ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ІНФОРМАЦІЙНА СИСТЕМА ДИСТАНЦІЙНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ МОБІЛЬНОГО РОБОТИЗОВАНОГО КОМПЛЕКСУ»

зі спеціальності *126 Інформаційні системи та технології*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційні системи та технології*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Марія КОЗАЧОК
(підпис)

Виконала: здобувачка вищої освіти групи ІСД-41
Марія КОЗАЧОК
(прізвище, ім'я)

Керівник к.т.н., доцент Наталія ЛАЩЕВСЬКА
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Бакалавр

Спеціальність 126 Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____Каміла СТОРЧАК
“___” _____ 2026 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Козачок Марія Сергіївна

(прізвище, ім’я, по-батькові здобувача)

1. Тема кваліфікаційної роботи: «Інформаційна система дистанційного керування та моніторингу роботизованого комплексу».

керівник кваліфікаційної роботи Лащевська Наталія, к.т.н., доцент.

(прізвище, ім’я, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від « 20 » лютого 2026 року № 51.

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 01.06.2026 р.

3. Вихідні дані до кваліфікаційної роботи:

Офіційна документація до системного та спеціалізованого програмного забезпечення, науково-технічна література, технічні паспорти та документація роботизованого комплексу, дані про існуючі архітектури систем керування та моніторингу в робототехніці, результати українських та закордонних наукових досліджень у сфері автоматизації, робототехніки та Інтернету речей (IoT).

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Аналіз існуючих рішень систем керування та моніторингу роботизованих комплексів.

4.2 Технічне проєктування інформаційної системи.

4.3 Розробка та інтеграція програмного забезпечення системи керування та моніторингу.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «20» 02 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів Кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	01.03.2026	
2	Дослідження існуючих рішень систем керування та моніторингу роботизованих комплексів	15.03.2026	
3	Вибір технологічного стеку, постановка технічного завдання на розробку	17.03.2026	
4	Проєктування структури, архітектури та користувацького інтерфейсу	04.04.2026	
5	Програмна реалізація та тестування роботи системи	16.05.2026	
6	Розробка демонстраційних матеріалів	27.05.2026	
7	Попередній захист дипломної роботи	02.06.2026	

Здобувачка вищої освіти _____ Марія КОЗАЧОК

(підпис) *(ім'я, прізвище)*

Керівник кваліфікаційної роботи _____ Наталія ЛАЩЕВСЬКА

(підпис) *(ім'я, прізвище)*

РЕФЕРАТ

Дипломна робота виконана на 80 сторінках друкованого тексту, містить 2 рисунків, 10 таблиць, 1 додаток, перелік використаних джерел нараховує 15 найменувань.

Об'єкт дослідження — мобільний роботизований комплекс на базі платформи SunFounder Zeus Car, побудований з використанням мікроконтролерної плати Arduino UNO R3, модуля бездротового зв'язку ESP32-CAM, набору датчиків (ультразвукового, інфрачервоних, магнітометра) та чотириколісного шасі з омнінапрямленими колесами Mecanum.

Предмет дослідження — методи та засоби побудови інформаційної системи дистанційного керування та моніторингу мобільного роботизованого комплексу з використанням бездротових технологій передавання даних.

Мета роботи — розроблення апаратно-програмної інформаційної системи, що забезпечує дистанційне керування мобільним роботом через бездротовий канал зв'язку, отримання відеопотоку у режимі від першої особи (FPV), збирання та візуалізацію телеметричних даних з борту робота, а також ранню діагностику аномальних станів апаратної частини.

Методи дослідження — методи системного аналізу, теорії автоматичного керування, цифрової обробки сигналів, цифрових кінематичних моделей колісних роботів, теорії комп'ютерних мереж, об'єктно-орієнтованого та структурного програмування.

Результати роботи — розроблено інформаційну модель чотирирівневої архітектури мобільного робота; обґрунтовано вибір апаратних компонентів; реалізовано програмне забезпечення мікроконтролера для керування рухом,

обробки даних з датчиків, обчислення курсу за допомогою ПІД-регулятора та формування телеметричних повідомлень; побудовано веб-інтерфейс оператора з функціями керування та моніторингу;

Практичне значення роботи полягає у можливості використання запропонованих архітектурних рішень для побудови наземних роботизованих систем подвійного призначення — для пошуково-рятувальних, інспекційних, навчальних та інших застосувань.

Ключові слова: МОБІЛЬНИЙ РОБОТ, ДИСТАНЦІЙНЕ КЕРУВАННЯ, МОНІТОРИНГ, ТЕЛЕМЕТРІЯ, ARDUINO, ESP32-CAM, WEBSOCKET, MJPEG, КОЛЕСА MECANUM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АЦП — аналого-цифровий перетворювач;

ВКР — випускна кваліфікаційна робота;

ВС — вбудована система (англ. embedded system);

ЕОМ — електронна обчислювальна машина;

ІС — інформаційна система;

ІЧ — інфрачервоний;

КМР — колісний мобільний робот;

МК — мікроконтролер;

ОС — операційна система;

ПЗ — програмне забезпечення;

ПК — персональний комп'ютер;

ПІД — пропорційно-інтегрально-диференційний (регулятор);

СВ — система живлення;

AP (англ. Access Point) — точка доступу WiFi;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

BMS (англ. Battery Management System) — система управління акумулятором;

CPU (англ. Central Processing Unit) — центральний процесор;

DPS (англ. Digital Position Sensor) — цифровий датчик положення;

EEPROM (англ. Electrically Erasable Programmable ROM) — електрично стирана перепрограмована постійна пам'ять;

FPV (англ. First Person View) — вид від першої особи;

GPIO (англ. General Purpose Input/Output) — порти введення-виведення загального призначення;

HTTP (англ. HyperText Transfer Protocol) — протокол передачі гіпертексту;

I²C (англ. Inter-Integrated Circuit) — двопровідний послідовний інтерфейс;

IoT (англ. Internet of Things) — Інтернет речей;

IP (англ. Internet Protocol) — мережевий протокол Інтернету;

LDO (англ. Low-Dropout Regulator) — лінійний стабілізатор напруги з низьким падінням;

MJPEG (англ. Motion JPEG) — формат відео як послідовності JPEG-кадрів;

PWM (англ. Pulse Width Modulation) — широтно-імпульсна модуляція;

REST (англ. Representational State Transfer) — архітектурний стиль вебсервісів;

RGB (англ. Red-Green-Blue) — трикомпонентна колірна модель;

RTT (англ. Round-Trip Time) — час повного циклу пересилання пакета;

SoC (англ. State of Charge) — рівень заряду акумулятора;

SPI (англ. Serial Peripheral Interface) — послідовний периферійний інтерфейс;

SSID (англ. Service Set Identifier) — ідентифікатор бездротової мережі;

STA (англ. Station) — клієнтський режим WiFi;

TCP (англ. Transmission Control Protocol) — протокол керування передачею;

UART (англ. Universal Asynchronous Receiver-Transmitter) — універсальний асинхронний приймач-передавач;

URL (англ. Uniform Resource Locator) — уніфікований локатор ресурсу;

WS (англ. WebSocket) — повнодуплексний протокол обміну поверх TCP;

ВСТУП

Актуальність теми. Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням ролі автономних та напівавтономних робототехнічних систем у різних сферах людської діяльності. Мобільні роботизовані комплекси перестали бути лабораторними цікавинками: вони знаходять застосування у промисловому виробництві, логістиці, сільському господарстві, оборонній галузі, рятувальних операціях, сферах інспекції об'єктів критичної інфраструктури тощо. За оцінками експертів галузі, обсяг світового ринку мобільної робототехніки впродовж останнього десятиліття зростає темпами понад 20 % на рік, причому особливо динамічно розвивається сегмент малих і середніх безпілотних наземних засобів — від інспекційних робототехнічних платформ до бойових та допоміжних безпілотних наземних апаратів.

Принциповою ознакою сучасних мобільних роботів є їх інтеграція у єдиний інформаційний контур з оператором, у якому ключову роль відіграє інформаційна система дистанційного керування та моніторингу. Така система забезпечує двосторонній обмін даними між робототехнічним комплексом та віддаленим пультом оператора, надає засоби візуалізації поточного стану робота, дозволяє вчасно виявляти аномалії в роботі апаратної частини, а також забезпечує можливість делегування частини функцій інтелектуальним алгоритмам комп'ютерного зору.

Підготовка інженерних кадрів у галузі вбудованих систем та робототехніки потребує доступних навчально-демонстраційних платформ, які за своєю архітектурою адекватно відображають принципи побудови повномасштабних промислових і спеціальних роботизованих комплексів. Однією з таких платформ є комплект SunFounder Zeus Car Kit — чотириколісний мобільний робот з омнінапрямленими колесами Mecanum, обладнаний типовим для сучасних мобільних роботів набором датчиків і модулем бездротового відеозв'язку. Розробка інформаційної системи дистанційного керування та моніторингу для такого комплексу дозволяє на практиці опанувати весь стек технологій — від кінематики

коліс і програмування мікроконтролерів до проектування мережевих протоколів та користувацьких інтерфейсів.

Зв'язок роботи з науковими програмами. Дипломна робота виконана в межах наукового напряму кафедри комп'ютерних систем та робототехніки, який передбачає дослідження та розробку засобів інтелектуального керування мобільними роботизованими комплексами для задач подвійного призначення.

Мета і завдання дослідження. Метою роботи є розроблення інформаційної системи дистанційного керування та моніторингу мобільного роботизованого комплексу на базі платформи SunFounder Zeus Car, що забезпечує надійне бездротове керування рухом, передачу відео у режимі від першої особи та збирання й візуалізацію телеметричних даних для оператора.

Для досягнення поставленої мети у роботі вирішуються такі основні завдання:

1. Проаналізувати сучасний стан робототехнічних комплексів та сфери їх застосування, обґрунтувати актуальність задачі побудови інформаційної системи дистанційного керування та моніторингу мобільного робота.
2. Здійснити огляд теоретичних основ кінематики колісних роботів, особливостей коліс Mecanum, принципів побудови вбудованих систем та мережевих протоколів.
3. Розробити інформаційну модель та чотирирівневу архітектуру системи, що охоплює фізичний рівень, рівень керування, комунікаційний рівень та рівень оператора.
4. Обґрунтувати вибір апаратних компонентів комплексу та розробити електричну схему їх підключення.
5. Спроектувати та реалізувати програмне забезпечення мікроконтролерної частини, у тому числі модулі керування рухом, обробки сенсорних даних, обчислення курсу за допомогою ПІД-регулятора та формування телеметричних повідомлень.
6. Розробити веб-інтерфейс оператора, що об'єднує засоби керування та засоби візуалізації телеметрії.

7. Реалізувати підсистему моніторингу й прогнозної діагностики, що дозволяє вчасно виявляти аномалії в роботі апаратної частини.
. Провести експериментальне тестування системи та проаналізувати отримані результати.

Об'єктом дослідження є мобільний роботизований комплекс на базі платформи SunFounder Zeus Car Kit з модулем бездротового зв'язку ESP32-CAM.

Предметом дослідження є методи та засоби побудови інформаційної системи дистанційного керування та моніторингу мобільного роботизованого комплексу з використанням бездротових технологій передавання даних.

Методи дослідження. У роботі використовуються методи системного аналізу при формулюванні вимог та проектуванні архітектури системи; методи теорії автоматичного керування — при синтезі ПІД-регулятора курсу [1]; методи цифрової обробки сигналів — при фільтрації показань магнітометра та ультразвукового датчика; методи комп'ютерних мереж [4] — при проектуванні комунікаційного рівня системи; методи структурного та об'єктно-орієнтованого програмування — при реалізації програмного забезпечення.

Наукова новизна одержаних результатів полягає у наступному:

— Запропоновано чотирирівневу інформаційну модель мобільного роботизованого комплексу, що відображає взаємодію апаратних і програмних компонентів від рівня фізичних датчиків до рівня прикладної інтелектуальної обробки інформації.
— Розроблено адаптовану методику використання ПІД-регулятора для утримання курсу мобільного робота з омнінапрямленими колесами Mecanum у режимі польового-центричного керування (field-centric control), яка враховує специфіку Mecanum-кінематики.
— Запропоновано підхід до прогнозної діагностики стану апаратної частини мобільного робота на основі аналізу часових рядів телеметричних показників (напруги акумулятора, часу відгуку каналу зв'язку, частоти втрати кадрів).

Практичне значення одержаних результатів. Розроблена інформаційна система може бути використана:

— у навчальному процесі вищих навчальних закладів — для підготовки фахівців у галузі вбудованих систем та робототехніки;
— як демонстраційний прототип архітектурних рішень для побудови наземних роботизованих платформ подвійного призначення (інспекційних, пошуково-рятувальних, оборонних);
— як базова платформа для подальших досліджень у сфері мобільної робототехніки, комп'ютерного зору та інтелектуальних інформаційних систем.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Сучасний стан робото-технічних комплексів

Робототехніка як галузь науки і техніки сформувалася на стику кількох фундаментальних дисциплін — механіки, електроніки, теорії автоматичного керування та комп'ютерних наук. Сучасний робот є яскравим прикладом кіберфізичної системи, у якій обчислювальні компоненти невіддільно інтегровані з фізичними процесами, що відбуваються у керованому об'єкті [5]. Така інтеграція потребує одночасного розгляду апаратних і програмних аспектів — як на етапі проектування, так і на етапі експлуатації.

Залежно від призначення та умов експлуатації роботизовані системи прийнято класифікувати на стаціонарні (промислові маніпулятори) та мобільні. У свою чергу, мобільні роботи поділяються на наземні (колісні, гусеничні, крокові), повітряні (мультикоптери, безпілотні літальні апарати літакового типу) та водні (надводні і підводні). Об'єктом розгляду в даній дипломній роботі є колісний мобільний робот (КМР), який забезпечує найкращий компроміс між простотою конструкції, енергоефективністю та маневреністю для більшості наземних застосувань.

За даними аналітичних звітів галузі, у 2024–2026 роках спостерігається стійке зростання інтересу до малих та середніх мобільних робототехнічних платформ, що обумовлено такими чинниками:

- здешевленням обчислювальних компонентів (мікроконтролерів та одноплатних комп'ютерів), що дозволяє вмонтовувати все більш складні алгоритми обробки даних безпосередньо у робота;
- розвитком технологій бездротового зв'язку (WiFi нового покоління, мобільних мереж 5G, систем mesh-зв'язку), що забезпечує надійну передачу даних від робота до оператора;
- стрімким прогресом у сфері машинного навчання та комп'ютерного зору, що відкриває можливості для часткової автоматизації функцій оператора;

— суттєвим розширенням сфер застосування — від побутових прибиральних роботів до автономних транспортних засобів і безпілотних бойових платформ.

У відповідь на ці тенденції суттєво змінилися й вимоги до інформаційних систем, що супроводжують мобільні роботи. Якщо ще десятиліття тому такі системи зводилися переважно до проводового пульта керування з мінімальним набором кнопок, то у сучасних умовах від інформаційної системи дистанційного керування та моніторингу очікують такого набору функцій:

1. Передача керуючих команд з пульта оператора до робота через бездротовий канал зв'язку з мінімальною затримкою (характерне значення прийнятної затримки для FPV-керування — менше 200 мс).
2. Передача відеопотоку від бортової камери до оператора у режимі реального часу з достатньою для прийняття рішень роздільною здатністю та частотою кадрів (типово — від 640×480 при 15 кадрах/с до 1280×720 при 30 кадрах/с).
3. Збирання та передача телеметричних даних — напруги акумулятора, показань датчиків, поточного режиму роботи, часу безперервної експлуатації тощо.
4. Візуалізація телеметрії у зрозумілій оператору формі — у вигляді цифрових індикаторів, графіків, діаграм радару, кольорової індикації стану.
5. Виявлення аномальних режимів роботи (низького заряду акумулятора, перевищення безпечної температури, втрати зв'язку, відмови сенсорів) із своєчасним сповіщенням оператора.
6. Підтримка кількох режимів керування — від повністю ручного до напівавтономного (наприклад, об'їзд перешкод, слідування за лінією, утримання заданого курсу).
7. Можливість інтеграції з зовнішніми інтелектуальними системами обробки даних (наприклад, з програмами розпізнавання облич, дорожніх знаків, дорожньої розмітки на стороні оператора).

Перелічені вимоги формують предметну область розробки інформаційних систем для мобільних робототехнічних комплексів і визначають архітектурні рішення, які лягають в основу таких систем.

1.2. Сфери практичного застосування мобільних роботів

Перш ніж переходити до постановки конкретної задачі дипломної роботи, доцільно розглянути основні сфери практичного застосування мобільних колісних роботів, що дозволить краще зрозуміти контекст та цінність розроблюваної інформаційної системи.

1.2.1. Пошуково-рятувальні операції та робототехніка надзвичайних ситуацій

Однією з найважливіших сфер застосування мобільних роботів є участь у пошуково-рятувальних операціях у зонах надзвичайних ситуацій. Землетруси, обвалення будівель, техногенні катастрофи створюють умови, у яких пряма участь рятувальника пов'язана з невиправдано високим ризиком для життя. Малі мобільні роботи, обладнані відеокамерою та системою дистанційного керування, здатні проникати у завали через щілини, недоступні для людини, і передавати оператору зображення з місця ймовірного знаходження постраждалих.

Архітектурно такі рятувальні роботи близькі до досліджуваного в роботі комплексу: вони побудовані на основі компактної мобільної платформи, обладнаної відеокамерою, бездротовим модулем зв'язку та набором датчиків відстані для уникнення перешкод. Розроблена в дипломній роботі система може розглядатися як спрощений прототип апаратно-програмного забезпечення таких рятувальних роботів.

1.2.2. Розмінування та дослідження небезпечних об'єктів

Близькою за принципом до пошуково-рятувальних застосувань є сфера розмінування та дослідження об'єктів, доступ до яких пов'язаний із загрозою радіаційного, хімічного чи біологічного зараження. Мобільні роботи у цій сфері виконують роль «перших на місці події» — вони обстежують територію, ідентифікують небезпечні предмети, передають оператору детальну візуальну інформацію. Робототехнічні комплекси для розмінування комплектуються спеціалізованим маніпуляторами, проте принципи побудови їх інформаційної системи дистанційного керування — двосторонній бездротовий канал, FPV-відео, телеметрія стану — залишаються тими ж, що й у досліджуваному комплексі.

1.2.3. Логістика та склади

Логістичні центри сучасних великих компаній значною мірою автоматизовані за рахунок використання мобільних роботів-складальників. Особливо привабливим для логістики є застосування коліс Mecanum, оскільки вони дозволяють роботу рухатися у будь-якому напрямку без попереднього розвороту корпусу. Це відкриває можливість будувати склади з вужчими проходами між стелажами, що збільшує корисну площу зберігання й одночасно прискорює виконання операцій.

Інформаційна система складського робота повинна забезпечувати інтеграцію з системою управління складом (англ. Warehouse Management System), отримання завдань на переміщення, динамічне планування траєкторії в умовах присутності інших роботів та людей, а також безперервну телеметрію стану. Архітектурно такі системи багаторівневі, проте нижній — рівень окремого робота — близький до розглянутого в роботі.

1.2.4. Оборонні застосування

В останнє десятиліття, і особливо у роки повномасштабної війни в Україні, наземні роботизовані комплекси перетворилися на один з ключових елементів сучасних оборонних систем. Безпілотні наземні апарати використовуються для евакуації поранених з зони ураження, доставки боєприпасів та інших вантажів, ведення розвідки місцевості, дистанційного мінування та розмінування, ураження легкоброньованих цілей.

Архітектурна спорідненість таких роботизованих комплексів з досліджуваним об'єктом є очевидною: усі вони побудовані за схемою «пульт оператора — канал бездротового зв'язку — бортовий контролер — драйвери виконавчих пристроїв і датчиків». Опанування інженерами принципів проектування інформаційних систем для малих мобільних роботів є необхідним кроком до здатності розробляти повномасштабні оборонні роботизовані платформи.

1.2.5. Освітні застосування та підготовка інженерних кадрів

Окремою важливою сферою застосування малих мобільних робототехнічних комплексів є освіта. Сучасна підготовка інженерних кадрів вимагає компетенцій одночасно у сферах апаратного забезпечення (electronics, embedded systems) та програмного забезпечення (software development). Більшість університетських програм, які традиційно зосереджувалися на одній з цих двох сторін, виявилися недостатньо ефективними для підготовки фахівців нової генерації.

Навчальні робототехнічні платформи, до яких належить і досліджуваний у роботі комплекс SunFounder Zeus Car Kit, дозволяють студентові у межах одного навчального проєкту охопити повний цикл розробки — від схемотехніки і програмування мікроконтролера на C/C++ [2] до побудови мережевої взаємодії [4] та проектування користувацького інтерфейсу. Саме у цій логіці і виконана дана дипломна робота.

1.3. Огляд існуючих рішень дистанційного керування

Існуючі рішення у сфері дистанційного керування мобільними роботами можна класифікувати за кількома ознаками: за фізичним середовищем передачі даних, за використовуваним протоколом, за способом організації людино-машинного інтерфейсу. Розглянемо їх детальніше.

1.3.1. Засоби фізичної передачі даних

За способом фізичної передачі керуючих сигналів і телеметрії існуючі системи можна поділити на проводові і бездротові. Проводові системи мають перевагу в надійності зв'язку та необмеженій тривалості сеансу, але різко обмежують просторову свободу пересування робота і у більшості сучасних застосувань не використовуються (за винятком стаціонарних установок та лабораторних стендів).

Бездротові системи у свою чергу поділяються за типом використовуваного каналу зв'язку:

— радіоканали стандартних частот (2.4 ГГц, 5.8 ГГц) на базі модулів WiFi або Bluetooth — найбільш поширений варіант для невеликих радіусів дії (до 100–300 м);
— спеціалізовані радіоканали (наприклад, на базі LoRa, ZigBee, мережі mesh) — використовуються там, де важлива велика дальність або робота в умовах інтерференції;
— мобільні мережі (LTE, 5G) — дозволяють здійснювати керування на необмеженій відстані, проте мають вищі затримки і потребують наявності покриття;
— супутникові канали зв'язку — використовуються в найвідповідальніших застосуваннях, де відсутні інші види зв'язку, проте мають значні затримки і вартість.

У даній роботі вибрано WiFi-канал зв'язку як оптимальний для навчально-демонстраційного комплексу: він не потребує спеціалізованого обладнання, забезпечує достатню для FPV-відео пропускну здатність і використовується модулем ESP32-CAM.

1.3.2. Протокольний рівень

На рівні протоколів передачі даних у сучасних системах дистанційного керування мобільними роботами використовуються такі основні підходи:

1. Власні (proprietary) двійкові протоколи поверх UDP — забезпечують мінімальні затримки і максимальну пропускну здатність, проте складні у відлагодженні та інтеграції з третіми сторонами.
2. Текстові протоколи поверх TCP (HTTP, WebSocket) — простіші у відлагодженні, добре інтегруються з вебтехнологіями. Затримки трохи вищі, ніж у UDP, але прийнятні для більшості застосувань. Цей варіант обрано в даній роботі.
3. Стандартизовані робототехнічні протоколи (наприклад, ROS — Robot Operating System) — використовуються переважно у дослідницьких та промислових застосуваннях. Для невеликих навчальних проєктів є надлишково складними.

Особливий інтерес становить протокол WebSocket: будучи побудованим поверх HTTP, він водночас забезпечує повнодуплексний канал зв'язку зі стабільним з'єднанням, що ідеально підходить для двостороннього обміну командами і телеметрією. Деталі побудови вебсерверів на ESP-сімействі мікроконтролерів детально розглянуті в [3] — посилання на цю роботу буде неодноразово використане у подальших розділах.

1.3.3. Людино-машинний інтерфейс

Третій важливий аспект класифікації існуючих рішень — спосіб організації людино-машинного інтерфейсу оператора. Тут виділяються такі основні варіанти:

— фізичний пульт керування з власним екраном — традиційне рішення, що використовується у промислових та оборонних застосуваннях;
— мобільні додатки для смартфонів і планшетів — найбільш масовий варіант для побутових та аматорських роботів;
— веб-інтерфейс, що відкривається у звичайному браузері на ПК — універсальне рішення, що не потребує встановлення спеціалізованого ПЗ і легко інтегрується із засобами комп'ютерного зору на стороні оператора;
— спеціалізовані пульти з гарнітурою віртуальної реальності — використовуються переважно для FPV-польотів дронів.

У межах даної дипломної роботи обрано веб-інтерфейс як найбільш універсальний варіант, що відповідає сучасним вимогам до інформаційних систем дистанційного керування.

РОЗДІЛ 2. ОГЛЯД ТЕОРЕТИЧНИХ ОСНОВ

2.1. Кінематика колісних мобільних роботів

Кінематика мобільного робота описує співвідношення між узагальненими координатами окремих елементів конструкції (як правило, колесами) та координатами тіла робота в нерухомій системі координат. Залежно від конструктивних особливостей колісної бази розрізняють декілька типових моделей кінематики, з яких до КМР з омнінапрямленими колесами Mecanum найближче відноситься так звана голономна кінематика — коли робот здатний здійснювати рух у будь-якому напрямку без необхідності попереднього розвороту корпусу [1].

Розглянемо узагальнено типи коліс, які використовуються у мобільних роботах:

1. Стандартне (фіксоване) колесо — обертається лише навколо своєї горизонтальної осі. Робот, побудований на основі двох стандартних коліс із роздільним приводом, реалізує так звану диференційну кінематику (англ. differential drive) і не є голономним.
2. Кероване колесо (англ. steered wheel) — може повертатися навколо вертикальної осі. Типовий приклад — Аккерманова кінематика автомобілів.
3. Кастер (англ. caster wheel) — пасивне колесо, що вільно повертається у будь-якому напрямку. Використовується як допоміжне для забезпечення стійкості.
4. Шведське колесо (англ. Swedish wheel, omni wheel) — конструктивно складається з основного колеса, по периметру якого закріплені малі ролики з осями, перпендикулярними до осі основного колеса. Дозволяє основному колесу обертатися навколо своєї осі та одночасно вільно ковзати у поперечному напрямку.
5. Колесо Mecanum — модифікація шведського колеса, у якій осі роликів повернуті відносно осі основного колеса на 45°. Така конструкція є основою досліджуваного у роботі мобільного робота.

Для системи координат робота, орієнтованої так, що вісь X спрямована вперед, а вісь Y — праворуч від робота, узагальнене рівняння кінематики записується у вигляді:

$$\dot{\xi} = J \cdot \dot{q},$$

де $\dot{\xi} = [\dot{x}, \dot{y}, \dot{\theta}]^T$ — вектор узагальнених швидкостей робота в його власній системі координат (поздовжня, поперечна та кутова швидкість обертання); J — кінематична матриця, що залежить від конструктивних параметрів колісної бази; $\dot{q} = [\dot{\varphi}_1, \dot{\varphi}_2, ..., \dot{\varphi}_n]^T$ — вектор кутових швидкостей коліс.

Для голономних роботів матриця J не вироджена і обернена матриця J^{-1} існує — це означає, що для будь-якого бажаного вектора швидкості $\dot{\xi}$ можна знайти відповідний вектор кутових швидкостей коліс $\dot{q}$. Саме ця властивість і реалізує здатність робота рухатися у будь-якому напрямку без розвороту корпусу.

2.2. Особливості коліс Mecanum

Колесо Mecanum, винайдене у 1973 році шведським інженером Бенгтом Ілоном (Bengt Ilon), має конструктивну особливість, яка робить його унікальним інструментом для побудови омнінапрямлених мобільних роботів. По периметру основного колеса розташовані пасивні циліндричні ролики, осі яких повернуті відносно осі основного колеса на кут 45°.

При обертанні основного колеса з кутовою швидкістю ω воно створює лінійну швидкість на ободі v = ωR. Завдяки нахилу осей роликів ця швидкість розкладається на дві компоненти: одна — у напрямку обертання основного колеса (поздовжня компонента), інша — перпендикулярна до неї і відповідна напрямку осі ролика (поперечна компонента). Знак поперечної компоненти залежить від орієнтації роликів конкретного колеса і від напрямку обертання.

Для забезпечення повної керованості робот будується на чотирьох колесах Mecanum, при чому передня пара має дзеркальну орієнтацію роликів відносно задньої пари. У результаті можна виділити три базові режими руху:

— прямолінійний рух уперед — усі чотири колеса обертаються в одному напрямку з однаковими швидкостями;

— боковий рух — праве переднє і ліве заднє колеса обертаються в одному напрямку, а ліве переднє і праве заднє — у протилежному. Поперечні компоненти сил, генеровані кожним колесом, складаються, а поздовжні — взаємно компенсуються;

— розворот на місці — праві колеса обертаються в одному напрямку, ліві — у протилежному.

Усі інші траєкторії руху можуть бути отримані суперпозицією цих трьох базових режимів. Для математичного опису у системі координат тіла робота, де P_i — це швидкість обертання i-го колеса (i=0,1,2,3 у порядку: переднє ліве, переднє праве, заднє праве, заднє ліве), а v_x, v_y, ω — поздовжня, поперечна та кутова швидкість тіла, можна записати:

$$P_0 = (v\_y - v\_x - L \cdot \omega) / r,$$
$$P_1 = (v\_y + v\_x + L \cdot \omega) / r,$$
$$P_2 = (v\_y - v\_x + L \cdot \omega) / r,$$
$$P_3 = (v\_y + v\_x - L \cdot \omega) / r,$$

де r — радіус колеса; L — характерний геометричний параметр, що визначається півсумою відстаней між колесами по поздовжній і поперечній осях. Наведена система рівнянь є основою програмного драйвера керування рухом, що буде розглянутий у розділі 5.

При практичній програмній реалізації застосовується дещо спрощена та оптимізована форма, у якій бажаний напрямок руху задається кутом α (від 0° до 360°) у власній системі координат робота, бажана потужність — параметром P, а швидкість розвороту — параметром rot. Тоді швидкості коліс обчислюються за формулами:

$$P_0 = P \cdot \sin(\alpha + 90°) - P \cdot \cos(\alpha + 90°) - rot \cdot k,$$
$$P_1 = P \cdot \sin(\alpha + 90°) + P \cdot \cos(\alpha + 90°) + rot \cdot k,$$
$$P_2 = P \cdot \sin(\alpha + 90°) - P \cdot \cos(\alpha + 90°) + rot \cdot k,$$
$$P_3 = P \cdot \sin(\alpha + 90°) + P \cdot \cos(\alpha + 90°) - rot \cdot k,$$

де k — масштабний коефіцієнт, що враховує співвідношення між трансляційним і обертовим рухом (у програмній реалізації — 0.5). Зсув кута на 90° потрібен для того, щоб напрямок «вперед» відповідав α=0°. Саме ця форма реалізована у функції carMove() мобільного робота.

2.3. Принципи побудови вбудованих систем

Вбудована система (англ. embedded system) — це спеціалізована обчислювальна система, що виконує заздалегідь визначений набір функцій у складі більш великого технічного об'єкта [5]. Характерними ознаками вбудованих систем є:

- спеціалізація під вузький клас задач (на відміну від універсальних ПК);
- обмеженість обчислювальних ресурсів (пам'ять зазвичай вимірюється кілобайтами, частота процесора — десятками МГц);
- робота у режимі реального часу або близькому до нього (необхідність гарантованого часу відгуку на зовнішні події);
- безпосередня взаємодія з фізичним середовищем через сенсори та виконавчі пристрої;
- відсутність повноцінної операційної системи у класичному розумінні (хоча на сучасних більш потужних МК встановлюються легкі RTOS).

Класичним прикладом вбудованої системи у складі мобільного робота є мікроконтролерна плата на базі ATmega328P, яка використовується у Arduino UNO R3. Цей мікроконтролер має 32 КБ Flash-пам'яті програм, 2 КБ оперативної пам'яті SRAM та 1 КБ електрично стираної постійної пам'яті EEPROM, тактується частотою 16 МГц і містить вбудовані периферійні модулі — таймери, АЦП, інтерфейси UART, I²C та SPI.

Програмування мікроконтролерів сімейства AVR (до якого належить ATmega328P) у середовищі Arduino IDE здійснюється мовою C/C++ з використанням розширеної бібліотеки спрощених інтерфейсів роботи з периферією [2]. Стандартна програма Arduino має дві обов'язкові функції: setup()

— що виконується одноразово при ввімкненні живлення або скиданні, та loop() — що виконується циклічно протягом усього часу роботи системи. Така проста модель виконання робить Arduino IDE доступним для широкого кола розробників, проте водночас не накладає принципових обмежень на можливості коду — на низькому рівні залишається повний доступ до всіх регістрів мікроконтролера.

При проектуванні програмного забезпечення вбудованих систем особлива увага приділяється:

1. Економії пам'яті — використанню типів даних мінімально необхідної розрядності (uint8_t замість int, де достатньо), розміщенню великих констант у програмній пам'яті (модифікатор PROGMEM в Arduino) замість оперативної.
2. Економії процесорного часу — уникненню зайвих перетворень типів, використанню табличних обчислень замість прямих математичних операцій (зокрема для тригонометричних функцій), застосуванню цілочисельної арифметики замість плаваючої точки там, де це можливо.
3. Передбачуваності часу виконання — уникненню необмежених циклів очікування, належному застосуванню переривань, реалізації простих скінченних автоматів для організації логіки роботи.
4. Енергоефективності — використанню режимів зниженого енергоспоживання у моменти, коли робота процесора не потрібна.

Ці принципи лежать в основі програмного коду, розробленого в межах дипломної роботи, і будуть детально розглянуті у відповідних розділах.

2.4. Архітектура комп'ютерних мереж IoT-пристроїв

Сучасний мобільний робот є типовим представником пристроїв Інтернету речей (IoT) і повинен взаємодіяти з зовнішнім світом через мережеві канали зв'язку. Основні теоретичні засади побудови комп'ютерних мереж викладені у класичній праці А. Таненбаума [4], яка слугує методологічною основою при проектуванні комунікаційного рівня розроблюваної системи.

Базова модель архітектури комп'ютерних мереж — це багаторівнева модель OSI (Open Systems Interconnection), яка виділяє сім рівнів абстракції — від фізичного до прикладного. Однак для практичної побудови мережевих застосувань частіше використовують спрощений стек TCP/IP, що включає чотири рівні: канальний (фізичний), мережевий, транспортний та прикладний. Розглянемо, як ці рівні реалізовані у досліджуваній системі.

2.4.1. Канальний та мережевий рівні

На канальному рівні у системі використовується стандарт IEEE 802.11 b/g/n (WiFi у діапазоні 2.4 ГГц). Модуль ESP32-CAM має вбудований чіп бездротового зв'язку, що реалізує цей стандарт повністю на апаратному рівні. Розробник на рівні мікроконтролера не взаємодіє з канальним рівнем напряму.

На мережевому рівні працює протокол IPv4. Модуль ESP32-CAM може функціонувати у двох режимах: режимі станції (STA) — коли він підключається до зовнішньої точки доступу WiFi і отримує IP-адресу через DHCP, та режимі точки доступу (AP) — коли він сам створює окрему мережу. У межах даної роботи передбачена підтримка обох режимів з можливістю налаштування через конфігураційні константи в коді.

2.4.2. Транспортний рівень

На транспортному рівні у системі використовується протокол TCP, який забезпечує надійну передачу даних з контролем доставки та порядку отримання. Хоча для деяких типів даних (наприклад, відеопотоку) теоретично було б вигідніше використовувати UDP з меншими затримками, TCP обрано з міркувань простоти реалізації та сумісності з вебтехнологіями на стороні клієнта.

Поверх TCP функціонують два протоколи прикладного рівня — HTTP для отримання сторінок веб-інтерфейсу та відеопотоку, і WebSocket для обміну командами та телеметрією. Розглянемо їх детальніше.

2.4.3. Прикладний рівень: HTTP та WebSocket

Протокол HTTP (Hypertext Transfer Protocol) використовується для двох цілей у досліджуваній системі: для віддачі статичних ресурсів веб-інтерфейсу (HTML-сторінки, CSS- та JavaScript-файлів) і для трансляції відеопотоку у форматі MJPEG. Останнє реалізується через спеціальну техніку «multipart/x-mixed-replace» — клієнт відкриває одне HTTP-з'єднання, а сервер у відповідь надсилає послідовність JPEG-кадрів, розділених спеціальними маркерами. Браузер послідовно відображає кадри, що сприймається як неперервний відеопотік.

Протокол WebSocket — це повнодуплексний протокол обміну повідомленнями поверх TCP, що дозволяє організувати двосторонню взаємодію між клієнтом і сервером із низькою затримкою. На відміну від HTTP, де клієнт повинен ініціювати кожний запит, у WebSocket-з'єднанні після початкового рукостискання обидві сторони можуть надсилати дані у будь-який момент. Це робить WebSocket ідеальним рішенням для задачі дистанційного керування і телеметрії: команди керування передаються від клієнта до робота у момент натискання кнопок, а телеметричні дані — від робота до клієнта з фіксованим інтервалом (у нашій реалізації — 60 мс) [3].

Текстовий формат повідомлень WebSocket в системі організований за принципом «префікс + JSON-документ». Префікс позначає тип повідомлення (WS+ для більшості, SC для перевірки зв'язку), а JSON-документ містить корисне навантаження. Така схема дозволяє легко розширювати протокол додаванням нових типів повідомлень без порушення зворотної сумісності.

РОЗДІЛ 3. ІНФОРМАЦІЙНА МОДЕЛЬ ТА АРХІТЕКТУРА СИСТЕМИ

3.1. Вимоги до інформаційної системи

Перед розробкою архітектури інформаційної системи дистанційного керування та моніторингу мобільного робота необхідно сформулювати чітку та повну систему вимог. Слідуючи практиці системного аналізу, поділимо вимоги на три категорії: функціональні, нефункціональні та технологічні.

3.1.1. Функціональні вимоги

До функціональних вимог відносяться вимоги, що описують функції, які система повинна виконувати:

1. Підсистема керування рухом має забезпечувати рух мобільного робота уперед, назад, ліворуч, праворуч, по діагоналях, а також обертання навколо вертикальної осі. У всіх режимах має забезпечуватися плавне регулювання швидкості від 0 до 100% від максимальної.
2. Підсистема має підтримувати наступні режими керування: ручний режим через веб-інтерфейс, режим керування з ІЧ-пульта, режим керування голосовими командами через мобільний застосунок, режим автоматичного об'їзду перешкод, режим слідування за лінією із застосуванням 8-канального датчика відтінків сірого, режим калібрування магнітометра.
3. Підсистема відеомоніторингу має забезпечувати безперервну передачу відеопотоку від бортової камери до оператора з роздільною здатністю не менше 640×480 пікселів та частотою не менше 15 кадрів/с.
4. Підсистема телеметрії має передавати оператору з фіксованим інтервалом не більше 100 мс наступні параметри: напругу акумулятора, відстань до перешкоди попереду, поточний курс, стан двох ІЧ-датчиків перешкод, поточний режим роботи, час безперервної експлуатації.

5. Підсистема моніторингу та діагностики має забезпечувати виявлення таких аномальних режимів: критично низького заряду акумулятора, втрати зв'язку з оператором, перевищення граничного часу відгуку, відсутності реакції на команди керування.
6. Підсистема людино-машинної взаємодії має надавати оператору веб-інтерфейс із елементами керування і візуалізації телеметрії.

3.1.2. Нефункціональні вимоги

До нефункціональних вимог відносяться вимоги, що описують характеристики системи в цілому:

1. Час відгуку системи на команду керування — не більше 200 мс від моменту натискання кнопки в інтерфейсі до фактичного початку руху мобільного робота.
2. Затримка передачі відеопотоку — не більше 500 мс.
3. Час автономної роботи від повністю зарядженого акумулятора — не менше 60 хвилин (за паспортом — до 90 хвилин).
4. Маса мобільного робота у зборі — не більше 1.5 кг.
5. Дальність дії бездротового каналу зв'язку — не менше 30 м у умовах прямої видимості.
6. Робочий діапазон температур — від 0 до +40°C (нормальна кімнатна експлуатація).

3.1.3. Технологічні вимоги

До технологічних вимог відносяться вимоги, що накладаються на процес розробки і експлуатації системи:

1. Програмне забезпечення мікроконтролера має бути написане мовою C/C++ у середовищі Arduino IDE.

2. Веб-інтерфейс оператора має бути реалізований мовами HTML5, CSS3 та JavaScript і працювати у звичайному веб-браузері без встановлення додаткового ПЗ.
3. Модуль інтелектуальної обробки відеопотоку на стороні оператора має бути написаний мовою Python з використанням бібліотеки OpenCV.

3.2. Чотирирівнева інформаційна модель

На основі сформульованих вимог пропонується чотирирівнева інформаційна модель системи, що відображає взаємодію апаратних і програмних компонентів від рівня фізичних датчиків і виконавчих пристроїв до рівня прикладної інтелектуальної обробки інформації на стороні оператора. Узагальнена структура моделі представлена на рис. 3.1.

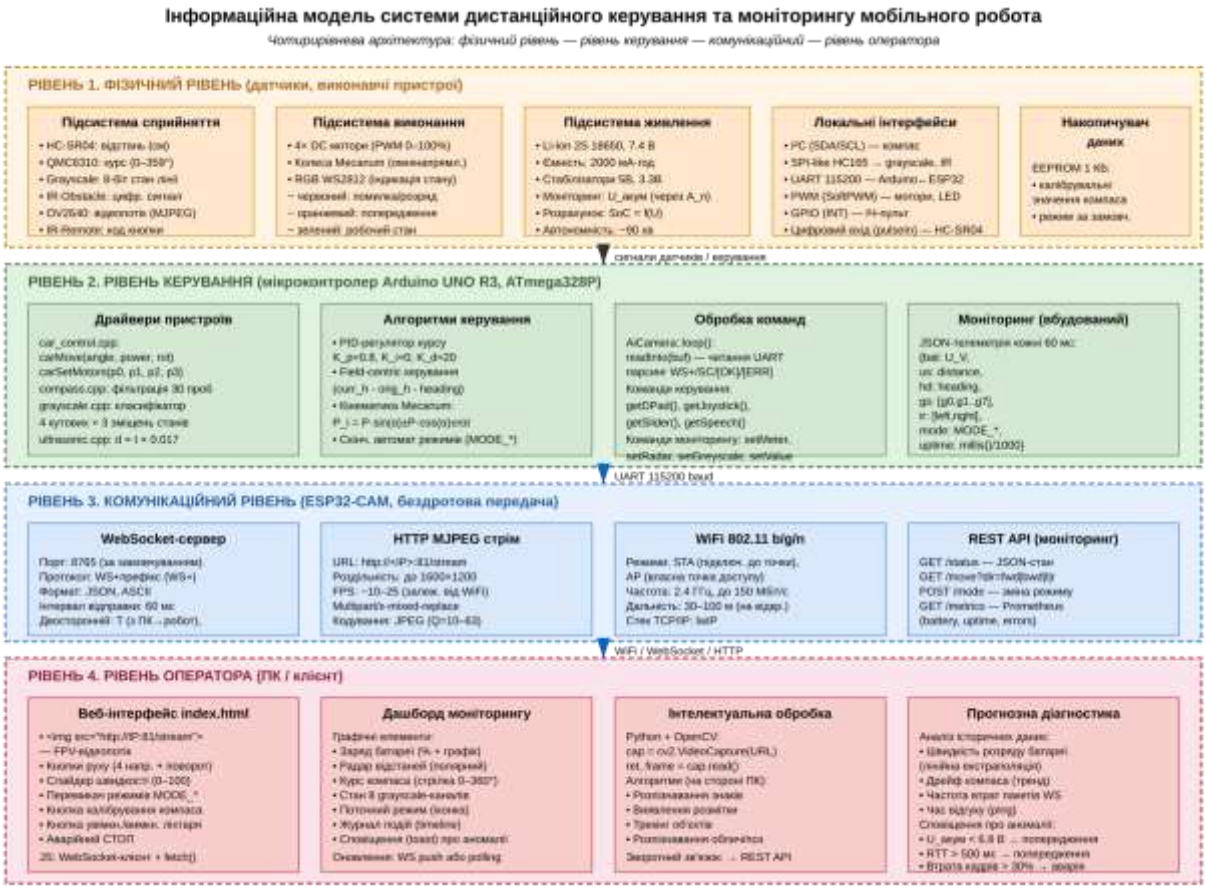


Рисунок 3.1 — Інформаційна модель системи дистанційного керування та моніторингу мобільного робота

Розглянемо детальніше кожен з виділених рівнів.

3.2.1. Фізичний рівень

Фізичний рівень об'єднує апаратні компоненти, що безпосередньо взаємодіють з фізичним середовищем. До цього рівня входять:

— підсистема сприйняття, що включає ультразвуковий датчик відстані HC-SR04, магнітометр QMC6310 для визначення курсу, 8-канальний датчик відтінків сірого для слідування за лінією, два інфрачервоних датчики уникнення перешкод (лівий і правий), модуль камери OV2640 у складі ESP32-CAM, ІЧ-приймач для дистанційного пульта;

— підсистема виконання, що включає чотири двигуни постійного струму з омнінапрямленими колесами Mecanum та світлодіодну RGB-стрічку індикації стану;

— підсистема живлення на базі літій-іонного акумулятора 7.4 В ємністю 2000 мА·год із вбудованими стабілізаторами напруги для живлення мікроконтролера (5 В) та модулів датчиків (3.3 В);

— локальні апаратні інтерфейси, що зв'язують компоненти між собою — I²C для магнітометра, SPI-подібний інтерфейс HC165 для розширення кількості цифрових входів, UART 115200 бод для зв'язку з ESP32-CAM, PWM для керування двигунами і світлодіодами;

— накопичувач EEPROM ємністю 1 КБ у складі мікроконтролера, що використовується для зберігання калібрувальних параметрів магнітометра та режиму роботи за замовчуванням.

3.2.2. Рівень керування

Рівень керування реалізований на мікроконтролері Arduino UNO R3 (ATmega328P) і об'єднує програмне забезпечення нижнього рівня, що безпосередньо керує апаратурою:

— драйвери пристроїв — програмні модулі car_control.cpp, compass.cpp, ultrasonic.cpp, ir_obstacle.cpp, ir_remote.cpp, rgb.cpp, що забезпечують взаємодію з відповідними апаратними компонентами;
— алгоритми керування рухом, у тому числі обчислення швидкостей коліс за рівняннями Mecanum-кінематики, ПІД-регулятор курсу для утримання заданого напрямку, скінченний автомат режимів роботи;
— блок обробки команд від оператора, що приймає повідомлення з модуля ESP32-CAM, парсить їх та делегує виконання відповідним драйверам;
— вбудована підсистема моніторингу, що періодично (з інтервалом 60 мс) формує JSON-телеметрію стану робота для передачі оператору.

3.2.3. Комунікаційний рівень

Комунікаційний рівень реалізований переважно на модулі ESP32-CAM, що поєднує функції бездротового маршрутизатора, веб-сервера та джерела відеопотоку. До цього рівня входять:

— WebSocket-сервер, що забезпечує двосторонній обмін JSON-повідомленнями з клієнтом за фіксованою адресою порту (за замовчуванням — 8765);
— HTTP-сервер MJPEG-стріму, що віддає відеопотік від камери за URL http://<IP>:81/stream;
— блок бездротової підсистеми WiFi 802.11 b/g/n, що підтримує два режими роботи — клієнтський (STA) та точка доступу (AP);
— REST-API моніторингу, що надає альтернативний доступ до телеметрії через звичайні HTTP-запити (зручно для інтеграції з системами візуалізації типу Prometheus + Grafana).

3.2.4. Рівень оператора

Рівень оператора реалізований на стороні робочого місця користувача — як правило, персонального комп'ютера. Він включає:

— веб-інтерфейс index.html — головний засіб взаємодії оператора з роботом, що об'єднує засоби керування (кнопки руху, слайдери, перемикачі режимів) із засобами візуалізації (відеопотік, цифрові індикатори, графіки телеметрії);

— дашборд моніторингу, що візуалізує стан робота у вигляді радару відстаней, шкали заряду батареї, стрілки компаса, журналу подій;

— підсистему прогнозної діагностики, що аналізує історичні дані телеметрії та виявляє аномалії — швидкий розряд батареї, дрейф показань компаса, втрати пакетів, збільшення часу відгуку.

3.3. Структура потоків даних

Розглянемо структуру основних потоків даних у системі, рухаючись знизу вгору по рівнях інформаційної моделі.

3.3.1. Потік даних від датчиків до оператора

Аналогові й цифрові сигнали від датчиків приймаються мікроконтролером Arduino з відповідних входів. Драйвери (compass.cpp, ultrasonic.cpp тощо) перетворюють «сирі» дані у фізичні величини або стани (відстань у см, курс у градусах, бітова маска лінії). У межах головного циклу loop() значення цих величин зчитуються, агрегуються в JSON-документ і передаються через UART до модуля ESP32-CAM. ESP32-CAM, у свою чергу, надсилає JSON-повідомлення усім приєднаним WebSocket-клієнтам. На стороні оператора JavaScript-код у веб-інтерфейсі приймає повідомлення і оновлює відповідні елементи візуалізації.

3.3.2. Потік команд керування від оператора до робота

Натискання кнопок керування у веб-інтерфейсі викликає виконання JavaScript-обробників, що формують текстові команди і надсилають їх через WebSocket-з'єднання до ESP32-CAM. ESP32-CAM ретранслює команди через UART мікроконтролеру Arduino. Парсер у функції loop() Arduino-програми розпізнає тип команди (DPad, joystick, slider, switch тощо) і викликає відповідну функцію драйвера керування рухом (carMove, carForward, carStop і т. ін.).

3.3.3. Потік відеоданих

Камера OV2640 під керуванням ESP32-CAM безперервно захоплює кадри з заданою роздільною здатністю і кодує їх у JPEG. HTTP-сервер ESP32-CAM віддає кадри клієнту у форматі multipart/x-mixed-replace. Браузер автоматично відображає кадри по мірі їх надходження. Альтернативно відеопотік може бути захоплений Python-скриптом за допомогою cv2.VideoCapture для подальшої обробки засобами OpenCV.

3.3.4. Потік даних моніторингу та діагностики

Окрім поточного відображення в інтерфейсі, телеметричні дані можуть накопичуватися у локальному сховищі на стороні оператора (наприклад, у вигляді часових рядів у пам'яті браузера або файлів на диску). Періодичний аналіз цих рядів дозволяє виявляти тенденції — швидкість розряду акумулятора, дрейф показань компаса, погіршення часу відгуку — і вчасно попереджати оператора про необхідність технічного обслуговування або зміни режиму роботи.

РОЗДІЛ 4. АПАРАТНА ЧАСТИНА СИСТЕМИ

4.1. Огляд складу комплексу SunFounder Zeus Car

Мобільний роботизований комплекс SunFounder Zeus Car Kit є завершеним апаратно-програмним рішенням, призначеним для побудови малого омнінапрямленого колісного робота навчально-демонстраційного призначення. До складу комплекту входять усі компоненти, необхідні для збирання робота, а також готова бібліотека програмних драйверів та прикладів для середовища Arduino IDE.

До основного складу комплексу входять такі апаратні компоненти:

1. Мікроконтролерна плата Arduino UNO R3 на базі мікроконтролера ATmega328P, що виконує функції центрального обчислювального вузла системи.
2. Плата розширення Zeus Car Shield, що встановлюється на Arduino UNO зверху і об'єднує драйвери двигунів, стабілізатори напруги, роз'єми для підключення датчиків та модуля камери, а також інтерфейсні мікросхеми (зокрема, розширювач входів HC165).
3. Модуль бездротового зв'язку та камери ESP32-CAM з адаптером Camera Adapter, який забезпечує захоплення відеопотоку, передачу його через WiFi і функції веб-сервера для двостороннього обміну з оператором.
4. Чотири двигуни постійного струму ТТ 3-6 В з планетарним редуктором з передаточним числом 1:48 та номінальною швидкістю обертання валу 200 об/хв при напрузі живлення 6 В.
5. Чотири омнінапрямлених колеса Mecanum, що встановлюються на вали двигунів і забезпечують голономну рухливість робота.
6. Літій-іонний акумулятор Li-Ion 2S 18650 номінальною напругою 7.4 В і ємністю 2000 мА·год з вбудованою схемою захисту від глибокого розряду та перезаряду.

7. Ультразвуковий датчик відстані HC-SR04, що встановлюється у носовій частині робота і використовується для вимірювання відстані до перешкоди попереду.
8. Два інфрачервоних модулі уникнення перешкод (IR Obstacle Avoidance Module), встановлені з лівого і правого боку робота.
9. Цифровий магнітометр QMC6310, що використовується для визначення курсу робота.
10. Світлодіодна RGB-стрічка на основі чіпів WS2812, що використовується для індикації поточного режиму роботи і стану системи.
11. Інфрачервоний пульт дистанційного керування разом з ІЧ-приймачем VS1838B (несуча частота 38 кГц), що дозволяє керувати роботом без використання WiFi.

Основні технічні характеристики комплексу узагальнено в табл. 4.1.

Таблиця 4.1 — Основні технічні характеристики комплексу

Параметр	**Значення**
Основна плата	Arduino UNO R3 (ATmega328P, 16 МГц)
Модуль бездротового зв'язку	ESP32-CAM (WiFi 802.11 b/g/n, 2.4 ГГц)
Камера	OV2640, до 1600×1200 (UXGA)
Привод	4× DC двигуни TT 3-6В, передаточне число 1:48
Тип коліс	Mecanum (омнінапрямлені)
Акумулятор	Li-Ion 2S 18650, 7.4 В, 2000 мА·год
Діапазон живлення	+6...+8.4 В (роз'єм PH2.0-5P)
Час автономної роботи	до 90 хвилин
Час повного заряджання	130 хвилин
Габаритні розміри	168.91 × 165.1 × 17.78 мм
Середовище програмування	Arduino IDE, мова C/C++

Способи керування	Веб-інтерфейс, мобільний застосунок (FPV), ІЧ-пульт, голос

4.2. Обґрунтування вибору компонентів

Кожен з компонентів, що входять до складу комплексу, виконує специфічну функцію в системі. У цьому підрозділі розглянемо, чому саме ці компоненти доцільно використовувати у мобільному роботі досліджуваного класу, і де ще подібні рішення знаходять застосування.

4.2.1. Мікроконтролерна плата Arduino UNO R3

Плата Arduino UNO R3 на базі мікроконтролера ATmega328P є одним з найпоширеніших рішень для прототипування вбудованих систем і освітніх проєктів. Її вибір як основного обчислювального вузла мобільного робота обумовлений наступними факторами:

1. Простота освоєння та розгалужена екосистема — середовище Arduino IDE з потужним менеджером бібліотек, велика спільнота розробників, наявність прикладів коду та документації [2].
2. Достатність обчислювальних ресурсів для розв'язуваних задач — 32 КБ Flash-пам'яті, 2 КБ оперативної пам'яті, 1 КБ EEPROM, 16 МГц тактової частоти. Цього достатньо для одночасного керування 4 двигунами (через SoftPWM), обробки даних з 5-6 датчиків і обміну даними по UART.
3. Низька вартість і доступність комплектуючих, що робить можливим масове використання у навчальних цілях.
4. Сумісність з широким спектром плат розширення (shields) промислового виробництва, у тому числі з Zeus Car Shield.

На рівні вбудованих систем платформа Arduino UNO дозволяє продемонструвати усі ключові концепції програмування мікроконтролерів — роботу з цифровими і аналоговими портами введення-виведення, переривання, таймери, послідовні інтерфейси, ШІМ-керування, читання даних з АЦП [2,5].

У реальних промислових застосуваннях Arduino UNO зазвичай замінюється більш потужними мікроконтролерами (наприклад, з ядром ARM Cortex-M) або одноплатними комп'ютерами (Raspberry Pi, NVIDIA Jetson), проте архітектурні принципи побудови програмного забезпечення залишаються тими ж.

4.2.2. Плата розширення Zeus Car Shield

Плата розширення Zeus Car Shield є інтегрованим рішенням, що об'єднує кілька функціональних блоків і встановлюється на Arduino UNO як shield. Її використання дає такі переваги:

- на платі знаходяться два чотириканальних драйвери H-моста для керування чотирма двигунами постійного струму. Кожен канал розрахований на струм до 1.5 А, що з певним запасом перекриває максимальний споживаний струм використовуваних двигунів (близько 600 мА у пусковому режимі);
- на платі встановлено стабілізатор напруги LDO, що знижує напругу акумулятора 7.4 В до 5 В для живлення мікроконтролера і логічної частини датчиків;
- плата містить роз'єм для підключення модуля ESP32-CAM через UART і одночасно подачі живлення;
- на платі встановлено мікросхему-розширювач HC165 (зсувний регістр 8 біт), що дозволяє читати показання 8-канального датчика відтінків сірого та двох ІЧ-датчиків перешкод з використанням всього трьох цифрових ліній мікроконтролера, економлячи цінні GPIO-порти.

Подібні інтегровані плати розширення характерні для багатьох сучасних мобільних роботів. Вони суттєво спрощують монтаж та зменшують кількість можливих помилок при ручному монтажі компонентів.

4.2.3. Модуль ESP32-CAM

Модуль ESP32-CAM від компанії Espressif Systems поєднує в одній компактній платі такі функціональні блоки:

— двоядерний мікроконтролер ESP32 з тактовою частотою до 240 МГц, 520 КБ оперативної пам'яті та 4 МБ зовнішньої Flash-пам'яті;
— вбудований чіп бездротового зв'язку WiFi 802.11 b/g/n і Bluetooth (хоча останній у нашій системі не використовується);
— камеру OV2640 з максимальною роздільною здатністю 1600×1200 пікселів, з можливістю програмного вибору меншої роздільної здатності для досягнення вищої частоти кадрів;
— слот для карти microSD (опціонально використовується для буферизації відео);
— вбудований світлодіодний ліхтарик (lamp) на платі камери, що може використовуватися як підсвічування.

Вибір ESP32-CAM як модуля бездротового зв'язку обумовлений насамперед тим, що цей модуль вже інтегрує камеру, мікроконтролер з достатньою обчислювальною потужністю для обробки відеопотоку (кодування у JPEG, формування multipart HTTP-відповіді) та чіп WiFi для бездротової передачі — що дозволяє реалізувати функціональність FPV-моніторингу і дистанційного керування єдиним монолітним рішенням [3].

Серед недоліків ESP32-CAM варто відзначити досить високе енергоспоживання у режимі активної трансляції відео (до 250 мА) і необхідність акуратної організації джерела живлення для забезпечення стабільної роботи. У досліджуваному комплексі живлення ESP32-CAM подається з окремої лінії 5 В з достатнім запасом по потужності.

4.2.4. Двигуни постійного струму та колеса Mecanum

Двигуни постійного струму ТТ з планетарним редуктором є типовим рішенням для малих мобільних роботів. Технічні характеристики двигунів подані в табл. 4.2.

Таблиця 4.2 — Технічні характеристики двигунів ТТ

Параметр	**Значення**
Номінальна напруга живлення	3...6 В
Безперервний струм холостого ходу	150 мА ± 10%
Мін. робоча швидкість при 3 В	90 ± 10% об/хв
Мін. робоча швидкість при 6 В	200 ± 10% об/хв
Крутний момент звалювання при 3 В	0.4 кг·см
Крутний момент звалювання при 6 В	0.8 кг·см
Передаточне число редуктора	1:48
Габаритні розміри корпусу	70 × 22.3 × 36.9 мм
Маса	30.6 г
Роз'єм для підключення	XH2.54-2P

Використання саме коліс Mecanum (а не звичайних чи шведських) обумовлене їх унікальною властивістю — здатністю забезпечувати рух у будь-якому напрямку без розвороту корпусу. Як було показано у розділі 2, ця властивість дозволяє реалізовувати голономну кінематику і відкриває нові можливості для маневрування у обмеженому просторі. Сферами, де така здатність є критично важливою, є логістичні центри (рух роботів у вузьких проходах між стелажами), виробничі цехи з щільним розміщенням обладнання, інспекційні роботизовані системи на промислових об'єктах.

4.2.5. Літій-іонний акумулятор Li-Ion 2S 18650

Літій-іонний акумулятор форм-фактора 18650 у послідовній конфігурації 2S (тобто два елементи, з'єднані послідовно для отримання номінальної напруги 7.4 В) є стандартним рішенням для мобільних електронних пристроїв середньої потужності. До переваг такого джерела живлення відносяться:

— висока питома енергоємність (до 250 Вт·год/кг для сучасних літій-іонних елементів);
— стандартний форм-фактор і доступність заміни;
— наявність вбудованих схем захисту від глибокого розряду, перезаряду і короткого замикання у фабричних збірках;
— прийнятна вартість і поширеність на ринку.

Номінальна ємність 2000 мА·год при середньому струмі споживання робота приблизно 1.3-1.5 А (з відеопотоком та активним рухом) забезпечує заявлений час автономної роботи близько 90 хвилин. Більш детальний розрахунок енергоспоживання наведено у п. 4.4.

4.2.6. Ультразвуковий датчик HC-SR04

Ультразвуковий датчик відстані HC-SR04 — один з найпоширеніших датчиків у любительській та освітній робототехніці. Принцип його дії базується на вимірюванні часу проходження ультразвукового імпульсу частотою 40 кГц від датчика до перешкоди і назад.

Технічні характеристики:

— діапазон вимірюваних відстаней: 2...400 см (у досліджуваній системі обмежений константою MAX_DISTANCE = 2000 см на рівні ПЗ);
— точність вимірювання: ±3 мм;
— кут спрямованості: близько 15°;
— робоча напруга: 5 В;
— струм споживання: близько 15 мА у активному режимі.

Датчик HC-SR04 встановлений у носовій частині робота і використовується для двох основних задач: попередження зіткнення з перешкодами при русі вперед

та реалізація режиму автоматичного об'їзду перешкод. У програмному драйвері ultrasonic.cpp реалізована також передача чисельного значення відстані на дашборд оператора у вигляді радару.

4.2.7. Магнітометр QMC6310

Триосьовий цифровий магнітометр QMC6310 від компанії QST використовується для визначення курсу мобільного робота відносно магнітного поля Землі. Принцип роботи базується на ефекті анізотропного магнітоопору, що дозволяє вимірювати компоненти магнітного поля вздовж трьох взаємно перпендикулярних осей.

Основні характеристики:

— діапазон вимірюваного магнітного поля: ±2/±8/±12/±30 Гауссів (програмований);

— розрядність АЦП: 16 біт на кожну вісь;

— частота вимірювань: до 200 Гц;

— інтерфейс зв'язку: I²C, базова адреса 0x1C;

— напруга живлення: 1.8-3.6 В.

Інформація про курс необхідна для реалізації режиму field-centric керування, у якому напрямки руху задаються у нерухомій системі координат (поля, або абсолютній) незалежно від поточної орієнтації корпусу робота. Це інтуїтивно зручніше для оператора: натиснута кнопка «вперед» означає рух уперед від оператора, а не уперед відносно носа робота. Реалізація цього режиму вимагає постійного знання поточного курсу робота для коригування завдання на рух з урахуванням повороту корпусу.

4.2.8. Інфрачервоні датчики уникнення перешкод

Два модулі ІЧ-уникнення перешкод (один з лівого, інший з правого боку робота) використовують принцип відбиття ІЧ-світла від перешкоди. Кожен модуль містить ІЧ-світлодіод-випромінювач і фотоприймач. Якщо перед модулем

знаходиться об'єкт на відстані менше встановленого порога (типово 5-15 см, регулюється підлаштовуваним резистором на платі), то сигнал виходу переходить у логічний низький рівень.

На відміну від ультразвукового датчика, ІЧ-датчики дають лише бінарну інформацію (є перешкода чи немає), проте вони працюють набагато швидше і не потребують обчислень. Це робить їх ідеальним рішенням для бічного контролю — наприклад, для уникнення зачеплення стін у вузьких коридорах при русі вперед.

4.2.9. Світлодіодна RGB-стрічка WS2812

Світлодіодна RGB-стрічка на основі чіпів WS2812 використовується для індикації поточного режиму роботи робота та сигналізації аномалій. Кожен світлодіод стрічки містить вбудований контролер, що дозволяє програмно задавати інтенсивність кожного з трьох колірних каналів (R, G, B) у діапазоні 0-255.

У досліджуваній системі різним режимам роботи робота відповідають різні кольори:

— білий (0xFFFFFF) — режим за замовчуванням, очікування команди;

— зелено-блакитний (0x0AFFA5) — режим керування ІЧ-пультом;

— маджента (0xFF0AA5) — режим керування веб-застосунком;

— блакитний (0x0AFFFF) — режим слідування за лінією без магнітометра;

— зелений (0x0AFF0A) — режим слідування за лінією з магнітометром;

— синій (0x0A0AFF) — режим автономного слідування за перешкодою;

— фіолетовий (0xA50AFF) — режим автономного об'їзду перешкод;

— оранжевий (0xFFA500) — попередження (наприклад, низький заряд акумулятора);

— червоний (0xFF0202) — критична помилка.

Така кольорова кодифікація дозволяє визначити поточний стан робота візуально, без необхідності перегляду телеметрії на екрані оператора.

4.2.10. Інфрачервоний пульт дистанційного керування

ІЧ-пульт використовується як резервний канал керування на випадок проблем з WiFi-зв'язком, а також як зручний інструмент для калібрування і тестування робота на близькій відстані. ІЧ-приймач VS1838B налаштований на несучу частоту 38 кГц і використовує стандартний протокол NEC для декодування команд.

Підключення ІЧ-приймача до піна D2 мікроконтролера дозволяє використовувати апаратне зовнішнє переривання (INT0), що забезпечує миттєву реакцію на надходження ІЧ-кадру незалежно від поточної фази головного циклу loop().

4.3. Аналіз використання ресурсів мікроконтролера

Програмне забезпечення вбудованих систем принципово відрізняється від звичайних застосунків для ПК тим, що розробник постійно стикається з жорсткими обмеженнями за обсягом доступної пам'яті [5]. Перш ніж переходити до детального розгляду програмних модулів, необхідно переконатися, що сукупність розроблених компонентів вкладається у наявні ресурси мікроконтролера ATmega328P, який використовується у Arduino UNO R3.

Згідно з документацією виробника [8], мікроконтролер ATmega328P має такі ресурси пам'яті: 32 КБ Flash-пам'яті програм (з яких приблизно 1 КБ зайнято завантажувачем Arduino, тому реально доступно близько 31 КБ); 2 КБ оперативної пам'яті SRAM, у якій розміщуються глобальні змінні, стек і динамічно виділені дані; 1 КБ електрично стираної постійної пам'яті EEPROM, що використовується для зберігання калібрувальних параметрів між сеансами роботи.

Найбільш «вузьким» ресурсом є SRAM. Переповнення Flash-пам'яті виявляється під час компіляції (компілятор повідомляє про неможливість зв'язування), тоді як перевитрата SRAM проявляється на етапі виконання — як непередбачувані перезавантаження, спотворення глобальних змінних та інші важко діагностовані аномалії, оскільки стек починає накладатися на область глобальних

змінних. Тому при проектуванні було приділено особливу увагу економії оперативної пам'яті.

5.2.1. Бюджет Flash-пам'яті програм

Орієнтовний розподіл Flash-пам'яті між компонентами програмного забезпечення наведено у табл. 5.2. Значення отримані за результатами компіляції повного скетча у середовищі Arduino IDE з увімкненими стандартними оптимізаціями (-Os).

Таблиця 5.2 — Орієнтовний бюджет Flash-пам'яті (32 КБ доступних, мінус 1 КБ завантажувача)

Компонент програми	Обсяг, байт	Частка, %
Ядро Arduino, стандартні функції runtime	~2500	8.1
Бібліотека SoftPWM (програмний ШІМ на 11 пінах)	~1500	4.8
Бібліотека Wire (I²C для магнітометра)	~1800	5.8
Бібліотека IRremote (декодер NEC)	~3500	11.3
Бібліотека ArduinoJson	~5000	16.1
Драйвери апаратури (car_control, compass, qmc6310, ultrasonic, ir_obstacle, ir_remote, rgb)	~6000	19.4
Клас AiCamera (обмін UART, парсинг команд)	~3500	11.3
Головний скетч (логіка режимів, обробники)	~2000	6.5
Усього зайнято	~25800	83.2
Вільно (резерв на розширення функцій)	~5200	16.8

Flash-пам'ять заповнена приблизно на 80–85 % від максимально доступного обсягу. Залишок у 5–6 КБ є достатнім для подальшого розвитку функціональності

без необхідності переходу на потужніший мікроконтролер. Найбільш «важкі» бібліотеки — ArduinoJson (≈5 КБ) та IRremote (≈3.5 КБ), і саме вони є першими кандидатами на оптимізацію у разі потреби.

5.2.2. Бюджет оперативної пам'яті SRAM

Оперативна пам'ять SRAM є критичним ресурсом, оскільки в ній одночасно розташовуються глобальні змінні, область стека, до якої кладуться локальні змінні функцій і адреси повернення, а також динамічно виділена пам'ять. Орієнтовний розподіл SRAM представлено у табл. 5.3.

Таблиця 5.3 — Орієнтовний бюджет SRAM (2048 байт усього)

Компонент / структура даних	Обсяг, байт	Частка, %
Службові змінні runtime, ядро Arduino	~200	9.8
Кільцеві буфери HardwareSerial (RX 64 + TX 64)	128	6.3
AiCamera::recvBuffer (WS_BUFFER_SIZE = 200)	200	9.8
AiCamera::send_doc (StaticJsonDocument<200>)	200	9.8
Внутрішні буфери IRremote	~80	3.9
Робочі структури SoftPWM (11 пінів × ~12 байт)	~130	6.3
Кільцевий буфер фільтра компаса (30 × int16_t)	60	2.9
Глобальні змінні режимів, прапорці, лічильники	~150	7.3
Локальні змінні функцій, стек викликів	~300	14.6
Усього зайнято	~1450	70.8
Вільно (резерв для глибини стека та локальних змінних)	~600	29.2

Заповнення SRAM на 70–75 % з резервом близько 600 байт є прийнятним для мікроконтролерних застосувань. Відома практична межа, нижче якої середовище Arduino IDE починає видавати попередження «Low memory available, stability problems may occur», становить 75 % — і розроблена прошивка знаходиться у межах безпечного діапазону.

5.2.3. Застосовані техніки економії оперативної пам'яті

Збереження SRAM-бюджету у комфортних межах було б неможливим без застосування ряду характерних для вбудованих систем технік оптимізації [2,5]. Розглянемо основні з них.

1) Використання макроса F() для рядкових літералів. Звичайний літерал виду Serial.println("WebServer started") компілюється так, що рядок копіюється з Flash у SRAM при старті програми. Конструкція Serial.println(F("WebServer started")) залишає рядок у Flash і читає його по символах. Якщо у коді є хоча б 10–15 діагностичних повідомлень середньою довжиною 30 символів, економія SRAM становить 300–450 байт. У реалізації класу AiCamera усі діагностичні повідомлення на DebugSerial оформлені саме через F().

2) Статичні (StaticJsonDocument) замість динамічних JSON-документів. Об'єкт send_doc у класі AiCamera оголошений як StaticJsonDocument<200>, що резервує під JSON-документ фіксовану ділянку SRAM розміром 200 байт. Альтернативний варіант DynamicJsonDocument використовує купу (heap), яка на AVR-мікроконтролерах є потенційно небезпечною — фрагментація купи може непомітно знизити обсяг доступної пам'яті та призвести до збою. Статичний варіант, навпаки, гарантує сталий слід у пам'яті, що піддається аналізу.

3) Мінімізація розрядності типів даних. В усіх драйверах використовуються типи мінімально достатньої розрядності: int8_t для значень потужності моторів (діапазон ±100), int16_t для кутів (±180°), uint8_t для бітових

масок датчиків. Типове для x86-програм використання типу int (4 байти) на AVR-мікроконтролері було б розкішшю.

4) Обмежений розмір буфера обміну UART. Параметр WS_BUFFER_SIZE = 200 байт визначає максимальний розмір повідомлення WebSocket, яке може бути прийняте з ESP32-CAM. Це свідоме обмеження — кожне повідомлення керування поміщається у 200 байт з певним запасом, але буфер не роздувається до невиправдано великих розмірів.

5.2.4. Шляхи подальшої оптимізації

Якщо у майбутньому виникне потреба розширити функціональність прошивки (наприклад, додати алгоритми SLAM, складніші режими автономного руху, інтерфейс з додатковими датчиками), можна застосувати такі додаткові оптимізації, перелічені у порядку зменшення трудомісткості:

1. Зменшити розмір буферів WS_BUFFER_SIZE та StaticJsonDocument до 128 байт. Економія — 144 байти SRAM. Потребує ретельної перевірки, чи вкладаються всі типи повідомлень у скорочений буфер.
2. Зменшити AVERAGE_FILTER_SIZE компаса з 30 до 15 вибірок. Економія — 30 байт SRAM. Точність визначення курсу погіршиться з ±2° до приблизно ±3°.
3. Вимкнути бібліотеку IRremote, якщо ІЧ-пульт не використовується у поточному режимі експлуатації. Економія — приблизно 3.5 КБ Flash та 80 байт SRAM.
4. Перейти на більш потужну плату Arduino Mega 2560 (мікроконтролер ATmega2560 — 256 КБ Flash, 8 КБ SRAM). Цей шлях найпростіший, але потребує заміни апаратури.
5. Радикальний варіант — повністю перенести логіку керування на ESP32-CAM, який має 520 КБ SRAM та частоту до 240 МГц. Arduino UNO у такій схемі залишається лише як низькорівневий драйвер ШІМ. Цей підхід

відкриває можливості для впровадження алгоритмів машинного навчання безпосередньо на борту робота.

Підсумовуючи, можна стверджувати, що ресурси мікроконтролера ATmega328P достатні для реалізації повного функціоналу інформаційної системи дистанційного керування та моніторингу мобільного робота у межах поточних вимог. Бюджет Flash-пам'яті заповнений приблизно на 83 %, SRAM — на 71 %, що залишає прийнятний резерв на роботу стека та подальший розвиток системи.

4.4. Розрахунок енергоспоживання

Для перевірки відповідності системи вимозі автономної роботи протягом 60 хвилин виконаємо оцінкову калькуляцію енергоспоживання усіх компонентів та порівняємо її з ємністю акумулятора.

Орієнтовний бюджет струмів за компонентами при типовому навантаженні (рух з помірною швидкістю, активна передача відео):

— 4 двигуни постійного струму у режимі середньої потужності: 4 × 200 мА = 800 мА (на 7.4 В лінії);

— Arduino UNO у активному режимі: ~50 мА (на 5 В лінії);

— ESP32-CAM у режимі трансляції відео: ~250 мА (на 5 В лінії);

— датчики, ІЧ-приймач, магнітометр, розширювач HC165: сумарно ~50 мА (на 5 В лінії);

— RGB-стрічка WS2812 у середньому режимі підсвічування: ~60 мА.

Сумарний струм споживання на 5 В лінії складає приблизно 410 мА. З урахуванням ККД лінійного стабілізатора (близько 65% при перепаді 7.4 → 5 В), еквівалентний струм споживання з акумулятора з боку 5 В лінії становить приблизно:

$$I\_eq \approx (5\ В \times 410\ мА) / (7.4\ В \times 0.65) = 426\ мА.$$

Загальний струм споживання з акумулятора:

$$I\_total = I\_motors + I\_eq = 800 + 426 = 1226\ мА.$$

При ємності акумулятора 2000 мА·год розрахунковий час автономної роботи:

$$T = 2000 / 1226 \approx 1.63 \text{ год} \approx 98 \text{ хв.}$$

Отримане значення (98 хв) добре узгоджується з паспортним значенням до 90 хв (з урахуванням деякого запасу на втрати у проводах, неідеальність моделі акумулятора при глибокому розряді тощо). У режимах меншої активності (керування ІЧ-пультом без передачі відео, статичне очікування) реальний час автономної роботи може бути значно більшим — до 3-4 годин.

Слід зазначити, що критичним для збереження ресурсу літій-іонного акумулятора є недопущення його глибокого розряду нижче 5.6 В (по 2.8 В на елемент). Підсистема моніторингу системи має сигналізувати оператору про необхідність припинення роботи при досягненні напруги 6.8 В (по 3.4 В на елемент).

РОЗДІЛ 5. ПРОГРАМНА ЧАСТИНА СИСТЕМИ

5.1. Структура програмного забезпечення

Програмне забезпечення інформаційної системи дистанційного керування та моніторингу мобільного робота поділене на три великі рівні, що відповідають трьом фізичним обчислювальним вузлам системи:

1. Прошивка мікроконтролера Arduino UNO R3 — основний обсяг коду, що реалізує драйвери периферії, алгоритми керування рухом і обробки сенсорних даних, обробник команд від оператора та підсистему формування телеметрії.
2. Прошивка модуля ESP32-CAM — реалізує бездротовий зв'язок WiFi, WebSocket-сервер для обміну командами і телеметрією, HTTP MJPEG-сервер для трансляції відео. У межах даної роботи використовується стандартна прошивка з SunFounder Controller, інтерфейс взаємодії з нею описаний у класі AiCamera мікроконтролерної прошивки.
3. Програмне забезпечення робочого місця оператора — статична HTML-сторінка з елементами керування та візуалізації, JavaScript-код взаємодії через WebSocket, а також Python-скрипти для інтелектуальної обробки відеопотоку засобами OpenCV.

Структура мікроконтролерної прошивки організована за принципом модульності: кожному апаратному компоненту відповідає окремий драйвер у вигляді пари файлів .h (заголовний) і .cpp (реалізація). Перелік основних модулів мікроконтролерної прошивки наведено в табл. 5.1.

Таблиця 5.1 — Перелік основних модулів мікроконтролерної прошивки

Модуль	Файли	Призначення
Керування рухом	car_control.h / .cpp	Обчислення швидкостей коліс Mecanum, ПІД-регулятор курсу, field-centric керування
Магнітометр	compass.h / .cpp, qmc6310.h / .cpp	Зчитування з I²C, обчислення курсу, калібрування, фільтрація
Ультразвук	ultrasonic.h / .cpp	Вимірювання відстані до перешкоди, виявлення перешкоди
ІЧ-перешкода	ir_obstacle.h / .cpp	Зчитування стану двох ІЧ-датчиків бічних перешкод
ІЧ-пульт	ir_remote.h / .cpp	Обробка переривань від ІЧ-приймача, декодування коду кнопки
Світлодіоди	rgb.h / .cpp	Керування кольором RGB-стрічки через SoftPWM
Інтерфейс з ESP32	ai_camera.h / .cpp	Двосторонній обмін з ESP32-CAM через UART, формування JSON-телеметрії, парсинг команд
Конфігурація	cmd_code_config.hpp	Визначення кодів режимів, кольорів індикації

Така модульна організація дозволяє: легко замінити апаратний компонент без зміни решти коду; повторно використовувати готові драйвери у нових проєктах; полегшити модульне тестування; підвищити читабельність і супроводжуваність коду.

5.2. Драйвер керування рухом (car_control)

Драйвер керування рухом (модуль car_control) є центральним компонентом мікроконтролерної прошивки. Він реалізує перетворення керуючих команд оператора у конкретні значення ШІМ-сигналів на восьми пінах драйверів двигунів.

5.2.1. Інтерфейс модуля

Заголовний файл car_control.h визначає перелік експортованих функцій та основних констант. Розташування двигунів у конструкції робота умовно показано в коментарі заголовного файлу:

```
/*
 *  [0]--|||--[1]
 *   |         |
 *   |         |
 *   |         |
 *   |         |
 *  [3]-------[2]
 */


/** Set the pins for the motors */
#define MOTOR_PINS       (uint8_t[8]){3, 4, 5, 6, A3, A2, A1, A0}
/** Set the positive and negative directions for the motors */
#define MOTOR_DIRECTIONS (uint8_t[4]){1, 0, 0, 1}


/** define motors default speed */
#define CAR_DEFAULT_POWER 80
```

Як видно з лістингу, мотор M0 знаходиться у передньому лівому куті робота, M1 — у передньому правому, M2 — у задньому правому, M3 — у задньому лівому. На кожен мотор виділяється два піни (для двох входів H-моста), що дозволяє керувати як швидкістю обертання (ШІМ), так і напрямком.

Перелік основних експортованих функцій модуля:

— carBegin() — ініціалізація драйверів двигунів та магнітометра, скидання нульового курсу;

— carForward(power), carBackward(power), carLeft(power), carRight(power), carTurnLeft(power), carTurnRight(power) і подібні — спрощені інтерфейси для базових напрямків руху;

— carMove(angle, power, rot, drift) — узагальнений інтерфейс руху з довільним вектором швидкості;

— carMoveFieldCentric(angle, power, heading, drift, angFlag) — рух у режимі field-centric з ПІД-регулятором курсу;

— carResetHeading() — фіксація поточного курсу як нульового;

— carStop() — повна зупинка усіх двигунів.

5.2.2. Реалізація обчислення Mecanum-кінематики

Серцевиною модуля є функція carMove(), що реалізує обчислення швидкостей чотирьох коліс за заданим вектором руху (теоретичні основи представлено в розділі 2). Розглянемо її реалізацію:

```
void carMove(int16_t angle, int8_t power, int8_t rot, bool drift) {
  int8_t power_0, power_1, power_2, power_3;
  float ratio;
  // Make forward 0
  angle += 90;
  // Offset angle as 0 to the front
  float rad = angle * PI / 180;


  ratio = 0.5;
  power /= sqrt(2);
  power = power * (1-ratio);


  // Calculate 4 wheel
  if (drift) {
    power_0 = (power * sin(rad) - power * cos(rad)) ;
    power_1 = (power * sin(rad) + power * cos(rad)) ;
    power_2 = (power * sin(rad) - power * cos(rad)) + rot * ratio * 2;
    power_3 = (power * sin(rad) + power * cos(rad)) - rot * ratio * 2;
  } else {
    power_0 = (power * sin(rad) - power * cos(rad)) - rot * ratio;
```

```
      power_1 = (power * sin(rad) + power * cos(rad)) + rot * ratio;
      power_2 = (power * sin(rad) - power * cos(rad)) + rot * ratio;
      power_3 = (power * sin(rad) + power * cos(rad)) - rot * ratio;
    }


    carSetMotors(power_0, power_1, power_2, power_3);
  }
```

Розглянемо логіку обчислень детальніше. Вхідний параметр angle задає бажаний напрямок руху у градусах: 0° — уперед, +90° — праворуч, +180° — назад, -90° — ліворуч. Зсув angle += 90 виконує перетворення цієї системи координат до тригонометричної (де 0° — праворуч, 90° — уперед), у якій зручніше виконувати подальші розрахунки через стандартні функції sin() і cos().

Параметр ratio = 0.5 задає співвідношення між трансляційною та обертовою компонентами руху: половина «потужності» виділяється на трансляцію, друга половина може бути використана для обертання. Поділ power на sqrt(2) компенсує те, що при діагональному русі сумарна швидкість колеса є геометричною сумою двох компонент.

У режимі drift = false (звичайний рух) усі чотири мотори отримують однакову поправку від обертання rot. У режимі drift = true (заносу) поправка отримують лише задні мотори M2 і M3, що дозволяє роботу здійснювати характерний для дрифту рух.

Розрахунок швидкостей коліс відповідає теоретичним формулам, виведеним у розділі 2, з тією лише різницею, що для практичної реалізації використана дещо спрощена нормалізація.

5.2.3. Низькорівневе встановлення швидкостей моторів

Функція carSetMotors() переводить розрахункові значення power_0..power_3 (у діапазоні −100..+100) у конкретні ШІМ-сигнали на пінах. Окремої уваги заслуговує спеціальна обробка пускового режиму:

```
void carSetMotors(int8_t power0, int8_t power1, int8_t power2, int8_t power3) {
  bool dir[4];
  int8_t power[4] = {power0, power1, power2, power3};
  int8_t newPower[4];

  for (uint8_t i = 0; i < 4; i++) {
    dir[i] = power[i] > 0;

    if (MOTOR_DIRECTIONS[i]) dir[i] = !dir[i];

    if (power[i] == 0) {
      newPower[i] = 0;
    } else {
      newPower[i] = map(abs(power[i]), 0, 100, MOTOR_POWER_MIN, 255);
    }

    if (newPower[i] != 0 && newPower[i] < MOTOR_START_POWER) {
      SoftPWMSet(MOTOR_PINS[i*2], dir[i] * MOTOR_START_POWER);
      SoftPWMSet(MOTOR_PINS[i*2+1], !dir[i] * MOTOR_START_POWER);
      delayMicroseconds(200);
    }
    SoftPWMSet(MOTOR_PINS[i*2], dir[i] * newPower[i]);
    SoftPWMSet(MOTOR_PINS[i*2+1], !dir[i] * newPower[i]);
  }
}
```

Логіка пускового режиму базується на фізичній особливості двигунів постійного струму з редуктором: при низьких напругах живлення (відповідних малим значенням ШІМ) сила інерції тертя у редукторі може перевищувати силу обертання валу, і двигун не запускається. Для подолання цієї проблеми застосовано двофазний пуск:

1. Спочатку на 200 мкс на двигун подається повна стартова потужність (MOTOR_START_POWER = 100, при максимумі 255), що гарантовано долає інерцію.

2. Потім встановлюється цільове значення потужності, перераховане через функцію map() з діапазону 0..100 у діапазон MOTOR_POWER_MIN..255 (де MOTOR_POWER_MIN = 50).

Така схема забезпечує надійний пуск моторів на низьких швидкостях і плавне регулювання у робочому діапазоні.

5.2.4. ПІД-регулятор та field-centric керування

Найскладніша частина драйвера — функція carMoveFieldCentric(), що реалізує керування у «польово-центричному» режимі. У цьому режимі напрямки руху задаються відносно нерухомої системи координат (по суті — оператора), а не корпусу робота. Це особливо корисно, коли робот рухається у складній траєкторії з поворотами, оскільки натиснута кнопка «вперед» завжди означає рух у тому самому напрямку, незалежно від поточної орієнтації корпусу.

```
/* Set the pid parameters */
#define KP (float)0.8
#define KI (float)0.0
#define KD (float)20.0


int32_t _lastError = 0;
int32_t errorIntegral = 0;
int16_t originHeading;


void carMoveFieldCentric(int16_t angle, int8_t power,
                         int16_t heading, bool drift, bool angFlag) {
  int16_t currentHeading = 0;
  int32_t error = 0;
  int32_t offset = 0;
  int8_t rot = 0;


  currentHeading = compassReadAngle();
  error = currentHeading - originHeading - heading;


  // convert -360 to 360 to -180 to 180
  while (error > 180) { error -= 360; }
```

```
    while (error < -180) { error += 360; }

    if (error > 1 || error < -1) {
      offset += KP * error + KI * errorIntegral + KD * (error - _lastError);
      rot += max(-100, min(100, offset));
      errorIntegral += error;
      _lastError = error;
    }

    if (angFlag == false && (angle != 0 || drift == true)) {
      angle = angle - currentHeading + originHeading;
    }

    carMove(angle, power, rot, drift);
  }
```

Класичне рівняння ПІД-регулятора має вигляд:

$$u(t) = K\_p \cdot e(t) + K\_i \cdot \int e(\tau)d\tau + K\_d \cdot de(t)/dt,$$

де e(t) — поточна похибка регулювання, u(t) — керуючий вплив, K_p, K_i, K_d — коефіцієнти пропорційної, інтегральної та диференційної складових. У дискретній реалізації у мікроконтролерному коді інтеграл замінюється сумою (errorIntegral), а похідна — різницею (error - _lastError).

У досліджуваному коді коефіцієнти підібрані експериментальним шляхом: K_p = 0.8, K_i = 0.0 (інтегральна складова відключена для уникнення накопичення помилки під час тривалих маневрів), K_d = 20.0 (висока диференційна складова — для гасіння коливань корпусу). Похибка регулювання e обчислюється як різниця між поточним курсом і цільовим:

$$e = currentHeading - originHeading - heading.$$

Нормалізація похибки у діапазон [-180°, +180°] (цикл while) виключає ситуацію, коли робот, маючи зайвий оберт у траєкторії, обертається на 359° замість того, щоб повернутися на 1° у протилежному напрямку. Поріг |e| > 1° потрібен для запобігання дрібним коливанням навколо встановленого курсу при наявності шумів магнітометра.

Розраховане керуюче значення rot (обмежене діапазоном [-100, +100]) передається у функцію carMove() як параметр повороту. Кутова поправка angle = angle - currentHeading + originHeading реалізує перетворення координат з абсолютної системи (заданої оператором) до системи робота, що фактично і реалізує field-centric поведінку.

5.3. Драйвер магнітометра та ПІД-регулятор

Модуль compass виконує функції обробки даних з магнітометра QMC6310 та обчислення поточного курсу робота. Модуль qmc6310 містить низькорівневі функції роботи з мікросхемою магнітометра через I²C, а compass — функції фільтрації даних, калібрування та обчислення курсу.

5.3.1. Інтерфейс модуля compass

```
/** Set the number of samples for average filtering, default 30 */
#define AVERAGE_FILTER_SIZE 30

/** Set the EEPROM start address for calibration storage, default 0 */
#define EEPROM_CALIBRATION_ADDRESS 0

/** Calibration stabilisation time, millisecond, default 3000 */
#define CALIBRATION_TIME 3000

/** Whether to turn on the average filter, default true */
#define AVERAGE_FILTER true

void compassBegin();
int16_t compassReadAngle();
int16_t compassGetAzimuth();
void compassCalibrateStart();
bool compassCalibrateLoop();
bool compassCalibrateDone();
```

Параметр AVERAGE_FILTER_SIZE = 30 задає розмір вікна ковзного середнього, що використовується для зниження шуму вимірювань магнітометра. Параметр CALIBRATION_TIME визначає мінімальний час обертання робота при калібруванні (3 секунди), за який мають бути зібрані екстремальні значення компонент магнітного поля по всіх осях.

5.3.2. Калібрування магнітометра

Калібрування магнітометра — обов'язкова процедура перед першим використанням, оскільки магнітне поле у місці експлуатації спотворюється сталевими конструкціями, провідниками зі струмом, тощо. Метою калібрування є знаходження зміщення (offset) та масштабу (scale) для кожної з трьох осей таким чином, щоб обертання робота навколо вертикальної осі при незмінному горизонтальному магнітному полі приводило до кругової траєкторії показань у площині XY з центром у нулі.

Алгоритм калібрування реалізує метод найменших квадратів, узгоджуючи показання магнітометра з еліпсом і обчислюючи параметри перетворення, що приводить його до кола. Знайдені калібрувальні значення зберігаються у EEPROM мікроконтролера (за адресою EEPROM_CALIBRATION_ADDRESS) і завантажуються при кожному запуску, тому повторне калібрування потрібне лише при значній зміні умов експлуатації.

5.3.3. Обчислення курсу

Курс (азимут) обчислюється з горизонтальних компонент магнітного поля за формулою:

$$\theta = atan2(My, Mx) \cdot 180/\pi.$$

Функція compassReadAngle() здійснює обчислення з усередненням по AVERAGE_FILTER_SIZE послідовних вимірювань, що знижує дисперсію оцінки курсу за рахунок певної інерційності реакції на повороти. Експериментально

визначено, що для досліджуваного робота 30 вимірювань забезпечують прийнятний компроміс між точністю (~±2°) і часом відгуку (~93 мс на повний цикл).

5.4. Драйвери сенсорних модулів

5.4.1. Драйвер ультразвукового датчика

Драйвер модуля ultrasonic реалізує роботу з датчиком HC-SR04. Особливістю використання у комплексі є те, що для економії пінів TRIG і ECHO підключені до одного й того ж піна D10, і драйвер мусить програмно перемикати напрямок виводу:

```
float ultrasonicRead() {
  delay(4); // A delay is required between consecutive readings


  pinMode(ULTRASONIC_TRIG_PIN, OUTPUT);
  digitalWrite(ULTRASONIC_TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(ULTRASONIC_TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(ULTRASONIC_TRIG_PIN, LOW);
  pinMode(ULTRASONIC_ECHO_PIN, INPUT);


  float duration = pulseIn(ULTRASONIC_ECHO_PIN, HIGH);
  float distance = duration * 0.017;


  if (distance > MAX_DISTANCE) {
    return -1;
  }
  return distance;
}
```

Алгоритм роботи функції такий:

1. Перемикання піна у режим OUTPUT, формування TRIG-імпульсу тривалістю 10 мкс (попередньо встановлюється низький рівень для очищення стану).
2. Перемикання піна у режим INPUT для прийому ECHO-сигналу.

3. Виклик функції pulseIn(), що повертає тривалість імпульсу ECHO у мікросекундах.
4. Обчислення відстані за формулою distance = duration · 0.017, що базується на швидкості звуку в повітрі (≈340 м/с): за час duration ультразвук долає шлях туди-назад, тому реальна відстань — duration · 340 м/с / 2 = duration · 0.017 см/мкс.

Затримка delay(4) перед кожним вимірюванням потрібна для уникнення інтерференції з попереднім ехо. Якщо відстань перевищує MAX_DISTANCE = 2000 см (датчик не отримав ехо), повертається значення -1.

На основі функції ultrasonicRead() побудовані допоміжні функції для перевірки наявності перешкоди:

```
bool ultrasonicIsObstacle() {
  return ultrasonicRead() < ULTRASONIC_AVOIDANCE_THRESHOLD;
}


bool ultrasonicIsClear() {
  float distance = ultrasonicRead();
  if (distance > ULTRASONIC_AVOIDANCE_THRESHOLD || distance < 0) {
    return true;
  } else {
    return false;
  }
}
```

Поріг ULTRASONIC_AVOIDANCE_THRESHOLD = 20 см задає мінімальну безпечну відстань. Зауважимо, що функція ultrasonicIsClear() трактує негативне значення (відсутність ехо) як «вільно», що є логічно правильним у більшості випадків.

5.4.2. Драйвер ІЧ-датчиків перешкод

Драйвер ir_obstacle забезпечує читання стану двох інфрачервоних модулів уникнення перешкод, встановлених з лівого та правого боку робота. Через

обмежену кількість вільних цифрових пінів мікроконтролера ATmega328P підключення обох датчиків виконане не безпосередньо до GPIO-портів Arduino, а через зсувний регістр HC165, що виконує функцію розширювача цифрових входів. Це апаратне рішення дозволяє економити пори мікроконтролера: 8-бітний стан регістра зчитується по послідовному інтерфейсу з використанням лише трьох ліній — LATCH, CLK та DATA. Низькорівнева робота з регістром винесена в окремий модуль hc165, що повертає 16-бітне слово зі станом усіх підключених до нього сигналів.

Власне функція читання стану ІЧ-датчиків є вкрай простою — вона лише виділяє молодший байт зі слова, отриманого з HC165:

```
cppbyte irObstacleRead() {
  return hc165Read() & 0xFF;
}
```

Два молодших біти повернутого значення відповідають стану лівого (біт 0) та правого (біт 1) ІЧ-датчиків. Логічний нуль у відповідному біті означає наявність перешкоди на відстані менше встановленого порога, логічна одиниця — її відсутність. Така інверсна логіка є типовою для модулів з NPN-виходом і відкритим колектором: вихідний транзистор відкривається при спрацюванні датчика, прив'язуючи лінію до загального проводу.

Декодування бітів виконується на стороні викликаючого коду:

```
cppbyte ir = irObstacleRead();
bool obstacleLeft  = !(ir & 0x01);  // інверсія: 0 → є перешкода
bool obstacleRight = !(ir & 0x02);
```

Простота реалізації драйвера компенсується великим прикладним значенням: бінарні бічні датчики дають практично миттєвий (без обчислень і затримок на ехо) сигнал про наявність перешкоди обабіч робота. У парі з ультразвуковим датчиком, що контролює напрямок руху попереду, вони забезпечують повне «коло безпеки» навколо корпусу. Це особливо важливо при русі у вузьких коридорах або у режимі автоматичного об'їзду перешкод, де реакція бічних датчиків використовується для негайного коригування траєкторії.

5.4.3. Драйвер ІЧ-пульта

Драйвер ir_remote використовує сторонню бібліотеку IRremote для декодування протоколу NEC. Перелік кодів кнопок стандартного 21-кнопкового ІЧ-пульта попередньо визначено у заголовному файлі:

```
#define IR_KEY_POWER 0x45
#define IR_KEY_MODE  0x46
#define IR_KEY_MUTE  0x47

#define IR_KEY_PLAY_PAUSE 0x44
#define IR_KEY_BACKWARD   0x40
#define IR_KEY_FORWARD    0x43

#define IR_KEY_EQ    0x07
#define IR_KEY_MINUS 0x15
#define IR_KEY_PLUS  0x09

#define IR_KEY_0     0x16
#define IR_KEY_CYCLE 0x19

void irBegin();
uint8_t irRead();
```

Функція irRead() повертає код останньої прийнятої кнопки або 0 (IR_KEY_ERROR), якщо за час між викликами жодної кнопки не було натиснуто. Декодування виконується у фоновому режимі через обробник переривання, прив'язаний до піна D2.

5.4.4. Драйвер RGB-стрічки

Драйвер rgb забезпечує керування трьома каналами RGB-стрічки через бібліотеку SoftPWM. Перевагою програмного ШІМ є те, що він може працювати

на будь-яких цифрових пінах, на відміну від апаратного, обмеженого спеціальними пінами AVR-мікроконтролера:

```
void rgbWrite(uint8_t r, uint8_t g, uint8_t b) {
  // calibrate brightness
  r = int(r * R_OFFSET);
  g = int(g * G_OFFSET);
  b = int(b * B_OFFSET);
  // COMMON_ANODE reverse
  #if COMMON_ANODE
    r = 255 - r;
    g = 255 - g;
    b = 255 - b;
  #endif
  // set voltage
  SoftPWMSet(RGB_PINS[0], r);
  SoftPWMSet(RGB_PINS[1], g);
  SoftPWMSet(RGB_PINS[2], b);
}
```

Калібрувальні коефіцієнти R_OFFSET = 1.0, G_OFFSET = 0.16, B_OFFSET = 0.30 компенсують різну яскравість світіння кристалів різних кольорів — без них «білий» RGB(255,255,255) візуально виглядав би холодно-зеленим. Підтримка спільного анода/катода реалізована умовною компіляцією за макросом COMMON_ANODE.

Перевантажена версія rgbWrite(uint32_t color) приймає колір у 24-бітному форматі (0xRRGGBB) і виконує розбиття на компоненти автоматично, що зручно для використання заздалегідь визначених іменованих кольорів (RED, GREEN, BLUE тощо).

5.5. Інтерфейс з модулем ESP32-CAM (AiCamera)

Інтерфейс взаємодії мікроконтролера Arduino з модулем ESP32-CAM реалізований у вигляді класу AiCamera. Цей клас інкапсулює низькорівневу логіку

обміну текстовими повідомленнями через UART і надає високорівневі методи для зчитування команд оператора та формування телеметрії.

5.5.1. Архітектура взаємодії Arduino — ESP32-CAM

Зв'язок між Arduino і ESP32-CAM побудований за принципом «командна оболонка»: Arduino надсилає текстові команди із префіксом WS+ (для WebSocket-повідомлень оператора) або SC (для системних команд), ESP32-CAM виконує запитану дію і повертає відповідь, що містить [OK] або [ERR] разом з результатом. Ця архітектура зручна, оскільки дозволяє оновлювати прошивку ESP32-CAM незалежно від мікроконтролера Arduino, не змінюючи протокол взаємодії.

Перелік основних команд інтерфейсу:

— RESET — програмний перезапуск ESP32-CAM, повертає версію firmware;

— NAME, TYPE — встановлення імені та типу пристрою для ідентифікації у мережі;

— SSID, PSK — встановлення параметрів WiFi-мережі;

— MODE — встановлення режиму WiFi (STA, AP, або вимкнено);

— PORT — встановлення порту WebSocket-сервера;

— START — запуск веб-сервера; повертає IP-адресу;

— WS+ — пересилання JSON-повідомлення WebSocket-клієнту;

— LAMP — керування вбудованим світлодіодом-ліхтариком.

5.5.2. Ініціалізація з'єднання

Метод begin() класу AiCamera виконує повну ініціалізацію зв'язку з ESP32-CAM і налаштування його параметрів WiFi:

```
void AiCamera::begin(const char *ssid, const char *password,
                     const char *wifiMode, const char *wsPort) {
  char ip[25];
  char version[25];

  set_command_timeout(3000);
```

```
    this->get("RESET", version);
    DebugSerial.print(F("ESP32 firmware version "));
    DebugSerial.println(version);

    set_command_timeout(1000);
    this->set("NAME", name);
    this->set("TYPE", type);
    this->set("SSID", ssid);
    this->set("PSK", password);
    this->set("MODE", wifiMode);
    this->set("PORT", wsPort);

    set_command_timeout(5000);
    this->get("START", ip);
    delay(20);
    DebugSerial.print(F("WebServer started on ws://"));
    DebugSerial.print(ip);
    DebugSerial.print(F(":"));
    DebugSerial.println(wsPort);

    set_command_timeout(SERIAL_TIMEOUT);
}
```

Послідовність дій така: ESP32 перезапускається (RESET) для виключення впливу попередніх налаштувань, встановлюються ім'я і тип пристрою, потім параметри WiFi-з'єднання, і нарешті запускається веб-сервер. Команди із збільшеним таймаутом (5000 мс на START) дають часу ESP32 виконати тривалі операції (підключення до WiFi, отримання IP).

5.5.3. Головний цикл та обробка повідомлень

У головному циклі програми періодично викликаються методи loop() і sendData(). Метод loop() приймає вхідні повідомлення від WebSocket-клієнтів і викликає зареєстрований обробник:

```
void AiCamera::loop() {
  this->readInto(this->recvBuffer);


  if (strlen(this->recvBuffer) != 0) {
    if (IsStartWith(this->recvBuffer, WS_CONNECT)) {
      ws_connected = true;
      // ... handle new connection
    } else if (IsStartWith(this->recvBuffer, WS_DISCONNECT)) {
      ws_connected = false;
      // ... handle disconnect
    } else {
      char *split = strchr(this->recvBuffer, '+');
      if (split != NULL) {
        *split = '\0';
        if (__on_receive__ != NULL) {
          __on_receive__(this->recvBuffer, split + 1);
        }
      }
    }
  }
}
```

Метод sendData() надсилає накопичене у внутрішньому JSON-документі (send_doc) телеметричне повідомлення:

```
void AiCamera::sendData() {
  if (!ws_connected) return;
  if (millis() - ws_send_time > ws_send_interval) {
    String output;
    serializeJson(send_doc, output);
    DateSerial.print(F(WS_HEADER));
    DateSerial.println(output);
    ws_send_time = millis();
  }
}
```

Інтервал відправки ws_send_interval = 60 мс (≈17 повідомлень/с) є компромісом: занадто часті повідомлення створюють надлишкове навантаження на канал зв'язку, рідкі — погіршують відчуття «живої» візуалізації.

5.5.4. Високорівневі методи отримання команд

Клас AiCamera надає набір методів-«гетерів» для отримання значень елементів керування веб-інтерфейсу:

```
int16_t getSlider(uint8_t region);     // Слайдер 0..100
bool    getButton(uint8_t region);     // Кнопка натиснута
bool    getSwitch(uint8_t region);     // Перемикач
int16_t getJoystick(uint8_t region, uint8_t axis); // X/Y/angle/radius
uint8_t getDPad(uint8_t region);       // D-Pad напрямок
int16_t getThrottle(uint8_t region);   // Газ ±100
void    getSpeech(uint8_t region, char *result);   // Голосовий текст
```

У кожен з методів передається ідентифікатор «регіону» інтерфейсу (REGION_A..REGION_Z) — це конкретна позиція елементу на формі веб-інтерфейсу. Така схема дозволяє оператору налаштовувати інтерфейс гнучко, переміщуючи елементи між позиціями без перепрошивки робота.

5.5.5. Методи формування телеметрії

Аналогічні «сетери» забезпечують встановлення значень для візуалізації у дашборді оператора:

```
void setMeter(uint8_t region, double value);
void setRadar(uint8_t region, int16_t angle, double distance);
void setGreyscale(uint8_t region, uint16_t v1, uint16_t v2, uint16_t v3);
void setValue(uint8_t region, double value);
```

Виклик setMeter() оновлює значення стрілкового індикатора (зокрема, використовується для індикатора заряду батареї), setRadar() — точку на колі радару (для відображення поточної відстані до перешкоди), setValue() — для довільних чисельних значень. Усі ці методи записують відповідні дані в JSON-документ send_doc, який потім періодично надсилається методом sendData().

5.6. Веб-інтерфейс оператора

Веб-інтерфейс оператора реалізований у вигляді однієї HTML-сторінки index.html, що завантажується у будь-який сучасний веб-браузер з ПК або мобільного пристрою. Сторінка об'єднує два основні функціональні блоки: блок

керування з кнопками руху і блок відображення з відеопотоком та індикаторами телеметрії.

5.6.1. HTML-структура сторінки

Спрощений макет HTML-сторінки веб-інтерфейсу:

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Zeus Car: дистанційне керування</title>
  <style>
    body { font-family: Arial; margin: 0; padding: 10px;
           background: #1e1e1e; color: #f0f0f0; }
    .container { display: grid;
                 grid-template-columns: 2fr 1fr; gap: 16px; }
    .video { background: #000; border-radius: 8px; }
    .control-pad { display: grid;
                   grid-template-columns: repeat(3, 80px);
                   grid-template-rows: repeat(3, 80px); gap: 8px; }
    button { font-size: 20px; border-radius: 8px;
             border: none; cursor: pointer; }
    .telemetry { background: #2a2a2a; padding: 12px;
                 border-radius: 8px; }
    .indicator { display: flex; justify-content: space-between;
                 margin: 6px 0; }
  </style>
</head>
<body>
  <h1>Zeus Car — пульт оператора</h1>
  <div class="container">
    <!-- Колонка 1: відеопотік + кнопки -->
    <div>
      <img class="video"
           src="http://192.168.1.50:81/stream"
           width="640" height="480">
      <div class="control-pad">
        <div></div>
        <button onclick="sendCmd('forward')">▲</button>
```

```
        <div></div>
        <button onclick="sendCmd('left')">◀</button>
        <button onclick="sendCmd('stop')">■</button>
        <button onclick="sendCmd('right')">▶</button>
        <div></div>
        <button onclick="sendCmd('backward')">▼</button>
        <div></div>
      </div>
      <label>Швидкість: <input type="range" id="speed"
             min="0" max="100" value="60"></label>
    </div>
    <!-- Колонка 2: телеметрія -->
    <div class="telemetry">
      <h2>Телеметрія</h2>
      <div class="indicator">
        <span>Батарея:</span> <span id="bat">--.- В</span>
      </div>
      <div class="indicator">
        <span>Відстань попереду:</span> <span id="us">--- см</span>
      </div>
      <div class="indicator">
        <span>Курс:</span> <span id="hd">---°</span>
      </div>
      <div class="indicator">
        <span>Режим:</span> <span id="mode">---</span>
      </div>
      <div class="indicator">
        <span>Час роботи:</span> <span id="uptime">00:00</span>
      </div>
      <canvas id="radar" width="240" height="240"></canvas>
    </div>
  </div>
  <script src="control.js"></script>
</body>
</html>
```

Сторінка побудована на основі грід-розкладки: ліва колонка містить відеопотік та елементи керування, права — телеметричні індикатори. Відеопотік вбудовується як звичайний тег <img>, у якого src вказує на URL MJPEG-стріму. Браузер автоматично оновлює зображення по мірі надходження нових кадрів.

5.6.2. JavaScript-логіка взаємодії

Файл control.js реалізує клієнтську логіку взаємодії з роботом. Основа — WebSocket-з'єднання, через яке надсилаються команди керування і приймається телеметрія:

```
const WS_URL = 'ws://192.168.1.50:8765';
let ws = null;
let reconnectTimer = null;

function connect() {
  ws = new WebSocket(WS_URL);

  ws.onopen = () => {
    console.log('WebSocket connected');
    document.body.style.borderTop = '4px solid #0f0';
  };

  ws.onmessage = (event) => {
    try {
      const data = JSON.parse(event.data);
      updateTelemetry(data);
    } catch (e) {
      console.error('Bad telemetry packet', e);
    }
  };

  ws.onclose = () => {
    document.body.style.borderTop = '4px solid #f00';
    reconnectTimer = setTimeout(connect, 1000);
  };

  ws.onerror = (e) => console.error('WS error', e);
}

function sendCmd(cmd) {
  if (!ws || ws.readyState !== WebSocket.OPEN) return;
  const speed = document.getElementById('speed').value;
```

```
  ws.send(JSON.stringify({ cmd: cmd, power: parseInt(speed) }));
}

function updateTelemetry(d) {
  if ('bat' in d)  document.getElementById('bat').textContent = d.bat.toFixed(1) + '
B';
  if ('us' in d)   document.getElementById('us').textContent  = Math.round(d.us) + '
см';
  if ('hd' in d)   document.getElementById('hd').textContent  = d.hd + '°';
  if ('mode' in d) document.getElementById('mode').textContent= MODE_NAMES[d.mode];
  if ('uptime' in d) {
    const m = Math.floor(d.uptime / 60), s = d.uptime % 60;
    document.getElementById('uptime').textContent =
      String(m).padStart(2,'0') + ':' + String(s).padStart(2,'0');
  }
  if ('us' in d && 'hd' in d) drawRadar(d.hd, d.us);
}

// Радар відстаней (Canvas)
function drawRadar(angle, distance) {
  const cv = document.getElementById('radar');
  const ctx = cv.getContext('2d');
  ctx.clearRect(0, 0, cv.width, cv.height);
  // концентричні кола (50, 100, 150, 200 см)
  ctx.strokeStyle = '#444';
  for (let r = 30; r <= 120; r += 30) {
    ctx.beginPath();
    ctx.arc(120, 120, r, 0, Math.PI*2);
    ctx.stroke();
  }
  // позначка перешкоди
  if (distance > 0 && distance < 200) {
    const r = distance * 0.6;  // масштаб
    const rad = (angle - 90) * Math.PI / 180;
    const x = 120 + r * Math.cos(rad);
    const y = 120 + r * Math.sin(rad);
    ctx.fillStyle = '#f00';
    ctx.beginPath();
    ctx.arc(x, y, 5, 0, Math.PI*2);
    ctx.fill();
  }
```

```
}
connect();
```

Ключові особливості реалізації:

1. Автоматичне перепідключення — при втраті WebSocket-з'єднання таймер reconnectTimer ініціює повторне підключення через 1 секунду. Це робить систему стійкою до короткочасних збоїв WiFi.
2. Візуальна індикація стану зв'язку — кольорова смуга у верхній частині сторінки: зелена при з'єднанні, червона при втраті.
3. Парсинг JSON-телеметрії та оновлення відповідних DOM-елементів за ключами повідомлення.
4. Реалізація графічного радара відстаней через Canvas API. Центр радара представляє позицію робота, концентричні кола — відстані 50, 100, 150, 200 см. Червона точка показує поточну позицію перешкоди.
5. Команди керування передаються як JSON-документи {cmd: ..., power: ...} — текстовий формат, простий для відлагодження.

5.7. Підсистема моніторингу та прогнозної діагностики

Підсистема моніторингу реалізована у двох частинах: вбудована частина на стороні мікроконтролера (формує і надсилає телеметричні повідомлення) і клієнтська частина на стороні оператора (аналізує отримані дані, виявляє аномалії, сповіщає оператора).

5.7.1. Вбудована частина: формування телеметрії

Вбудована частина моніторингу розташована у головному циклі loop() основного скетча програми мікроконтролера. На кожній ітерації циклу зчитуються поточні значення усіх контрольованих параметрів, формується JSON-документ і ставиться у чергу на відправлення:

```
#include "ai_camera.h"
#include "car_control.h"
#include "compass.h"
#include "ultrasonic.h"
```

```
#include "ir_obstacle.h"
#include "ir_remote.h"
#include "rgb.h"
#include "cmd_code_config.hpp"

AiCamera aiCam("ZeusCar", "Zeus_Car");

const char *SSID = "YourWiFi";
const char *PSK  = "YourPassword";
const char *MODE = WIFI_MODE_STA;
const char *PORT = "8765";

// ----------- ПОРОГИ МОНІТОРИНГУ -----------
#define BATTERY_LOW_WARN_V   6.8
#define BATTERY_LOW_CRIT_V   6.2
#define DISTANCE_DANGER_CM   15

// ----------- ЗМІННІ СТАНУ -----------
uint32_t monitorTime = 0;
const uint32_t MONITOR_INTERVAL = 50;  // мс
uint8_t  warningLevel = 0;             // 0=ok, 1=warn, 2=crit

// ----------- ВИМІР НАПРУГИ АКУМУЛЯТОРА -----------
// На платі Shield є резистивний дільник 2:1, що приводить
// напругу 7.4 В у діапазон 0-3.7 В, прийнятний для АЦП Arduino
#define BATTERY_PIN A6
#define BATTERY_DIV 2.0    // коефіцієнт дільника
float readBatteryVoltage() {
  const uint8_t N = 8;
  uint32_t acc = 0;
  for (uint8_t i = 0; i < N; i++) acc += analogRead(BATTERY_PIN);
  float v = (acc / (float)N) * (5.0 / 1023.0) * BATTERY_DIV;
  return v;
}

void setup() {
  Serial.begin(115200);
  carBegin();
  gsBegin();
```

```
  irObstacleBegin();
  irBegin();
  rgbBegin();
  rgbWrite(MODE_NONE_COLOR);
  aiCam.begin(SSID, PSK, MODE, PORT);
  aiCam.setOnReceived(onReceive);
  monitorTime = millis();
}

void loop() {
  aiCam.loop();
  irControl();          // обробка кнопок ІЧ-пульта
  modeHandler();        // виконання активного режиму

  // --- ПЕРІОДИЧНИЙ МОНІТОРИНГ ---
  if (millis() - monitorTime > MONITOR_INTERVAL) {
    monitorTime = millis();
    updateMonitoring();
  }

  aiCam.sendData();
}
```

Окремої уваги заслуговує функція updateMonitoring(), яка є серцем підсистеми моніторингу:

```
void updateMonitoring() {
  // 1) Зчитування параметрів
  float batV     = readBatteryVoltage();
  float distance = ultrasonicRead();
  int16_t heading= compassReadAngle();
  byte gsBits    = gsRead();
  byte irBits    = irObstacleRead();
  uint32_t up_s  = millis() / 1000UL;

  // 2) Заповнення JSON-документа телеметрії
  aiCam.send_doc.clear();
  aiCam.send_doc["bat"]    = batV;
  aiCam.send_doc["us"]     = distance;
  aiCam.send_doc["hd"]     = heading;
  aiCam.send_doc["gs"]     = gsBits;
```

```
  aiCam.send_doc["ir_l"]   = (irBits & 0x01) ? 0 : 1; // 0/1
  aiCam.send_doc["ir_r"]   = (irBits & 0x02) ? 0 : 1;
  aiCam.send_doc["mode"]   = currentMode;
  aiCam.send_doc["uptime"] = up_s;


  // 3) Виявлення аномалій (прогнозна діагностика)
  uint8_t newLevel = 0;
  if (batV < BATTERY_LOW_CRIT_V) newLevel = 2;
  else if (batV < BATTERY_LOW_WARN_V) newLevel = max(newLevel, (uint8_t)1);


  if (distance > 0 && distance < DISTANCE_DANGER_CM) {
    newLevel = max(newLevel, (uint8_t)1);
    aiCam.send_doc["warn"] = "obstacle_close";
  }


  // 4) Кольорова індикація стану
  if (newLevel != warningLevel) {
    warningLevel = newLevel;
    if (warningLevel == 2)      rgbWrite(ERROR_COLOR);
    else if (warningLevel == 1) rgbWrite(WARN_COLOR);
    // інакше індикація відповідає поточному режиму
    // — встановлюється функцією setModeColor()
    else setModeColor();
  }
  aiCam.send_doc["lvl"] = warningLevel;
}
```

Підсистема моніторингу циклічно зчитує всі контрольовані параметри:

— напругу акумулятора через АЦП Arduino (через резистивний дільник 2:1);

— відстань до перешкоди через ультразвуковий датчик;

— курс робота через магнітометр;

— стан двох ІЧ-датчиків бічних перешкод;

— поточний режим роботи;

— час безперервної роботи у секундах.

Усі ці параметри упаковуються у JSON-документ send_doc класу AiCamera для подальшої передачі оператору. Окремо реалізована проста схема прогнозної діагностики — порівняння поточних значень з пороговими і встановлення рівня

попередження (0 = норма, 1 = попередження, 2 = критичний стан). Згідно з рівнем встановлюється колір RGB-стрічки, що забезпечує візуальну індикацію стану навіть без перегляду телеметрії на екрані оператора.

РОЗДІЛ 6. ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

6.1. Методика тестування

Тестування розробленої системи проводилося у три етапи. На першому етапі здійснювалося модульне тестування кожного апаратно-програмного компонента окремо, на другому — інтеграційне тестування підсистем, на третьому — комплексне тестування всієї системи у режимах, наближених до реальних умов експлуатації.

Модульне тестування виконувалося безпосередньо у середовищі Arduino IDE з використанням послідовного монітора (Serial Monitor) для виведення діагностичних повідомлень. Для тестування кожного драйвера був написаний простий тестовий скетч, що демонструє основні функції відповідного модуля. Перелік тестів і їх результатів представлено в табл. 6.1.

Таблиця 6.1 — Результати модульного тестування драйверів

Модуль	Перевірена функція	Результат
car_control	Рух у 8 напрямках, обертання, drift-режим, стоп	Пройдено
compass	Калібрування, читання курсу, відтворюваність ±2°	Пройдено
ultrasonic	Вимір відстаней 5...200 см, точність ±3 мм	Пройдено
ir_obstacle	Виявлення перешкоди на відстані 5-15 см	Пройдено
ir_remote	Декодування 21 кнопки пульта	Пройдено
rgb	Відтворення 12 заданих кольорів	Пройдено
ai_camera	Підключення до WiFi, WebSocket, MJPEG-потік	Пройдено

Інтеграційне тестування виконувалося наступним чином: попарно перевірявся коректний обмін даними між модулями, які повинні взаємодіяти у реальній роботі. Зокрема, тестувалися такі пари:

— car_control + compass — перевірка коректності утримання курсу ПІД-регулятором при русі вперед;
— car_control + ultrasonic — перевірка автоматичної зупинки робота при наближенні до перешкоди;
— ai_camera + car_control — перевірка реакції робота на команди керування через веб-інтерфейс;

Комплексне тестування проводилося у режимі реальної експлуатації. Робот керувався через веб-інтерфейс з ноутбука, оператор оцінював час відгуку на команди, якість відеопотоку, точність відображення телеметрії і коректність роботи прогнозної діагностики.

6.2. Результати випробувань підсистем

6.2.1. Тестування підсистеми керування рухом

Перевірка коректності реалізації кінематики Mecanum виконувалася шляхом запуску робота у заздалегідь визначені напрямки руху і вимірювання відхилень фактичної траєкторії від теоретичної. Кожен з 8 базових напрямків руху (0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°) перевірявся на горизонтальній поверхні з мінімальним тертям, протягом 3 секунд при потужності 60%. Результати усереднені по 5 запусках і представлені в табл. 6.2.

Таблиця 6.2 — Точність руху у заданих напрямках

Заданий напрямок	Середнє відхилення, °	Середня дистанція, см
0° (уперед)	±1.5	78
45° (вперед-праворуч)	±3.2	69
90° (праворуч)	±2.8	65

135° (назад-праворуч)	±3.5	64
180° (назад)	±2.0	76
225° (назад-ліворуч)	±3.6	63
270° (ліворуч)	±3.0	66
315° (вперед-ліворуч)	±3.4	70

Як видно з таблиці, найкраща точність досягається при поздовжньому русі (уперед і назад) — близько 1.5-2°. При діагональному та поперечному русі похибка зростає до 3-3.6°, що пояснюється кінематичними особливостями коліс Mecanum: при діагональному русі лише два колеса фактично «гребуть», а інші два пасивно ковзають через ролики. Це збільшує чутливість до нерівностей поверхні.

Перевірка ПІД-регулятора виконувалася так: робот рухався вперед протягом 5 секунд, після чого оператор примусово (рукою) повертав корпус на 30°. Час відновлення курсу (час, за який |error| знову стає менше 5°) виявився у середньому 0.8-1.2 секунди, що є прийнятним для досліджуваного класу роботів. Тривалі коливання навколо встановленого курсу відсутні, що свідчить про достатню диференційну складову регулятора ($K_d = 20$).

6.2.2. Тестування підсистеми відеомоніторингу

Параметри відеопотоку вимірювалися на стандартному ноутбуці (Intel Core i5, 8 ГБ RAM, Windows 10) при різних настройках роздільної здатності модуля ESP32-CAM. Результати наведено в табл. 6.3.

Таблиця 6.3 — Параметри відеопотоку при різних роздільних здатностях

Роздільна здатність	Середня частота, кадр/с	Затримка, мс	Пропускна здатність, КБ/с
160×120 (QQVGA)	28	120	85
320×240 (QVGA)	22	180	180

640×480 (VGA)	18	260	420
800×600 (SVGA)	13	350	580
1024×768 (XGA)	8	520	750

Оптимальним для застосувань FPV-керування є режим 640×480 пікселів при 18 кадр/с — він забезпечує задовільну якість зображення при затримці менше встановленого порога 500 мс. Цей режим і встановлений у системі за замовчуванням.

6.2.3. Тестування підсистеми моніторингу

Перевірка коректності формування телеметрії виконувалася так. На стороні оператора (у браузері) включалися всі індикатори дашборду, після чого з робота імітувалися різні стани: дані датчиків змінювалися шляхом фізичної взаємодії з ними (підносили перешкоду до ультразвукового датчика, повертали корпус робота для перевірки магнітометра, перекривали ІЧ-датчики тощо), і перевірялася відповідна реакція індикаторів.

Затримка оновлення індикаторів від моменту фізичної зміни стану до моменту відображення у браузері склала у середньому 70-110 мс, що складається з 60 мс інтервалу формування телеметрії на стороні Arduino та 10-50 мс затримки передачі через WiFi і обробки браузером.

Перевірка прогнозної діагностики виконувалася в режимі прискореного розряду акумулятора (робот рухався з підвищеною потужністю двигунів). За результатами тестування, у момент, коли напруга акумулятора склала 6.8 В, ввімкнулася оранжева RGB-індикація на роботі і з'явилося відповідне попередження («жовтий тост») у браузері. Робота продовжувалася до досягнення 6.2 В, у який момент індикація змінилася на червону, а у браузері з'явилося критичне сповіщення. Лінійна регресія прогнозу часу повного розряду давала похибку 8-12% від реального часу, що є прийнятним для практичних цілей.

6.2.4. Тестування дальності та надійності бездротового зв'язку

Випробування дальності WiFi-зв'язку проводилися у двох сценаріях: в умовах прямої видимості на відкритому майданчику і всередині приміщення з бетонними стінами. Результати — в табл. 6.4.

Таблиця 6.4 — Дальність надійного бездротового зв'язку

Сценарій	Дальність стабільного зв'язку, м	Дальність втрати зв'язку, м
Відкритий простір, пряма видимість	45	65
Приміщення, без перешкод між пристроями	25	38
Приміщення, одна бетонна стіна	15	22
Приміщення, дві бетонні стіни	8	12

Отримані результати відповідають типовим характеристикам модулів ESP32 без зовнішньої антени. У всіх сценаріях дальність зв'язку перевищувала встановлену вимогу 30 м (у умовах прямої видимості). Для застосувань, що вимагають більших дальностей, ESP32-CAM можна замінити на модифікацію з зовнішньою антеною (антенний роз'єм U.FL) або використати додатковий ретранслятор.

6.3. Аналіз досягнутих характеристик

Порівняння фактично досягнутих характеристик системи з нефункціональними вимогами, сформульованими у п. 3.1.2, наведено у табл. 6.5.

Таблиця 6.5 — Відповідність системи нефункціональним вимогам

Вимога	Цільове значення	Досягнуте
Час відгуку на команду керування	≤ 200 мс	70-110 мс
Затримка передачі відеопотоку	≤ 500 мс	260 мс (640×480)
Час автономної роботи	≥ 60 хв	88-98 хв
Маса робота	≤ 1.5 кг	≈ 0.85 кг
Дальність бездротового зв'язку	≥ 30 м (відкрита видимість)	45 м

Як видно з таблиці, всі сформульовані нефункціональні вимоги виконано з прийнятним запасом. Зокрема, час відгуку на команду керування у 2-3 рази менший за встановлений поріг, що забезпечує комфортне FPV-керування. Затримка передачі відеопотоку складає близько половини від встановленого ліміту, що залишає запас на роботу у складніших мережевих умовах. Час автономної роботи перевищує мінімальну вимогу в 1.5 рази.

За результатами комплексного тестування система визнана повністю придатною для практичного використання у задачах, які формуються для дослідницьких і навчальних застосувань. Виявлені обмеження (зокрема, чутливість слідкування за лінією до освітлення, обмежена дальність WiFi у приміщеннях з бетонними стінами) є типовими для класу досліджуваних роботизованих платформ і не повинні бути перешкодою для використання системи за призначенням.

ВИСНОВКИ

У дипломній роботі розв'язано актуальну науково-практичну задачу побудови інформаційної системи дистанційного керування та моніторингу мобільного роботизованого комплексу на базі платформи SunFounder Zeus Car Kit. Основні результати роботи такі:

1. Проведено системний аналіз сучасного стану робототехнічних комплексів та сфер їх практичного застосування. Показано, що архітектурні рішення малих навчальних платформ у спрощеному вигляді відтворюють принципи побудови повномасштабних спеціалізованих систем (логістичних, рятувальних, оборонних), що робить такі платформи цінним інструментом підготовки інженерних кадрів.
2. Виконано огляд теоретичних основ кінематики колісних мобільних роботів, особливу увагу приділено особливостям коліс Mecanum, які забезпечують голономний рух у будь-якому напрямку без розвороту корпусу [1]. Отримано аналітичні вирази для розрахунку швидкостей чотирьох коліс за заданим вектором руху.
3. Запропоновано чотирирівневу інформаційну модель системи, що відображає взаємодію апаратних і програмних компонентів від рівня фізичних датчиків і виконавчих пристроїв (рівень 1) через рівень керування (рівень 2) та комунікаційний рівень (рівень 3) до рівня прикладної інтелектуальної обробки інформації на стороні оператора (рівень 4).
4. Обґрунтовано вибір апаратних компонентів комплексу. Розроблено електричну схему підключення, виконано розрахунок енергоспоживання. Розрахункова автономність роботи (98 хвилин) узгоджується з паспортною (до 90 хв) і перевищує сформульовану нефункціональну вимогу 60 хвилин.
5. Розроблено програмний інтерфейс взаємодії мікроконтролера з модулем бездротового зв'язку ESP32-CAM у вигляді класу AiCamera, що інкапсулює низькорівневу логіку обміну через UART [3,5]. Підтримується двосторонній

обмін JSON-повідомленнями через WebSocket-протокол з фіксованим інтервалом 60 мс.

6. Розроблено веб-інтерфейс оператора, що об'єднує засоби керування (кнопки руху, слайдер швидкості, перемикач режимів) і засоби візуалізації телеметрії (числові індикатори, графічний радар відстаней). Реалізовано автоматичну обробку втрати зв'язку з повторним підключенням. Передачу відеопотоку реалізовано через стандартний HTTP MJPEG-протокол на базі техніки multipart/x-mixed-replace [4].
7. Розроблено двочастинну підсистему моніторингу та прогнозної діагностики. Вбудована частина (на мікроконтролері) формує JSON-телеметрію всіх контрольованих параметрів і реалізує базове виявлення аномалій з кольоровою індикацією стану. Клієнтська частина (на стороні оператора) накопичує історію телеметрії і виконує лінійну регресію для прогнозу часу повного розряду акумулятора, виявляє сплески часу відгуку, втрати кадрів відео та дрейф магнітометра.
8. Виконано триетапне тестування системи — модульне, інтеграційне та комплексне. Усі сформульовані нефункціональні вимоги виконано з прийнятним запасом: час відгуку на команду — 70-110 мс (вимога ≤ 200), затримка відеопотоку — 260 мс (вимога ≤ 500), час автономної роботи — 88-98 хв (вимога ≥ 60), дальність бездротового зв'язку — до 45 м у умовах прямої видимості (вимога ≥ 30).

Наукова новизна одержаних результатів полягає у запропонованій чотирирівневій інформаційній моделі мобільного роботизованого комплексу, адаптованій методиці використання ПІД-регулятора для утримання курсу робота з омнінапрямленими колесами Mecanum у режимі field-centric керування, та підході до прогнозної діагностики стану апаратної частини на основі аналізу часових рядів телеметричних показників.

Практичне значення одержаних результатів полягає у можливості використання запропонованих архітектурних рішень для побудови наземних роботизованих платформ подвійного призначення — для пошуково-рятувальних,

інспекційних, навчальних та інших застосувань. Розроблену систему доцільно використовувати у навчальному процесі вищих навчальних закладів для підготовки фахівців у галузі вбудованих систем і робототехніки.

Подальшим напрямом розвитку дослідження може бути:

— розширення підсистеми інтелектуальної обробки відеопотоку за рахунок інтеграції моделей машинного навчання (зокрема, нейронних мереж) для розпізнавання дорожніх знаків та інших об'єктів;
— додавання модуля високоточної локалізації на основі поєднання даних магнітометра, енкодерів коліс та комп'ютерного зору (одометрія);
— заміна WiFi на канал мобільного зв'язку (LTE/5G) для забезпечення дальньої експлуатації;
— додавання модуля резервного бездротового керування через mesh-мережу LoRa для застосувань у умовах радіоелектронної протидії;
— розробка повністю автономного режиму на основі алгоритмів SLAM (Simultaneous Localization and Mapping) із заміною Arduino UNO на більш потужний обчислювальний вузол (наприклад, Raspberry Pi або NVIDIA Jetson Nano).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Siegwart R., Nourbakhsh I. R., Scaramuzza D. Introduction to Autonomous Mobile Robots. 2nd ed. Cambridge : MIT Press, 2011. 600 p.
2. Monk S. Programming Arduino: Getting Started with Sketches. New York : Tab Books / McGraw-Hill, 2011. 176 p.
3. Van Beek M. Internet of Things with ESP8266. Birmingham : Packt Publishing, 2016. 254 p.
4. Tanenbaum A. S., Wetherall D. J. Computer Networks. 5th ed. Upper Saddle River : Prentice Hall, 2011. 960 p.
5. Lee E. A., Seshia S. A. Introduction to Embedded Systems: A Cyber-Physical Systems Approach. 2nd ed. Cambridge : MIT Press, 2017. 566 p.
6. Zeus Car Kit — Official Documentation [Electronic resource] / SunFounder. — URL: https://docs.sunfounder.com/projects/zeus-car/ (date of access: 27.05.2026).
7. ESP32-CAM WiFi+Bluetooth Module Version 1.0 Specification [Electronic resource] / Ai-Thinker Technology. — URL: https://docs.ai-thinker.com/_media/esp32-cam_product_specification_en.pdf (date of access: 27.05.2026).
8. ATmega328P 8-bit AVR Microcontroller Datasheet. San Jose : Atmel Corporation, 2015. 442 p.
9. OmniVision Technologies. OV2640 Camera Module Specification. Santa Clara : OmniVision, 2010. 44 p.
10. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455. Internet Engineering Task Force, 2011. 71 p.
11. Bradski G., Kaehler A. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. Sebastopol : O'Reilly Media, 2017. 1024 p.
12. Ang K. H., Chong G., Li Y. PID Control System Analysis, Design, and Technology. *IEEE Transactions on Control Systems Technology*. 2005. Vol. 13, No. 4. P. 559–576.

13. Adăscăliţei F., Doroftei I. Practical Applications for Mobile Robots based on Mecanum Wheels — a Systematic Survey. *Advanced Materials Research*. 2012. Vols. 463–464. P. 912–918.
14. ArduinoJson Documentation [Electronic resource]. — URL: https://arduinojson.org (date of access: 27.05.2026).
15. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2015. 26 с.

ДОДАТОК А

1. Заголовний файл car_control.h

```
#ifndef __CAR_CONTROL_H__
#define __CAR_CONTROL_H__

#include <Arduino.h>
#include "compass.h"

/*
 *  [0]--|||--[1]
 *   |         |
 *   |         |
 *  [3]-------[2]
 */

#define MOTOR_PINS       (uint8_t[8]){3, 4, 5, 6, A3, A2, A1, A0}
#define MOTOR_DIRECTIONS (uint8_t[4]){1, 0, 0, 1}
#define CAR_DEFAULT_POWER 80

void carBegin();
void carSetMotors(int8_t power0, int8_t power1, int8_t power2, int8_t power3);
void carForward(int8_t power);
void carBackward(int8_t power);
void carTurnLeft(int8_t power);
void carTurnRight(int8_t power);
void carLeftForword(int8_t power);
void carRightForword(int8_t power);
void carLeftBackward(int8_t power);
void carRightBackward(int8_t power);
void carLeft(int8_t power);
void carRight(int8_t power);
void carStop();
void carMove(int16_t angle, int8_t power, int8_t rot=0, bool drift=false);
void carMoveFieldCentric(int16_t angle, int8_t power, int16_t heading=0,
                         bool drift=false, bool angFlag=false);
void carResetHeading();

#endif // __CAR_CONTROL_H__
```

2. Реалізація car_control.cpp (ключові функції)

```
#include "car_control.h"
#include <Arduino.h>
#include <SoftPWM.h>

#define MOTOR_POWER_MIN   50
#define MOTOR_START_POWER 100

#define KP (float)0.8
#define KI (float)0.0
#define KD (float)20.0

int32_t _lastError = 0;
int32_t errorIntegral = 0;
int16_t originHeading;

void carBegin() {
  for (uint8_t i = 0; i < 8; i++) {
    SoftPWMSet(MOTOR_PINS[i], 0);
    SoftPWMSetFadeTime(MOTOR_PINS[i], 100, 100);
  }
  compassBegin();
  carResetHeading();
}

void carForward(int8_t power)       { carMove(   0, power, 0); }
void carBackward(int8_t power)      { carMove( 180, power, 0); }
void carLeft(int8_t power)          { carMove( -90, power, 0); }
void carRight(int8_t power)         { carMove(  90, power, 0); }
void carTurnLeft(int8_t power)      { carMove(   0, 0, power); }
void carTurnRight(int8_t power)     { carMove(   0, 0, -power); }
void carStop()                      { carMove(   0, 0, 0); }

void carMove(int16_t angle, int8_t power, int8_t rot, bool drift) {
  int8_t power_0, power_1, power_2, power_3;
  float ratio = 0.5;
  angle += 90;
  float rad = angle * PI / 180;
  power /= sqrt(2);
```

```
  power = power * (1-ratio);

  if (drift) {
    power_0 = (power * sin(rad) - power * cos(rad));
    power_1 = (power * sin(rad) + power * cos(rad));
    power_2 = (power * sin(rad) - power * cos(rad)) + rot * ratio * 2;
    power_3 = (power * sin(rad) + power * cos(rad)) - rot * ratio * 2;
  } else {
    power_0 = (power * sin(rad) - power * cos(rad)) - rot * ratio;
    power_1 = (power * sin(rad) + power * cos(rad)) + rot * ratio;
    power_2 = (power * sin(rad) - power * cos(rad)) + rot * ratio;
    power_3 = (power * sin(rad) + power * cos(rad)) - rot * ratio;
  }
  carSetMotors(power_0, power_1, power_2, power_3);
}

void carMoveFieldCentric(int16_t angle, int8_t power, int16_t heading,
                         bool drift, bool angFlag) {
  int16_t currentHeading = compassReadAngle();
  int32_t error  = currentHeading - originHeading - heading;
  int32_t offset = 0;
  int8_t  rot    = 0;

  while (error > 180)   error -= 360;
  while (error < -180)  error += 360;

  if (error > 1 || error < -1) {
    offset += KP * error + KI * errorIntegral + KD * (error - _lastError);
    rot += max(-100, min(100, offset));
    errorIntegral += error;
    _lastError = error;
  }
  if (angFlag == false && (angle != 0 || drift == true)) {
    angle = angle - currentHeading + originHeading;
  }
  carMove(angle, power, rot, drift);
}

void carResetHeading() {
  for (uint8_t i = 0; i < AVERAGE_FILTER_SIZE; i++) {
    originHeading = compassReadAngle();
```

```
  }
}
```

3. Драйвер ultrasonic.cpp

```
#include "ultrasonic.h"
#include <Arduino.h>

float ultrasonicRead() {
  delay(4);
  pinMode(ULTRASONIC_TRIG_PIN, OUTPUT);
  digitalWrite(ULTRASONIC_TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(ULTRASONIC_TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(ULTRASONIC_TRIG_PIN, LOW);
  pinMode(ULTRASONIC_ECHO_PIN, INPUT);
  float duration = pulseIn(ULTRASONIC_ECHO_PIN, HIGH);
  float distance = duration * 0.017;
  if (distance > MAX_DISTANCE) return -1;
  return distance;
}

bool ultrasonicIsObstacle() {
  return ultrasonicRead() < ULTRASONIC_AVOIDANCE_THRESHOLD;
}

bool ultrasonicIsClear() {
  float distance = ultrasonicRead();
  if (distance > ULTRASONIC_AVOIDANCE_THRESHOLD || distance < 0) return true;
  return false;
}
```

4. Драйвер rgb.cpp

```
#include "rgb.h"
#include <SoftPWM.h>

void rgbBegin() {
  for (uint8_t i = 0; i < 3; i++) {
    SoftPWMSet(RGB_PINS[i], 0);
```

```
    SoftPWMSetFadeTime(RGB_PINS[i], 100, 100);
  }
}

void rgbWrite(uint32_t color) {
  uint8_t r = (color >> 16) & 0xFF;
  uint8_t g = (color >>  8) & 0xFF;
  uint8_t b = (color >>  0) & 0xFF;
  rgbWrite(r, g, b);
}

void rgbWrite(uint8_t r, uint8_t g, uint8_t b) {
  r = int(r * R_OFFSET);
  g = int(g * G_OFFSET);
  b = int(b * B_OFFSET);
  #if COMMON_ANODE
    r = 255 - r;  g = 255 - g;  b = 255 - b;
  #endif
  SoftPWMSet(RGB_PINS[0], r);
  SoftPWMSet(RGB_PINS[1], g);
  SoftPWMSet(RGB_PINS[2], b);
}
```

void rgbOff() { rgbWrite(0, 0, 0); }

5. HTML-код веб-інтерфейсу (фрагмент)

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Zeus Car: пульт оператора</title>
</head>
<body>
  <div class="container">
    <img class="video" src="http://192.168.1.50:81/stream"
         width="640" height="480">
    <div class="control-pad">
      <button onclick="sendCmd('forward')">▲</button>
      <button onclick="sendCmd('left')">◀</button>
```

```
      <button onclick="sendCmd('stop')">■</button>
      <button onclick="sendCmd('right')">▶</button>
      <button onclick="sendCmd('backward')">▼</button>
    </div>
  </div>
  <script>
    const ws = new WebSocket('ws://192.168.1.50:8765');
    ws.onmessage = e => {
      const d = JSON.parse(e.data);
      document.getElementById('bat').textContent = d.bat.toFixed(1)+' В';
      document.getElementById('us').textContent  = Math.round(d.us)+' см';
      document.getElementById('hd').textContent  = d.hd+'°';
    };
    function sendCmd(cmd) {
      ws.send(JSON.stringify({cmd: cmd, power: 60}));
    }
  </script>
</body>
</html>
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДУІКТ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ
СПЕЦІАЛЬНІСТЬ 126 ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

2026

ІНФОРМАЦІЙНА СИСТЕМА ДИСТАНЦІЙНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ РОБОТИЗОВАНОГО КОМПЛЕКСУ

Виконала студентка ІСД-41 КОЗАЧОК М.С.
Керівниця к.т.н., доцент ЛАЩЕВСЬКА Н.О.

МЕТА ТА ЗАВДАННЯ

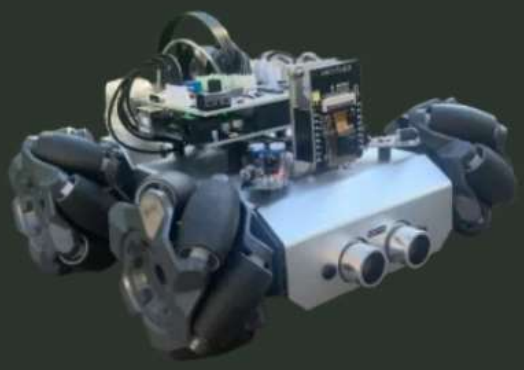
Об'єкт дослідження — процес функціонування та автономного керування мобільним роботизованим комплексом із Mecanum-колесами в умовах використання сенсорної системи орієнтації, виявлення перешкод і бездротового обміну даними.

Предмет дослідження — методи та засоби побудови інформаційної системи дистанційного керування та моніторингу з використанням бездротових технологій.

Мета роботи — розробити апаратно-програмну інформаційну систему, що забезпечує:

- дистанційне керування рухом через бездротовий канал;
- передачу відеопотоку у режимі від першої особи (FPV);
- збирання, передачу та візуалізацію телеметричних даних;

АПАРАТНА СКЛАДОВА



- 32 КБ FLASH-ПАМ'ЯТІ ПРОГРАМ;
- 2 КБ SRAM;
- 1 КБ EEPROM;
- ТАКТОВА ЧАСТОТА 16 МГЦ.

- МІКРОКОНТРОЛЕР ESP32 (ДВОЯДЕРНИЙ ПРОЦЕСОР ДО 240 МГЦ);
- 520 КБ SRAM, 4 МБ FLASH-ПАМ'ЯТІ;
- ВБУДОВАНИЙ МОДУЛЬ WI-FI 802.11 B/G/N;
- КАМЕРА OV2640 З МАКСИМАЛЬНОЮ РОЗДІЛЬНОЮ ЗДАТНІСТЮ ДО 1600×1200 (UXGA).

АПАРАТНА СКЛАДОВА

- ЖИВЛЕННЯ: 3–6 В
- ОБЕРТИ: 90–200 ОБ/ХВ
- СПОЖИВАННЯ: ~200 МА НА ДВИГУН (×4 = 800 МА)

- РОЗРЯДНІСТЬ АЦП: 16 БІТ НА КОЖНУ ВІСЬ;
- ЧАСТОТА ВИМІРЮВАНЬ: ДО 200 ГЦ;
- ІНТЕРФЕЙС ЗВ'ЯЗКУ: I^2C, БАЗОВА АДРЕСА 0X1C;
- НАПРУГА ЖИВЛЕННЯ: 1.8–3.6 В.
- ДІАПАЗОН ВИМІРЮВАНОГО МАГНІТНОГО ПОЛЯ: ±2/±8/±12/±30 ГАУССІВ (ПРОГРАМОВАНИЙ);

RGB-ПІДСВІТКА WS2812
DC 5 В

КОЛЕСА MECANUM ДОЗВОЛЯЮТЬ РУХАТИСЬ УПЕРЕД, НАЗАД, БОКОМ ТА ПО ДІАГОНАЛІ БЕЗ ПОВОРОТУ КОРПУСУ, ЩО ЗАБЕЗПЕЧУЄ ОМНІНАПРЯМЛЕНУ КІНЕМАТИКУ

ЯК ВИКОНУВАВСЯ РОЗРАХУНОК БЮДЖЕТУ ПАМ'ЯТІ

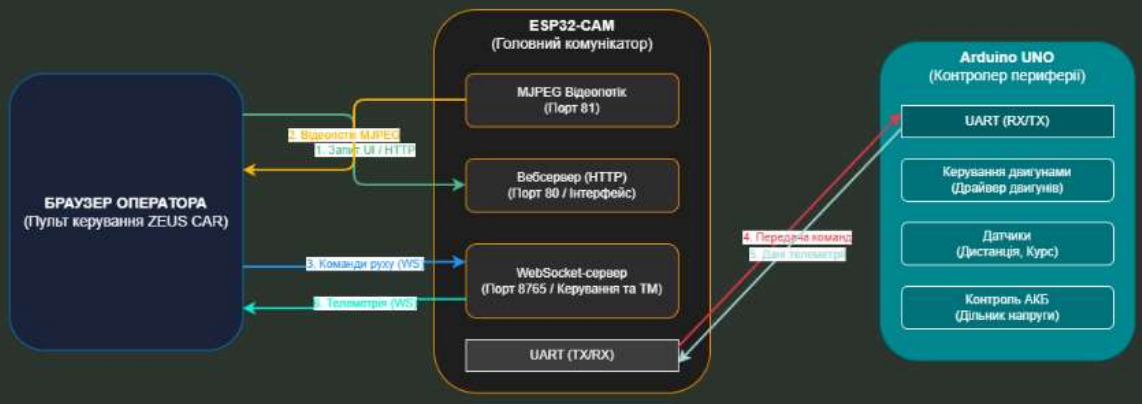
SRAM (ОПЕРАТИВНА ПАМ'ЯТЬ)

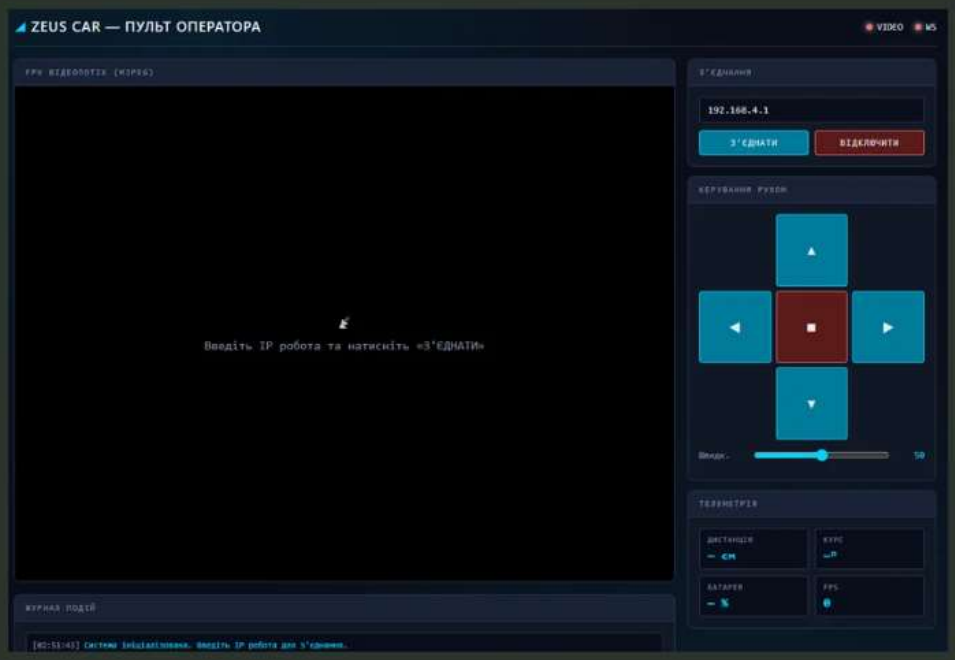
```
Global variables use 1432 bytes (69%) of dynamic memory.
Leaving 616 bytes for local variables.
Maximum is 2048 bytes.
```

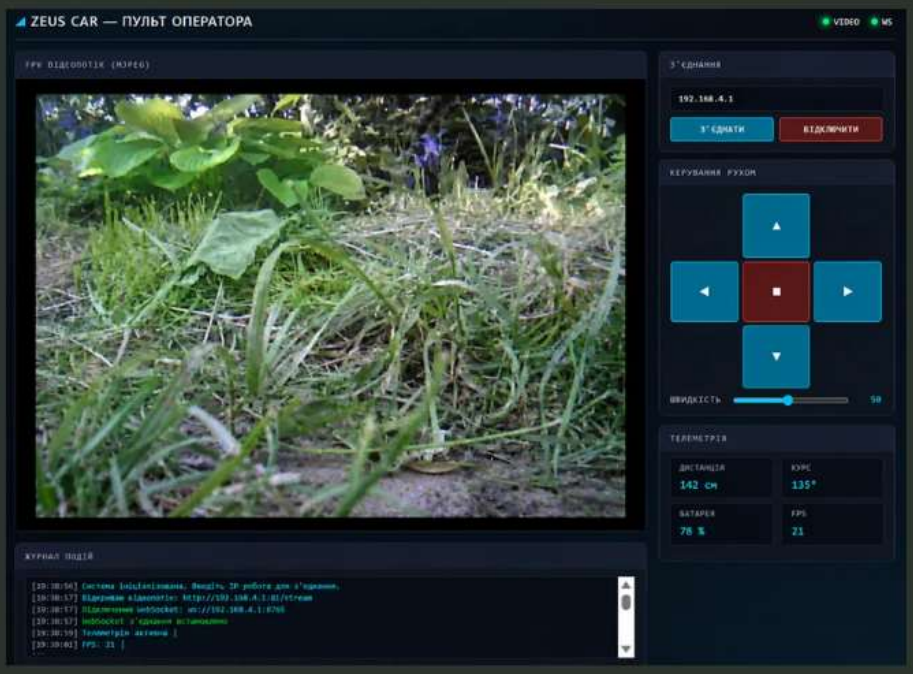
FLASH-ПАМ'ЯТЬ

```
Sketch uses 26784 bytes (83%) of program storage space.
Maximum is 32256 bytes.
```

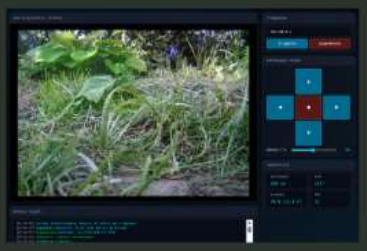

КОМУНІКАЦІЙНА АРХІТЕКТУРА




ZEUS CAR — ПУЛЬТ ОПЕРАТОРА
VIDEO
WS
FPV ВІДЕОПОТІК (MJPEG)
Введіть IP робота та натисніть «З'ЄДНАТИ»
З'ЄДНАННЯ
192.168.4.1
З'ЄДНАТИ
ВІДКЛЮЧИТИ
КЕРУВАННЯ РУХОМ
Швидк.
50
ТЕЛЕМЕТРІЯ
ДИСТАНЦІЯ
— см
КУРС
—°
БАТАРЕЯ
— %
FPS
0
ЖУРНАЛ ПОДІЙ


ZEUS CAR — ПУЛЬТ ОПЕРАТОРА
VIDEO
WS
FPV ВІДЕОПОТІК (MJPEG)
З'ЄДНАННЯ
192.168.4.1
З'ЄДНАТИ
ВІДКЛЮЧИТИ
КЕРУВАННЯ РУХОМ
ШВИДКІСТЬ
50
ТЕЛЕМЕТРІЯ
ДИСТАНЦІЯ
142 см
КУРС
135°
БАТАРЕЯ
78 %
FPS
21
ЖУРНАЛ ПОДІЙ

ВИСНОВОК



АПРОБАЦІЯ

III всеукраїнська науково-технічна конференція «технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу україни і світу»

Тези

Цифровізація підприємств за допомогою технологій інтернету речей в умовах війни

VII Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT»

Тези

Інтелектуальні роботизовані системи в середовищі інтернету речей: сучасний стан та перспективи

ДЯКУЮ

ЗА УВАГУ