розпізнавання перешкод для автопілоту».

Випробування проводяться з використанням програмних засобів, серед яких Visual Studio, ArduinoIDE.

6.2. Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

- Ознайомчий (1-й етап);
- Власне випробування програмного виробу та створеного макету (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

Перевірка функціональності:

Виявлення перешкод: система вірно виявляє перешкоди в оточенні за допомогою датчиків наближення; система правильно аналізує дані, отримані від датчиків наближення, та визначає відстань до перешкод; система приймає вірні рішення щодо керування макетом з метою уникнення перешкод.

Взаємодія зі смартфоном: система взаємодіє з мобільним додатком на смартфоні, та(або) персональним комп'ютером, через WiFi мережу для передачі команд керування та отримання звітів про стан роботи.

Автоматичне керування моделлю транспортного засобу: система здатна автоматично керувати макетом на основі отриманих даних та прийнятих рішень щодо уникнення перешкод.

Перевірка інтеграції:

Система може успішно підключатися до мікроконтролера ESP8266 через WiFi мережу. Система може зчитувати дані з датчиків наближення, підключених до мікроконтролера ESP8266. Система може передавати команди керування макетом через мікроконтролер ESP8266.

Система може успішно встановлювати з'єднання з мобільним додатком на смартфоні через WiFi мережу. Система може приймати команди керування від смартфону та відповідати на них адекватно. Система може надсилати звіти про свій стан роботи до мобільного додатку на смартфоні.

Для роботи необхідна Фізична модель та телефона, або комп'ютер, з можливістю під'єднання по Wi-Fi

1. Порядок проведення випробувань:

1.1. Запуск моделі здійснюється за допомогою тумблера на корпусі;

1.2. Після запуску програми необхідно під'єднатись до Wi-Fi мережі «NodeMCU»;

1.3. Після під'єднання – зайти в додаток керування для керування моделюю.

Тест:

1. Перевірка виконання програми для керування;
2. Перевірка алгоритму руху керування;
3. Перевірка роботи постійного аналізу даних з ультразвукових датчиків;
4. Перевірка автоматичного руху.

Автоматичний поворот ліворуч при приближенні до перешкоди:

Рис. В.1 Тест

Аналіз отриманих даних та відображення їх у програмі:

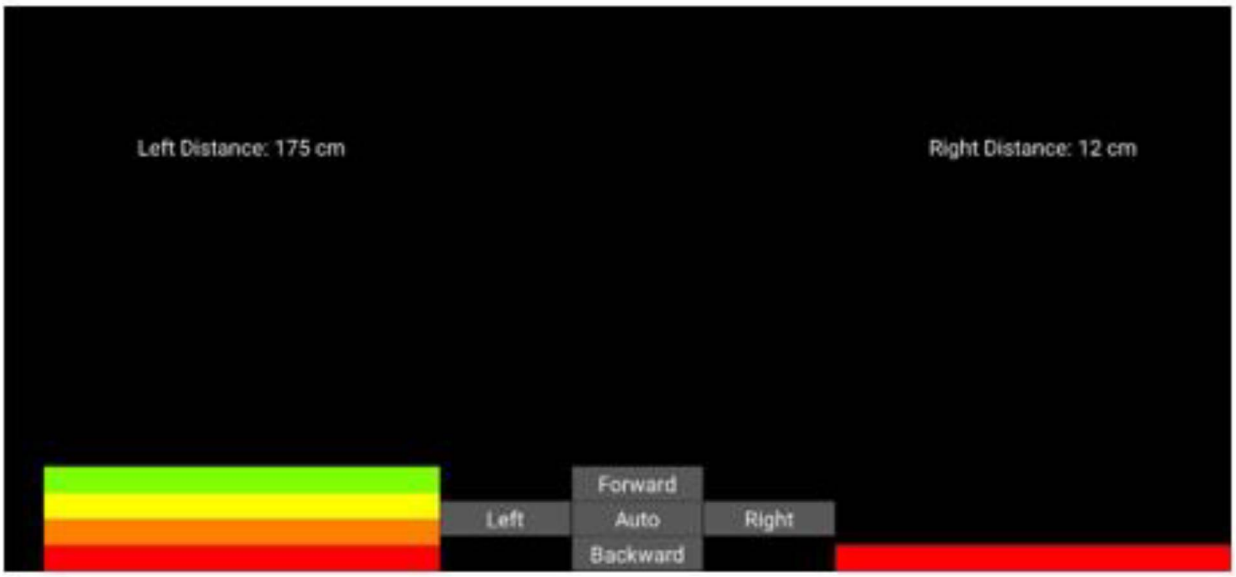


Рис. В.2 Тест

Автоматичний поворот праворуч при приближенні до перешкоди:

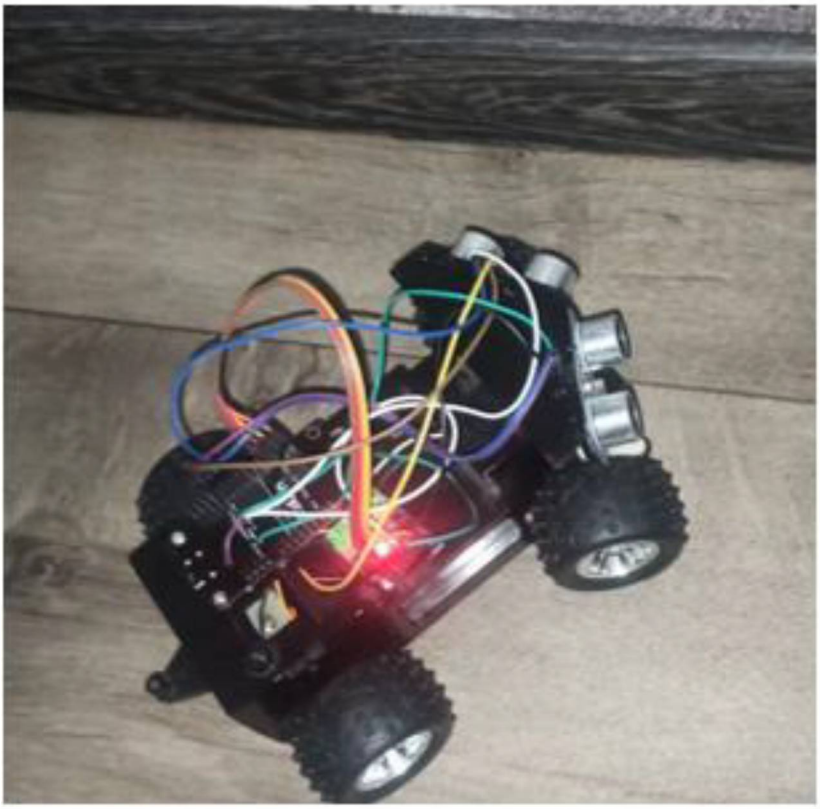


Рис. В.3 Тест

Аналіз отриманих даних та відображення їх у програмі:

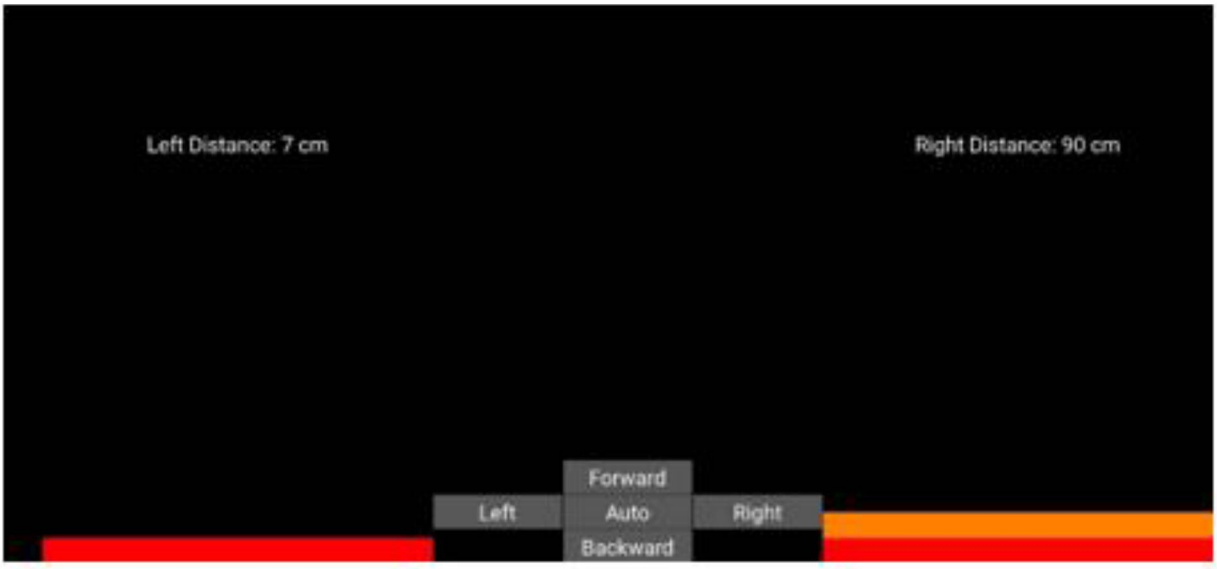


Рис. В.4 Тест

Автоматичний рух вперед при віддаленості від перешкод:

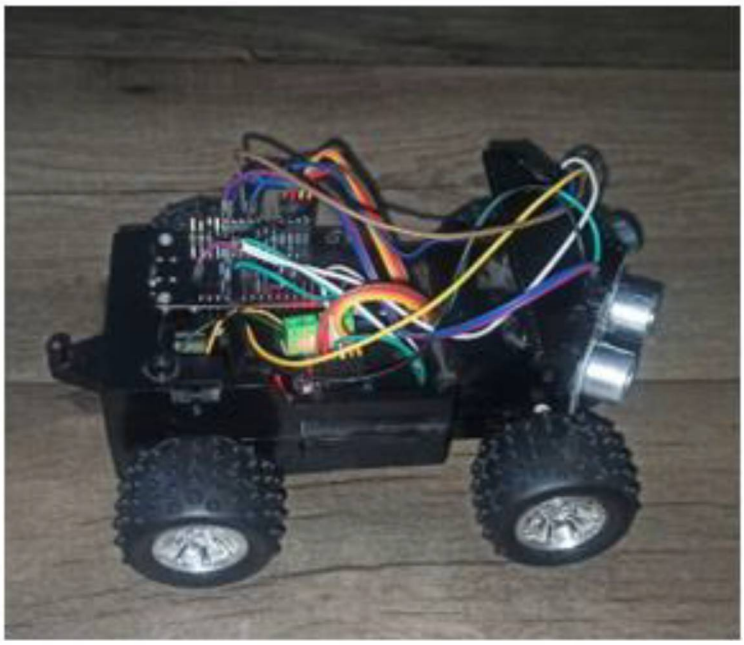


Рис. В.5 Тест

Аналіз отриманих даних та відображення їх у програмі:

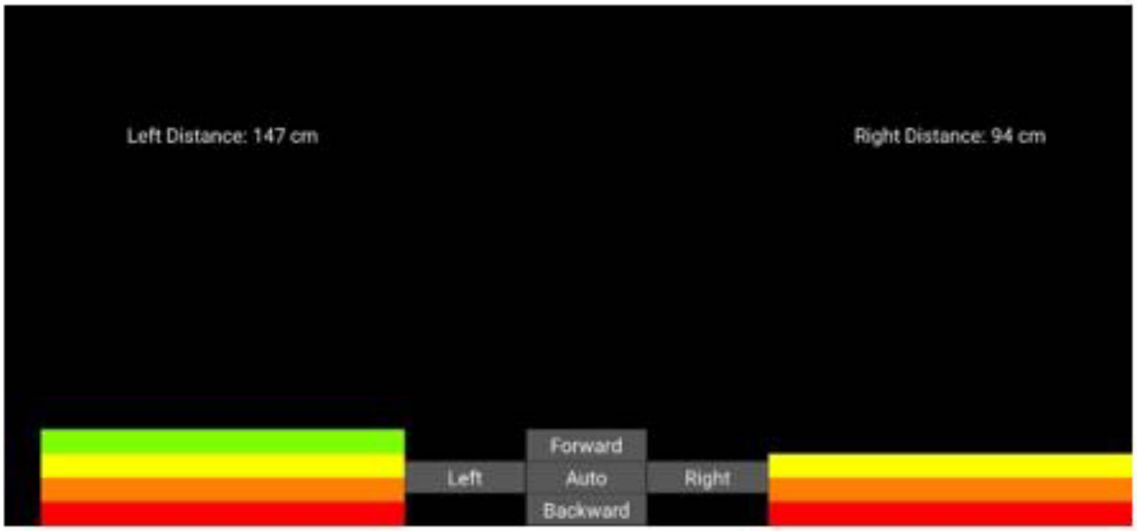


Рис. В.6 Тест

Виконавець

Студент групи КІ-41

СЕРГЄЄВ В.О.

Замовник

д. т. н., проф.

БИКОВА Т. В.

Додаток Г

Лістинг Г.1 – Програма керування моделюю

```
import requests
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
from kivy.clock import Clock
from kivy.uix.label import Label
from kivy.uix.gridlayout import GridLayout
from kivy.uix.widget import Widget
from kivy.graphics import Color, Rectangle
from kivy.core.window import Window

# Define a custom widget to represent a section of the pyramid
class PyramidSection(Widget):
    def __init__(self, **kwargs):
        super(PyramidSection, self).__init__(**kwargs)
        self.bind(size=self.update_color, pos=self.update_color)
        self.color = (0, 1, 0, 1)  # Default color is green

    # Update the color of the section
    def update_color(self, *args):
        with self.canvas.before:
            self.canvas.before.clear()
            Color(*self.color)
            Rectangle(pos=self.pos, size=self.size)

# Define a custom layout to represent the distance pyramid
class DistancePyramid(BoxLayout):
    def __init__(self, **kwargs):
        super(DistancePyramid, self).__init__(**kwargs)
        self.orientation = 'vertical'
        # Create 5 sections for the pyramid
        self.sections = [PyramidSection(size_hint_y=None,
height=50) for _ in range(5)]
        for section in self.sections:
            self.add_widget(section)

    # Update the colors of the pyramid sections based on
distance
    def update_pyramid(self, distance):
        thresholds = [210, 135, 90, 46, 2]
        colors = [
            (0, 1, 0, 1),  # Green
            (0.5, 1, 0, 1),  # Light Green
            (1, 1, 0, 1),  # Yellow
            (1, 0.5, 0, 1),  # Orange
            (1, 0, 0, 1)   # Red
        ]
        for i, section in enumerate(self.sections):
            if distance > thresholds[i]:
```

```
                section.color = colors[i]
            else:
                section.color = (0, 0, 0, 1)  # Off/Black
            section.update_color()

# Define the main application class
class MyApp(App):

    DEFAULT_LOGIN = "NodeMCU"
    DEFAULT_PASSWORD = "11112222"

    def build(self):
        # Create the main layout
        main_layout = BoxLayout(orientation='horizontal')

        # Define the left layout for the left sensor
        left_layout = BoxLayout(orientation='vertical')
        self.left_distance_label = Label(text='Left Distance: 0
cm')
        self.left_distance_pyramid = DistancePyramid()
        left_layout.add_widget(self.left_distance_label)
        left_layout.add_widget(self.left_distance_pyramid)

        # Define the right layout for the right sensor
        right_layout = BoxLayout(orientation='vertical')
        self.right_distance_label = Label(text='Right Distance:
0 cm')
        self.right_distance_pyramid = DistancePyramid()
        right_layout.add_widget(self.right_distance_label)
        right_layout.add_widget(self.right_distance_pyramid)

        # Define the control layout using GridLayout
        control_layout = GridLayout(cols=3, size_hint_y=None,
height=200)

        # Create buttons for control commands
        btn_left = Button(text='Left')
        btn_right = Button(text='Right')
        btn_forward = Button(text='Forward')
        btn_backward = Button(text='Backward')
        btn_autopilot = Button(text='Auto')

        # Bind button events to send corresponding commands
        btn_left.bind(on_press=lambda instance:
self.send_command("Left"), on_release=lambda instance:
self.send_command("WheelStop"))
        btn_right.bind(on_press=lambda instance:
self.send_command("Right"), on_release=lambda instance:
self.send_command("WheelStop"))
        btn_forward.bind(on_press=lambda instance:
self.send_command("Forward"), on_release=lambda instance:
self.send_command("MoveStop"))
```

```
        btn_backward.bind(on_press=lambda instance:
self.send_command("Backward"), on_release=lambda instance:
self.send_command("MoveStop"))
        btn_autopilot.bind(on_press=lambda instance:
self.send_command("Auto"))

        # Add buttons to the control layout
        control_layout.add_widget(Label())  # Empty label for
alignment
        control_layout.add_widget(btn_forward)
        control_layout.add_widget(Label())  # Empty label for
alignment
        control_layout.add_widget(btn_left)
        control_layout.add_widget(btn_autopilot)
        control_layout.add_widget(btn_right)
        control_layout.add_widget(Label())  # Empty label for
alignment
        control_layout.add_widget(btn_backward)
        control_layout.add_widget(Label())  # Empty label for
alignment

        # Add the left layout, control layout, and right layout
to the main layout
        main_layout.add_widget(left_layout)
        main_layout.add_widget(control_layout)
        main_layout.add_widget(right_layout)

        # Schedule the function to update sensor data every 0.5
seconds
        Clock.schedule_interval(self.update_distances, 0.5)

        # Set the window size
        Window.size = (750, 300)  # Width x Height

        return main_layout

    # Function to send a command to the NodeMCU device
    def send_command(self, command):
        url = 'http://192.168.4.1/command'

        # Use hardcoded login and password
        login = self.DEFAULT_LOGIN
        password = self.DEFAULT_PASSWORD

        try:
            # Send a POST request with the command
            response = requests.post(url, data={'command':
command, 'login': login, 'password': password})
            # Print the response from the device
            print(response.text)
        except requests.exceptions.RequestException as e:
            # Print an error message if the request fails
            print(f"Error sending command {command}: {e}")
```

```
    # Function to update distances from the sensors
    def update_distances(self, dt):
        url = 'http://192.168.4.1/distance'
        try:
            # Send a GET request to retrieve the distance data
            response = requests.get(url)
            distances = response.json()
            # Update the labels and pyramids with the new
distances
            left_distance = distances.get('left', 0)
            right_distance = distances.get('right', 0)
            self.left_distance_label.text = f'Left Distance:
{left_distance} cm'
            self.right_distance_label.text = f'Right Distance:
{right_distance} cm'

self.left_distance_pyramid.update_pyramid(left_distance)

self.right_distance_pyramid.update_pyramid(right_distance)
        except requests.exceptions.RequestException as e:
            # Print an error message if the request fails
            print(f"Error getting distances: {e}")

# Run the application
if __name__ == '__main__':
    MyApp().run()
```

Лістинг Г.2 – Налаштування та алгоритм роботи мікроконтролера

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <NewPing.h>

// Wi-Fi credentials
const char *ssid = "NodeMCU";        // SSID for the Wi-Fi
network
const char *password = "11112222";  // Password for the Wi-Fi
network
String text;                         // Text variable to hold
response messages

// Motor control pins
const int motorA1A = D0;             // Motor A control pin 1
const int motorA1B = D1;             // Motor A control pin 2
const int motorB1A = D2;             // Motor B control pin 1
const int motorB1B = D3;             // Motor B control pin 2

// Ultrasonic sensor pins
#define TRIGGER_PIN_1 D5             // Trigger pin for left
ultrasonic sensor
#define ECHO_PIN_1 D6                // Echo pin for left
ultrasonic sensor
```

```
#define TRIGGER_PIN_2 D7            // Trigger pin for right
ultrasonic sensor
#define ECHO_PIN_2 D8               // Echo pin for right
ultrasonic sensor
#define MAX_DISTANCE 400            // Maximum distance to ping
(in centimeters)

// Create NewPing objects for each ultrasonic sensor
NewPing sonar_1(TRIGGER_PIN_1, ECHO_PIN_1, MAX_DISTANCE);
NewPing sonar_2(TRIGGER_PIN_2, ECHO_PIN_2, MAX_DISTANCE);

// Create a web server object on port 80
ESP8266WebServer server(80);

bool autopilot = false;  // Flag to control autopilot mode

// Handle incoming commands from the web client
void handleCommand() {
  String command = server.arg("command");  // Get the command
from the request

  Serial.println("Received command: " + command);  // Print the
command to the serial monitor

  // Handle the "Auto" command to toggle autopilot mode
  if (command == "Auto") {
    autopilot = !autopilot;  // Toggle the autopilot flag

    if (autopilot == 0) {    // If autopilot is turned off
      motorA_stop();         // Stop motor A
      motorB_stop();         // Stop motor B
    }

    text = autopilot ? "Autopilot ON" : "Autopilot OFF";  // Set
response message based on autopilot status
  }
  // Handle manual motor commands if autopilot is off
  else if (!autopilot) {
    if (command == "Left") {
      motorA_forward();
      text = "Motor A Left";
    } else if (command == "Right") {
      motorA_backward();
      text = "Motor A Right";
    } else if (command == "WheelStop") {
      motorA_stop();
      text = "Motor A Stop";
    } else if (command == "Forward") {
      motorB_forward();
      text = "Motor B Forward";
    } else if (command == "Backward") {
      motorB_backward();
      text = "Motor B Backward";
```

```
    } else if (command == "MoveStop") {
      motorB_stop();
      text = "Motor B Stop";
    } else {
      text = "Command not recognized";  // Handle unrecognized
commands
    }
  }

  // Send the response back to the client
  server.send(200, "text/plain", text);
}

// Handle distance measurement requests from the web client
void handleDistance() {
  unsigned int distance_left = sonar_1.ping_cm();   // Get
distance from left sensor
  unsigned int distance_right = sonar_2.ping_cm();  // Get
distance from right sensor

  // Create a JSON response with the distances
  String json = "{\"left\": " + String(distance_left) + ",
\"right\": " + String(distance_right) + "}";
  server.send(200, "application/json", json);  // Send the JSON
response
}

void setup() {
  Serial.begin(115200);  // Start the serial communication
  delay(10);  // Small delay to ensure stability

  // Setup motor control pins as outputs
  pinMode(motorA1A, OUTPUT);
  pinMode(motorA1B, OUTPUT);
  pinMode(motorB1A, OUTPUT);
  pinMode(motorB1B, OUTPUT);

  // Setup Wi-Fi in Access Point mode
  WiFi.softAP(ssid, password);
  IPAddress ip = WiFi.softAPIP();  // Get the IP address of the
Access Point
  Serial.print("IP Address: ");
  Serial.println(ip);  // Print the IP address to the serial
monitor

  // Setup web server routes
  server.on("/command", HTTP_POST, handleCommand);  // Route for
handling commands
  server.on("/distance", HTTP_GET, handleDistance); // Route for
handling distance requests
  server.begin();  // Start the web server
  Serial.println("HTTP server started");  // Print confirmation
message
```

```
}

void loop() {
  server.handleClient();  // Handle incoming client requests

  if (autopilot) {  // If autopilot mode is enabled
    unsigned int distance_left = sonar_1.ping_cm();   // Get
distance from left sensor
    unsigned int distance_right = sonar_2.ping_cm();  // Get
distance from right sensor

    // If both sensors detect an obstacle within 50 cm, stop the
motors
    if ((distance_left <= 75 && distance_right <= 50) &&
(distance_left >= 2 && distance_right >= 2)) {
      motorA_stop();
      motorB_stop();
    } else {
      motorB_forward();  // Move forward by default in autopilot
mode

      // Adjust motor A based on sensor readings
      if (distance_right <= 50 && distance_right >= 2) {
        motorA_forward();  // Turn left
      } else if (distance_left <= 50 && distance_left >= 2) {
        motorA_backward(); // Turn right
      } else {
        motorA_stop();     // Move straight
      }
    }
  }
}

// Function to move motor A forward
void motorA_forward() {
  digitalWrite(motorA1A, HIGH);
  digitalWrite(motorA1B, LOW);
}

// Function to move motor A backward
void motorA_backward() {
  digitalWrite(motorA1A, LOW);
  digitalWrite(motorA1B, HIGH);
}

// Function to stop motor A
void motorA_stop() {
  digitalWrite(motorA1A, LOW);
  digitalWrite(motorA1B, LOW);
}

// Function to move motor B backward
void motorB_backward() {
```

```
  digitalWrite(motorB1A, HIGH);
  digitalWrite(motorB1B, LOW);
}

// Function to move motor B forward
void motorB_forward() {
  digitalWrite(motorB1A, LOW);
  digitalWrite(motorB1B, HIGH);
}

// Function to stop motor B
void motorB_stop() {
  digitalWrite(motorB1A, LOW);
  digitalWrite(motorB1B, LOW);
}
```