Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно-інтегрованих технологій та систем
(повна назва)
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Розроблення мобільної платформи з дистанційним керуванням та передачею відеозображення через Web-інтерфейс
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТАКІТ-22-2
Всеволод ПИСАРЕНКО
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)

Керівник професор Сергій НОВОСЕЛОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри КІТАРБІ _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2026 р.

Я, Писаренко Всеволод Дмитрович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

"12" червня 2026 р.  Всеволод ПИСАРЕНКО

Харківський національний університет радіоелектроніки

Факультет АКІТС
Кафедра КІТАРБІ
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 28 » квітня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві Писаренку Всеволоду Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення мобільної платформи з дистанційним керуванням та передачею відеозображення через Web-інтерфейс
затверджена наказом університету від 04.06.2026 р. № 472 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 17.07.2026 р.
3. Вихідні дані до роботи
3.1 Мобільна платформа: чотириколісне шасі з DC-двигунами
3.2 Керуючий модуль: Raspberry Pi Zero 2 WH
3.3 Драйвер керування двигунами: L298N
3.4 Засіб отримання відеозображення: Raspberry Pi NoIR Camera v2
3.5 Датчик орієнтації та параметрів руху: інерціальний модуль MPU-9250
3.6 Серверна та клієнтська частини: Flask API, React.js, Tailwind CSS
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз предметної області та застосування дистанційного керування
4.3 Аналіз типів мобільних платформ, способів керування та передавання відео
4.4 Проєктування архітектури, структурної схеми та алгоритму роботи системи
4.5 Розроблення апаратної та програмної частин мобільної платформи
4.6 Експериментальне дослідження роботи системи
4.7 Охорона праці
4.8 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Демонстраційний матеріал в форматі (*.ppt), А4 – 10 сторінок

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Актуальність роботи, постановка задачі	28.04.26 – 02.05.26	викон.
2	Визначення мети, об'єкта і предмета розробки	03.05.26 – 07.05.26	викон.
3	Аналіз предметної області та існуючих рішень	08.05.26 – 18.05.26	викон.
4	Вибір апаратних і програмних засобів системи	19.05.26 – 27.05.26	викон.
5	Розроблення структурної схеми та алгоритму роботи системи	28.05.26 – 05.06.26	викон.
6	Розроблення програмно-апаратної частини мобільної платформи	06.06.26 – 22.06.26	викон.
7	Розроблення серверної частини та Web-інтерфейсу	23.06.26 – 03.07.26	викон.
8	Оформлення пояснювальної записки та демонстраційного матеріалу	04.07.26 – 05.07.26	викон.
9	Нормоконтроль та перевірка оригінальності роботи	06.07.26 – 08.07.26	викон.
10	Отримання відгуку керівника, рецензії та попередній захист	09.07.26 – 16.07.26	викон.
11	Подання кваліфікаційної роботи в ЕК	17.07.26	викон.

Дата видачі завдання 27.04.2026 р.

Здобувач _____ Всеволод ПИСАРЕНКО
(підпис) (власне ім'я, прізвище)

Керівник роботи _____ професор Сергій НОВОСЕЛОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 104 с., 13 табл., 32 рис., 2 дод., 45 джерел.

МОБІЛЬНА ПЛАТФОРМА, ДИСТАНЦІЙНЕ КЕРУВАННЯ, WEB-ІНТЕРФЕЙС, ВІДЕОЗОБРАЖЕННЯ, RASPBERRY PI, DC-ДВИГУН, REACT.

Об’єкт розробки – процес дистанційного керування наземною мобільною платформою з передаванням відеозображення користувачу.

Предмет розробки – програмно-апаратні засоби мобільної платформи з Web-інтерфейсом керування, передаванням відеозображення та відображенням стану системи.

Мета роботи – підвищення ефективності дистанційного керування мобільною платформою шляхом розроблення програмно-апаратної системи з Web-інтерфейсом та передаванням відеозображення.

Методи розробки: аналіз мобільних роботизованих платформ і способів дистанційного керування, обґрунтування вибору апаратних і програмних засобів, проєктування структури системи, розроблення серверної та клієнтської частин, експериментальна перевірка роботи системи.

У роботі розглянуто технічні рішення для побудови наземної мобільної платформи. Для реалізації системи використано чотириколісне шасі, Raspberry Pi Zero 2 WH, драйвер L298N, камеру та модуль MPU-9250. Серверну частину реалізовано на основі Flask API, клієнтський Web-інтерфейс – на основі React.js.

Розроблена система забезпечує дистанційне керування платформою, передавання відеозображення, створення знімків, виявлення руху та відображення стану у Web-інтерфейсі, а її результати можуть використовуватися для дистанційного спостереження й огляду приміщень та технічних зон.

Отримані результати можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», оскільки система поєднує робототехніку, Web-технології та засоби автоматизації.

ABSTRACT

Explanatory note: 104 pages, 13 tables, 32 figures, 2 appendices, 45 references.

MOBILE PLATFORM, REMOTE CONTROL, WEB INTERFACE, VIDEO IMAGE, RASPBERRY PI, DC MOTOR, REACT.

The object of development is the process of remote control of a ground mobile platform with video image transmission to the user.

The subject of development is the hardware and software tools of a mobile platform with a Web control interface, video image transmission and system status display.

The purpose of the work is to improve the efficiency of remote control of a mobile platform by developing a hardware and software system with a Web interface and video image transmission.

Development methods include the analysis of mobile robotic platforms and remote control methods, justification of hardware and software tools, system structure design, development of the server and client parts, and experimental testing of the system.

The work considers technical solutions for building a ground mobile platform. The system uses a four-wheel chassis, Raspberry Pi Zero 2 WH, L298N motor driver, camera, and MPU-9250 module. The server part is implemented using Flask API, while the client Web interface is implemented using React.js.

The developed system provides remote platform control, video image transmission, snapshot creation, motion detection and system status display in the Web interface, and its results can be used for remote observation and inspection of rooms and technical areas.

The obtained results can be related to Sustainable Development Goal 9 “Industry, Innovation and Infrastructure”, since the system combines robotics, Web technologies and automation tools.

ЗМІСТ

Перелік скорочень 9
Вступ 10
1 Аналіз технічних рішень для розроблення мобільної платформи з дистанційним керуванням 12
1.1 Аналіз предметної області 12
1.2 Аналіз типів мобільних платформ 13
1.3 Аналіз способів дистанційного керування 16
1.4 Аналіз інтерфейсів керування мобільними платформами 18
1.5 Аналіз способів передавання відеозображення 20
1.6 Аналіз периферійних апаратних засобів мобільної платформи 23
1.7 Аналіз програмних засобів системи 25
2 Проєктування структури програмно-апаратної системи 29
2.1 Архітектура програмно-апаратної системи 29
2.2 Структурна схема поєднання блоків системи 31
2.3 Алгоритм роботи системи 33
2.4 Застосування елементів теорії автоматичного управління 35
3 Апаратна та програмна реалізація системи 38
3.1 Вибір апаратної платформи керування 38
3.2 Підключення апаратних компонентів мобільної платформи 40
3.3 Реалізація серверної частини 54
3.4 Реалізація клієнтської частини 70
4 Експериментальне дослідження роботи системи 79
4.1 Загальний вигляд розробленої системи 79
4.2 Робота Web-інтерфейсу та відеопотоку 82
4.3 Робота системи керування рухом мобільної платформи 84
4.4 Робота інформаційних блоків Web-інтерфейсу 86
5 Охорона праці 90
5.1 Загальні положення 90

5.2 Аналіз небезпечних та шкідливих факторів 90
5.3 Вимоги безпеки під час роботи з комп’ютерною технікою та електронними компонентами 91
5.4 Пожежна безпека 92
Висновки 93
Перелік джерел посилання 95
Додаток А Фрагменти програмного коду серверної частини 102
Додаток Б Демонстраційний матеріал 103

ПЕРЕЛІК СКОРОЧЕНЬ

АКІТС – Факультет автоматизації, комп’ютерно-інтегрованих технологій та систем;

КІТАРБІ – Кафедра комп’ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії;

ХНУРЕ – Харківський національний університет радіоелектроніки;

ШІМ – широтно-імпульсна модуляція;

API – прикладний програмний інтерфейс;

CSI – послідовний інтерфейс камери;

CSS – каскадні таблиці стилів;

DC – постійний струм;

GPIO – виводи загального призначення;

GND – загальний провід, земля;

HTML – мова гіпертекстової розмітки;

HTTP – протокол передавання гіпертексту;

I2C – послідовна шина обміну даними між електронними компонентами;

IMU – інерціальний вимірювальний модуль;

IP – інтернет-протокол;

JPEG – формат стиснення цифрових зображень;

JSON – текстовий формат обміну даними;

JSX – розширення синтаксису JavaScript для опису інтерфейсу;

MJPEG – формат потокового передавання послідовності JPEG-кадрів;

SCL – лінія синхронізації інтерфейсу I2C;

SDA – лінія передавання даних інтерфейсу I2C;

USB – універсальна послідовна шина;

URL – уніфікований покажчик ресурсу.

ВСТУП

Мобільні роботизовані платформи є одним із поширених напрямів розвитку сучасної робототехніки, оскільки вони дають змогу поєднати рухому апаратну частину, засоби керування, датчики та програмне забезпечення в єдину систему. Завдяки цьому такі платформи можуть виконувати задачі, пов'язані з переміщенням у просторі, отриманням даних про навколишнє середовище та передаванням інформації користувачу на відстані.

Особливо актуальними є наземні мобільні платформи, які мають просту конструкцію, можуть переміщуватися в межах приміщень і передавати користувачу відеозображення в реальному часі. Для таких систем важливими є зручність керування, стабільна передача команд, можливість візуального контролю та використання доступних апаратних компонентів.

Одним із зручних способів взаємодії користувача з мобільною платформою є Web-інтерфейс. Перевага полягає в тому, що керування може виконуватися через звичайний браузер без встановлення окремої програми. Це спрощує використання системи та дозволяє працювати з нею з різних пристроїв, наприклад з комп'ютера, ноутбука або смартфона.

Тема кваліфікаційної роботи – «Розроблення мобільної платформи з дистанційним керуванням та передачею відеозображення через Web-інтерфейс». У межах роботи розробляється чотириколісна платформа з камерою та програмним забезпеченням, завдяки яким виконується керування пристроєм. Основою системи є Raspberry Pi Zero 2 WH для обробки команд керування, роботи з камерою, передачею даних користувачеві та взаємодії з драйвером двигунів.

Актуальність роботи полягає у необхідності в доступному рішенні, що об’єднує апаратні та програмні компоненти для керування мобільною платформою на відстані, отримувати відеозображення в реальному часі та моніторити функціювання системи.

Це може бути корисним для дистанційного огляду приміщень, спостереження за важкодоступними зонами та виконання задач, де користувачеві треба отримувати відеозображення без безпосередньої присутності.

Метою кваліфікаційної роботи є підвищення ефективності дистанційного керування мобільною платформою шляхом розроблення програмно-апаратної системи з Web-інтерфейсом та передаванням відеозображення.

Об’єктом розробки є процес дистанційного керування наземною мобільною платформою з передаванням відеозображення користувачу.

Предметом розробки є програмно-апаратні засоби мобільної платформи з Web-інтерфейсом керування, передаванням відеозображення та відображенням стану системи.

Для досягнення поставленої мети, необхідно вирішити такі задачі:

– проаналізувати предметну область та сфери застосування мобільних платформ з дистанційним керуванням;

– розглянути основні типи мобільних платформ, способи керування та передавання відеозображення;

– обґрунтувати вибір апаратних і програмних засобів для реалізації системи;

– розробити структуру програмно-апаратної системи та алгоритм її роботи;

– реалізувати апаратну частину мобільної платформи;

– розробити серверну частину та клієнтський Web-інтерфейс;

– виконати експериментальну перевірку роботи системи.

Під час виконання кваліфікаційної роботи враховано вимоги щодо академічної доброчесності, методичні рекомендації для підготовки кваліфікаційних робіт, а також вимог до структури та оформлення звітів у сфері науки та техніки [1] – [4].

1 АНАЛІЗ ТЕХНІЧНИХ РІШЕНЬ ДЛЯ РОЗРОБЛЕННЯ МОБІЛЬНОЇ ПЛАТФОРМИ З ДИСТАНЦІЙНИМ КЕРУВАННЯМ

1.1 Аналіз предметної області

Мобільні роботизовані платформи є окремим напрямом робототехнічних систем, у яких поєднуються механічна конструкція, виконавчі механізми, система керування, датчики та програмне забезпечення. Основною особливістю таких платформ є можливість переміщення у просторі та виконання дій без постійної фізичної участі оператора біля об’єкта. Це дозволяє використовувати мобільні платформи для дистанційного спостереження, огляду приміщень, контролю технічних зон та отримання інформації з віддаленої точки [5].

Дистанційне керування використовується у випадках, коли користувачеві потрібно керувати пристроєм на відстані та отримувати інформацію про навколишнє середовище без безпосередньої присутності біля об’єкта. Це може бути необхідним під час огляду важкодоступних або небезпечних зон, технічних приміщень, промислових об’єктів, прихованих ділянок або інших місць, де перебування людини є незручним чи небажаним [5].

Для мобільної платформи дистанційне керування має важливе значення, оскільки оператор може самостійно задавати напрямок руху, зупиняти платформу, змінювати її положення та виконувати огляд потрібної ділянки. Такий підхід є доцільним тоді, коли повна автономність системи не є необхідною, а рішення про подальший рух або зупинку приймає користувач.

Окреме значення має передавання відеозображення. Без відеопотоку користувач не має достатньої інформації про простір перед платформою, тому керування стає менш точним. Камера дозволяє бачити напрямок руху, оцінювати наявність перешкод, контролювати положення платформи та виконувати візуальний огляд об’єктів на відстані.

У таких системах важливим є не тільки переміщення платформи, а й зручна взаємодія з користувачем. Web-інтерфейс дозволяє керувати платформою через

звичайний браузер без встановлення окремого програмного забезпечення. Це спрощує доступ до системи та дає змогу використовувати для керування комп’ютер, ноутбук або інший пристрій, підключений до відповідної мережі.

У межах цієї роботи предметна область пов’язана з наземними мобільними платформами, які використовуються для дистанційного керування, передавання відеозображення та відображення стану системи у Web-інтерфейсі. Для таких задач важливими є стабільне передавання команд, наявність відеопотоку, можливість контролю стану системи та зрозумілий інтерфейс взаємодії з користувачем.

1.2 Аналіз типів мобільних платформ

Мобільні платформи відрізняються способом переміщення, конструкцією ходової частини, прохідністю, складністю керування та сферою застосування. За способом переміщення можна виділити колісні, гусеничні, крокуючі та повітряні мобільні платформи [6–8]. Така класифікація дозволяє оцінити, який тип платформи доцільно використовувати для конкретних умов роботи.

Колісні мобільні платформи мають просту конструкцію, добру швидкість руху, енергоефективність і зручне керування. Вони доцільні для роботи на рівних або відносно підготовлених поверхнях: у приміщеннях, складських зонах, на промислових майданчиках і під час дослідницьких робіт. Прикладом колісної платформи є Clearpath Husky A300, яка використовується для робототехнічних досліджень, розроблення автономних систем та інтеграції різних сенсорів [9].

Гусеничні мобільні платформи застосовуються тоді, коли потрібна підвищена прохідність і стійкість на складній поверхні. Завдяки більшій площі контакту з опорною поверхнею вони можуть працювати на нерівному ґрунті, долати перешкоди та використовуватися у небезпечних умовах. Прикладом такої платформи є Teledyne FLIR PackBot 525, який застосовується для знешкодження вибухонебезпечних предметів, спостереження, розвідки, CBRN-виявлення та роботи з небезпечними матеріалами [10].

Крокуючі мобільні платформи мають складнішу механічну конструкцію, але краще пристосовуються до нерівностей, сходів, вузьких проходів та об’єктів зі складною геометрією. Такі роботи можуть використовуватися для інспекції промислових об’єктів, технічних приміщень і зон, де колісна або гусенична платформа має обмеження. Прикладом є Boston Dynamics Spot, який позиціонується як мобільний робот для безпечнішого виконання операцій, збирання даних та роботи у складних для людини умовах [11].

Повітряні мобільні платформи, зокрема квадрокоптери та інші безпілотні літальні апарати, застосовуються для огляду об’єктів зверху, картографування, моніторингу територій, інспекції інфраструктури та задач громадської безпеки. Їхньою перевагою є швидке переміщення у просторі та доступ до ділянок, які складно оглянути з землі. Прикладом такої платформи є DJI Matrice 350 RTK, яка належить до професійних безпілотних платформ DJI Enterprise [12].

Порівняння основних типів мобільних платформ наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння основних типів мобільних платформ

Тип платформи	Основні переваги	Основні обмеження
Колісна	Проста конструкція, відносно низька складність керування, ефективний рух по рівних поверхнях	Приміщення, склади, промислові майданчики, дослідницькі роботи
Гусенична	Вища прохідність, більша площа контакту з поверхнею, кращий рух по нерівностях	Небезпечні зони, технічна розвідка, робота з небезпечними матеріалами
Крокуюча	Можливість руху по складному рельєфу та подолання окремих перешкод	Складна механіка, складне керування, потреба у стабілізації

Продовження таблиці 1.1

Тип платформи	Основні переваги	Основні обмеження
Повітряна	Переміщення у тривимірному просторі, можливість огляду з висоти	Обмежений час автономної роботи, складніша стабілізація, додаткові вимоги до безпеки

Приклади реальних мобільних платформ різних типів наведено на рисунку 1.1.



а) колісна



б) гусенична



в) крокуюча



г) повітряна

Рисунок 1.1 – Основні типи мобільних роботизованих платформ: а) колісна [9]; б) гусенична [10]; в) крокуюча [11]; г) повітряна [12]

Для цієї роботи обрано колісну мобільну платформу, оскільки вона має просту конструкцію, доступну апаратну реалізацію, достатню маневреність для

переміщення в межах приміщення та добре підходить для реалізації дистанційного керування через Web-інтерфейс. На відміну від гусеничних, крокуючих або повітряних систем, колісна платформа не потребує складної механіки, спеціальних умов експлуатації або складних алгоритмів стабілізації руху.

1.3 Аналіз способів дистанційного керування

Дистанційне керування мобільною платформою передбачає передавання команд від користувача до виконавчих механізмів без безпосереднього фізичного контакту з пристроєм. Для таких систем важливими є стабільність зв'язку, швидкість реакції на команди, зручність керування для оператора, безпека передавання даних та можливість отримання інформації про стан платформи. У роботах, присвячених віддаленому керуванню мобільними роботами, ці вимоги розглядаються як одні з основних під час побудови систем керування [13].

Одним із варіантів дистанційного керування є використання Bluetooth-з'єднання. Такий підхід часто застосовується у невеликих мобільних моделях, де команди передаються з мобільного пристрою або комп'ютера на мікроконтролер через Bluetooth-модуль. У роботі [14] розглянуто приклад мобільного робота, у якому Arduino отримує команди від Bluetooth-модуля і керує рухом моделі через драйвер двигунів. Перевагою такого способу є простота реалізації, однак його доцільність обмежується невеликою дальністю зв'язку та потребою у додатковому застосунку або окремому інтерфейсі керування.

Іншим способом є використання радіочастотного пульта керування. Радіоканал може застосовуватися у випадках, коли потрібно забезпечити керування мобільним роботом на більшій відстані. У роботі [15] розглядається розроблення радіочастотного пульта дальнього радіусу дії для мобільного робота, де основна увага приділяється збільшенню радіусу дії мережевої системи зв'язку під час віддаленого керування. Такий спосіб є доцільним для задач, де

основним є передавання команд руху, однак він не завжди зручний для одночасного відображення відеозображення, стану системи та інших даних у єдиному інтерфейсі.

Для мобільних платформ, що працюють у межах локальної мережі, може використовуватися Wi-Fi з'єднання. Такий спосіб дає змогу організувати обмін даними між платформою та пристроєм користувача без дротового підключення. У роботі [16] описано дистанційно керовану мобільну платформу з функцією відеомоніторингу, якою можна керувати з пристрою, підключеного до тієї ж Wi-Fi мережі, що і сама платформа. Це є близьким до задачі розроблення системи, у якій керування рухом, передавання відеозображення та взаємодія з користувачем виконуються через мережевий інтерфейс.

Web-інтерфейс у такій системі це спосіб взаємодії користувача з платформою через браузер. Його використання дозволяє об'єднати на одній сторінці елементи керування рухом, відеозображення, стан системи та додаткові функції. На відміну від керування через окремий пульт або спеціалізований мобільний застосунок, Web-інтерфейс може бути відкритий на різних пристроях, які мають браузер і доступ до відповідної мережі.

Основні переваги та обмеження розглянутих способів дистанційного керування наведено у таблиці 1.2.

Таблиця 1.2 – Порівняння способів дистанційного керування мобільною платформою

Спосіб керування	Основні переваги	Основні обмеження
Bluetooth	Простота реалізації, доступність модулів, можливість керування з мобільного пристрою	Обмежена дальність зв'язку, потреба у додатковому застосунку або окремому інтерфейсі

Продовження таблиці 1.2

Спосіб керування	Основні переваги	Основні обмеження
Радіочастотний пульт	Можливість керування на більшій відстані, швидка передача команд руху	Обмежені можливості для передавання відеозображення та стану системи в інтерфейсі
Wi-Fi	Можливість бездротового обміну даними у локальній мережі, підтримка передавання команд і даних	Залежність від якості мережі та стабільності підключення
Web-інтерфейс	Керування через браузер, можливість об'єднати відео, команди руху та стан системи в одному середовищі	Потреба у налаштуванні серверної частини та мережевого доступу

З урахуванням задачі розроблення мобільної платформи з дистанційним керуванням та передаванням відеозображення через Web-інтерфейс доцільним є використання керування через Wi-Fi. Такий підхід дозволяє передавати команди руху та відображати стан системи в одному середовищі без використання окремого пульта або спеціалізованого мобільного застосунку.

1.4 Аналіз інтерфейсів керування мобільними платформами

Інтерфейс керування мобільною платформою повинен давати оператору основні засоби для спостереження, керування та контролю стану системи. Тому під час аналізу важливо звернути увагу не на зовнішній стиль інтерфейсу, а на його функціональні блоки: відеозображення, елементи керування, індикатори стану, службові кнопки та додаткові інформаційні панелі.

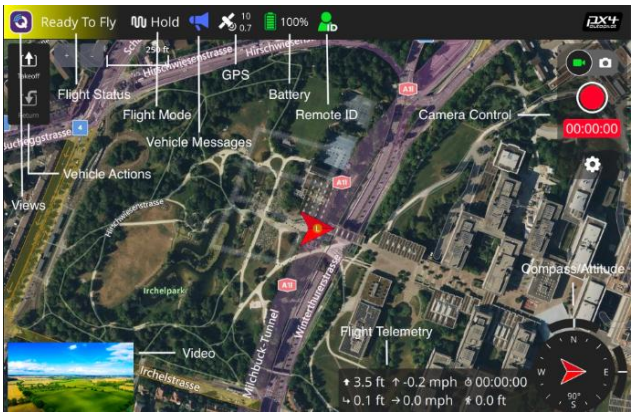
У розглянутих прикладах інтерфейсів можна побачити, що центральним елементом найчастіше є відеоблок або робоча зона спостереження. Це потрібно для того, щоб оператор міг бачити простір перед платформою, оцінювати ситуацію та приймати рішення під час керування. У системах DJI Fly та QGroundControl також передбачені елементи стану системи, зокрема режим роботи, рівень заряду, якість сигналу, параметри польоту або телеметричні дані [18], [19].



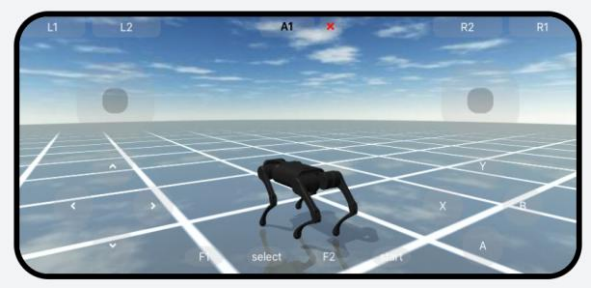
а) інтерфейс наземної роботизованої платформи



б) інтерфейс DJI Fly



в) інтерфейс QGroundControl



г) інтерфейс Unitree App

Рисунок 1.2 – Приклади інтерфейсів керування мобільними платформами: а) інтерфейс наземної роботизованої платформи [17]; б) інтерфейс DJI Fly [18]; в) інтерфейс QGroundControl [19]; г) інтерфейс Unitree App [20]

Крім відеоблоку, у таких інтерфейсах зазвичай присутні елементи керування. Вони можуть бути реалізовані у вигляді кнопок, віртуального

джойстика, панелі команд або окремих службових елементів. Їх призначення залежить від типу платформи: для дрона важливими є режими польоту, висота, навігація та камера, для наземної платформи – це рух, зупинка, стан підключення, відеоспостереження та додаткові функції системи [17] – [20].

Окрему роль має блок стану системи. У ньому можуть відображатися рівень заряду, режим роботи, стан підключення, телеметричні параметри, повідомлення про помилки або активні режими. Набір таких елементів залежить від призначення конкретної системи. Для розроблюваної мобільної платформи доцільно передбачити відображення статусу підключення, стану руху, режиму виявлення руху, а також основних службових повідомлень.

На основі аналізу розглянутих інтерфейсів можна визначити основні блоки, які доцільно реалізувати у Web-інтерфейсі мобільної платформи: відеоблок, панель керування рухом, блок стану системи та службові кнопки для взаємодії з камерою і режимами роботи. Така структура дозволяє об'єднати спостереження, керування та контроль стану платформи в одному інтерфейсі.

1.5 Аналіз способів передавання відеозображення

Передавання відеозображення є важливою частиною системи дистанційного керування мобільною платформою, оскільки оператор повинен бачити простір перед платформою та контролювати її рух. Для такої задачі можуть використовуватися різні варіанти отримання відео: камера, підключена безпосередньо до керуючого модуля, USB-камера або окрема IP-камера. Вибір способу залежить від апаратної платформи, вимог до якості зображення, затримки, складності підключення та способу передавання відео у Web-інтерфейс.

Одним із варіантів є використання USB-камери. Такий спосіб є простим у підключенні та може використовуватися з багатьма одноплатними комп'ютерами. Недоліком є те, що USB-камера займає USB-порт, може потребувати додаткового налаштування драйверів і не завжди є оптимальною

для компактної мобільної платформи.

Іншим варіантом є IP-камера, яка передає відеопотік через мережу. Такий підхід може бути зручним для систем відеоспостереження, однак для невеликої мобільної платформи він ускладнює конструкцію, оскільки камера є окремим мережевим пристроєм і потребує власного живлення або додаткового налаштування.

Для систем на базі Raspberry Pi доцільним є використання офіційної камери Raspberry Pi, яка підключається до плати через CSI-інтерфейс. В офіційній документації Raspberry Pi зазначено, що Camera Module 2 має 8-мегапіксельний сенсор і випускається у стандартній та NoIR-версії, а камери Raspberry Pi сумісні з платами, які мають CSI-роз'єм [21]. Це дозволяє використовувати камеру як частину єдиної програмно-апаратної системи без застосування окремого зовнішнього відеопристрою.

Для передавання відеозображення у Web-інтерфейс можуть використовуватися різні підходи. Простим варіантом є передавання послідовності кадрів у форматі MJPEG через HTTP. Такий спосіб зручний для відображення відео у браузері та може бути реалізований на серверній частині системи. Також для відео можуть застосовуватися H.264, RTSP або WebRTC, які краще підходять для складніших сценаріїв потокового відео, але потребують більшої кількості налаштувань [22], [23].

У межах даної роботи передавання відеозображення розглядається як частина Web-системи керування мобільною платформою. Камера отримує зображення, серверна частина обробляє та передає його користувачу, а Web-інтерфейс відображає відеоблок разом з елементами керування та станом системи. Такий підхід дозволяє поєднати відеоспостереження і дистанційне керування в одному робочому середовищі.

Таблиця 1.3 – Порівняння способів отримання та передавання відеозображення

Спосіб	Основні переваги	Основні обмеження
USB-камера	Просте підключення, доступність, можливість використання стандартних засобів роботи з відео	Займає USB-порт, може потребувати додаткових драйверів або налаштування
IP-камера	Передавання відео через мережу, можливість автономної роботи як окремого пристрою	Потреба в окремому живленні або налаштуванні мережі
Raspberry Pi Camera через CSI	Компактне підключення до Raspberry Pi, сумісність з камерою Raspberry Pi, зручність використання в єдиній системі	Прив'язка до плат Raspberry Pi з CSI-роз'ємом
MJPEG через HTTP	Просте відображення відео у Web-інтерфейсі, відносно нескладна реалізація на серверній частині	Може мати більший обсяг переданих даних порівняно з сучасними відеокодеками
RTSP або WebRTC	Краще підходить для повноцінного потокового відео та складніших мережевих сценаріїв	Складніша реалізація та потреба у додатковому налаштуванні

На основі розглянутих варіантів доцільно використовувати камеру Raspberry Pi NoIR Camera v2. Такий варіант відповідає обраній апаратній платформі, дозволяє отримувати відеозображення без окремого мережевого пристрою та спрощує інтеграцію камери з серверною частиною системи. Передавання зображення у Web-інтерфейс доцільно реалізувати через серверну частину, оскільки це дозволяє відображати відео разом з елементами керування, станом системи та іншими функціями платформи [21], [22].

1.6 Аналіз периферійних апаратних засобів мобільної платформи

Після вибору основної апаратної платформи керування необхідно визначити периферійні засоби, які забезпечують виконання основних функцій мобільної платформи. До таких засобів належать драйвер двигунів, камера для отримання відеозображення та інерціальний модуль для зчитування параметрів орієнтації.

Для керування рухом платформи необхідно використовувати окремий драйвер двигунів, оскільки Raspberry Pi не призначений для безпосереднього живлення двигунів постійного струму. Драйвер виконує роль проміжного елемента між керуючим модулем і виконавчими механізмами. Він приймає керуючі сигнали та забезпечує подавання живлення на двигуни. У роботі [31], присвяченій розробці структурної схеми підключення роботизованої платформи, також розглядається використання драйвера двигунів як складової апаратної частини системи, там було обрано драйвер L298N, оскільки мікросхема L298 може використовуватися для керування двигунами постійного струму [30], [33]. На рисунку 1.3 загальний вигляд драйверу L298N.



Рисунок 1.3 – Драйвер двигунів L298N [33]

Для отримання відеозображення доцільно використовувати камеру, яка сумісна з Raspberry Pi та може бути підключена безпосередньо до плати. В

офіційній документації Raspberry Pi зазначено, що Camera Module 2 має 8-мегапіксельний сенсор Sony IMX219 і випускається у стандартній та NoIR-версії. NoIR-версія не має інфрачервоного фільтра, тому може використовуватися разом з інфрачервоним підсвічуванням в умовах недостатнього освітлення [21]. З огляду на це для мобільної платформи обрано Raspberry Pi NoIR Camera v2 (рис. 1.4), оскільки вона дозволяє отримувати відеозображення та розширює можливості дистанційного спостереження у темних приміщеннях або при слабкому освітленні.

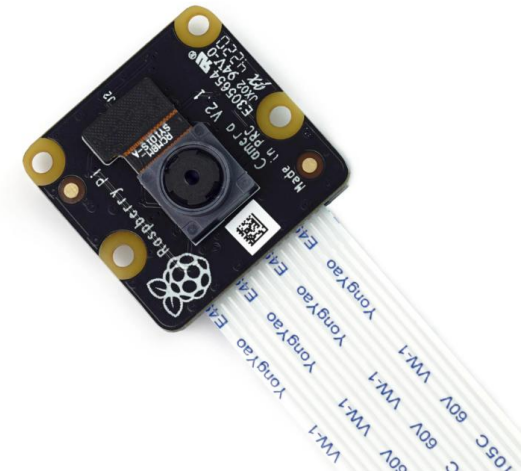


Рисунок 1.4 – Камера Raspberry Pi NoIR Camera v2 [21]

Для отримання додаткової інформації про положення та рух платформи використовується інерціальний модуль MPU-9250. Згідно з технічною документацією, MPU-9250 є 9-осьовим модулем, який поєднує тривісний акселерометр, тривісний гіроскоп і тривісний магнітометр [32]. Завдяки цьому модуль може використовуватися для отримання даних про нахил, прискорення та орієнтацію платформи. Для розроблюваної системи такі дані є корисними, оскільки вони дозволяють розширити інформацію про стан мобільної платформи та відображати її у Web-інтерфейсі.

Зовнішній вигляд інерціального модуля MPU-9250 наведено на рисунку 1.5.

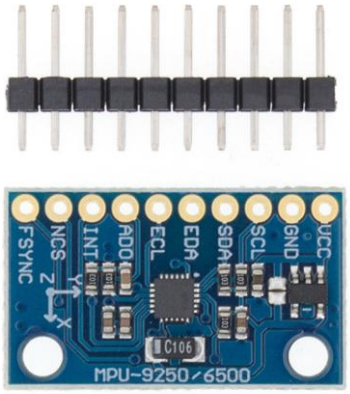


Рисунок 1.5 – Інерціальний модуль MPU-9250 [32]

Таким чином, для реалізації периферійної апаратної частини мобільної платформи обрано драйвер двигунів L298N, Raspberry Pi NoIR Camera v2 та інерціальний модуль MPU-9250. Ці компоненти забезпечують керування рухом, отримання відеозображення та зчитування параметрів орієнтації.

1.7 Аналіз програмних засобів системи

Для реалізації мобільної платформи з дистанційним керуванням недостатньо лише апаратної частини. Необхідно також розробити програмну частину, яка забезпечить взаємодію між користувачем, Web-інтерфейсом та апаратними засобами платформи. У такій системі серверна частина приймає запити від Web-інтерфейсу, обробляє їх і передає команди до апаратної частини. У зворотному напрямку вона отримує дані про стан системи, відеозображення або службову інформацію та передає їх користувачу.

Клієнтська частина потрібна для того, щоб користувач міг працювати з платформою через браузер. У Web-інтерфейсі мають відображатися відеоблок, елементи керування рухом, стан системи та додаткові службові функції. Тому програмні засоби системи доцільно розглядати за трьома напрямами: серверна частина, клієнтська частина та засоби стилізації інтерфейсу.

1.7.1 Аналіз засобів серверної частини

Серверна частина мобільної платформи повинна приймати команди від користувача, передавати їх до апаратної частини, повертати інформацію про стан системи та забезпечувати взаємодію з камерою.

Для реалізації серверної частини на Python можна розглядати Flask і Django. Flask в офіційній документації описується як легкий Web-фреймворк, який дозволяє швидко почати розробку і може масштабуватися до складніших застосунків [34]. Такий підхід добре підходить для системи, де серверна частина повинна виконувати конкретні задачі, такі як приймання HTTP-запитів, обробка команд керування, повернення статусу системи та передача відеозображення.

Django також є Python фреймворком, однак він орієнтований на створення повноцінних Web-застосунків. В офіційній документації Django розглядається як інструмент для database-driven Web app, де значна увага приділяється моделям даних, ORM, структурі проєкту та типовим задачам Web-розробки [35]. Для великої інформаційної системи це є перевагою, але для мобільної платформи з простими API-запитами така структура може бути надлишковою.

З урахуванням задачі розроблення системи дистанційного керування доцільно використати Flask. Він дозволяє реалізувати серверну частину без зайвої складності, організувати маршрути для команд керування, отримання стану системи, роботи з камерою та взаємодії з Web-інтерфейсом. Тому Flask обрано як основу серверної частини системи [34].

1.7.2 Аналіз засобів клієнтської частини

Клієнтська частина відповідає за відображення Web-інтерфейсу, через який користувач керує мобільною платформою. Такий інтерфейс складається з кількох окремих блоків, наприклад, області відеозображення, панелі керування рухом, блоку стану системи, службових кнопок та додаткових елементів. Тому для розроблення клієнтської частини доцільно використовувати засіб, який дозволяє зручно будувати інтерфейс із незалежних компонентів.

React є JavaScript бібліотекою для створення користувацьких інтерфейсів.

В офіційній документації зазначено, що інтерфейс у React може будуватися з невеликих багаторазових компонентів, які можна вкладати один в одного та об'єднувати у більші частини сторінки [36]. Такий підхід добре відповідає структурі інтерфейсам, оскільки відеоблок, панель керування, блок стану та службові елементи можна реалізувати як окремі компоненти.

Також важливою перевагою React є зручна робота зі станом інтерфейсу. На Web-сторінці частина даних, таких як, стан підключення, режим руху, активність режиму виявлення руху, службові повідомлення або інші параметри, може змінюватися під час роботи системи. React дозволяє зберігати такі дані у стані компонентів і оновлювати відповідні елементи сторінки після їх зміни [36]. Це є зручним для системи дистанційного керування, де користувач повинен бачити актуальний стан платформи без ручного оновлення сторінки.

1.7.3 Аналіз засобів стилізації Web-інтерфейсу

Окрім логіки роботи Web-інтерфейсу, важливим є його зовнішнє оформлення. Інтерфейс мобільної платформи повинен бути зрозумілим для користувача, мати чітко виділені блоки керування, відеозображення та стану системи. Для цього необхідно використати засіб стилізації, який дозволяє швидко оформлювати елементи сторінки та підтримувати єдиний вигляд інтерфейсу.

Tailwind CSS є CSS-фреймворком, у якому готові службові класи можна комбінувати безпосередньо в розмітці для побудови потрібного дизайну [37]. Такий підхід дозволяє швидко задавати відступи, розміри, кольори, розташування елементів, адаптивність та інші параметри інтерфейсу без написання великої кількості окремого CSS-коду.

Для даної роботи Tailwind CSS є доцільним засобом стилізації, оскільки Web-інтерфейс складається з багатьох візуальних блоків, зокрема відеопанелі, кнопок керування, карток стану системи та службових елементів. Використання utility-класів дозволяє швидко оформлювати ці блоки, зберігати єдиний стиль сторінки та спрощувати подальше редагування інтерфейсу.

Таким чином, для стилізації Web-інтерфейсу обрано Tailwind CSS. Він дозволяє швидко створити зрозумілий інтерфейс керування мобільною платформою та забезпечити зручне розміщення основних елементів сторінки [37].

2 ПРОЄКТУВАННЯ СТРУКТУРИ ПРОГРАМНО-АПАРАТНОЇ СИСТЕМИ

2.1 Архітектура програмно-апаратної системи

Програмно-апаратна система мобільної платформи призначена для дистанційного керування рухом, передавання відеозображення користувачу та відображення службової інформації про стан системи. Для виконання цих задач система повинна поєднувати апаратну частину, серверну частину та клієнтський Web-інтерфейс.

Апаратна частина забезпечує фізичне переміщення мобільної платформи, отримання відеозображення та зчитування додаткових даних із сенсорних модулів. До її складу входять мобільна база з двигунами, керуючий модуль, драйвер двигунів, камера та сенсорний модуль. Апаратна частина виконує команди, які надходять від серверної частини, і формує інформацію про поточний стан системи.

Серверна частина є проміжною ланкою між Web-інтерфейсом користувача та апаратними засобами мобільної платформи. Вона приймає запити від клієнтської частини, обробляє команди керування, передає сигнали на апаратну частину, отримує відеозображення та службові дані, а також повертає інформацію про стан системи користувачу.

Клієнтський Web-інтерфейс призначений для взаємодії користувача з мобільною платформою. Через нього користувач може подавати команди руху, переглядати відеозображення, отримувати інформацію про стан системи та виконувати додаткові дії, передбачені функціональністю системи. Використання Web-інтерфейсу дає змогу керувати платформою через браузер без встановлення окремого програмного забезпечення.

Загальна архітектура системи побудована за принципом поділу на функціональні блоки. Користувач взаємодіє з клієнтським Web-інтерфейсом, Web-інтерфейс передає запити до серверної частини, серверна частина обробляє

ці запити та взаємодіє з апаратними компонентами мобільної платформи. У зворотному напрямку передаються відеозображення, службові дані та інформація про поточний стан системи.

Основні функціональні блоки програмно-апаратної системи наведено у таблиці 2.1.

Таблиця 2.1 – Основні функціональні блоки програмно-апаратної системи

Функціональний блок	Призначення
Користувач	Формує команди керування та отримує інформацію про стан системи
Клієнтський Web-інтерфейс	Забезпечує взаємодію користувача із системою через браузер
Серверна частина	Обробляє запити, передає команди керування та формує відповіді для Web-інтерфейсу
Керуючий модуль	Забезпечує взаємодію серверної частини з апаратними компонентами
Драйвер двигунів	Перетворює керуючі сигнали у сигнали для роботи двигунів
Мобільна база	Забезпечує фізичне переміщення платформи
Камера	Формує відеозображення для передавання користувачу
Сенсорний модуль	Надає додаткові дані про стан мобільної платформи

Такий поділ дозволяє визначити роль кожної частини системи та показати загальну логіку взаємодії між користувачем, програмними засобами та апаратною частиною мобільної платформи. Архітектура системи забезпечує передавання команд керування до виконавчих механізмів, отримання відеозображення та відображення службової інформації про стан платформи.

2.2 Структурна схема поєднання блоків системи

Для опису взаємодії основних частин програмно-апаратної системи доцільно побудувати структурну схему поєднання блоків. Така схема дозволяє показати, як між собою пов’язані користувач, Web-інтерфейс, серверна частина, керуючий модуль та апаратні засоби мобільної платформи.

У системі можна виділити два основні напрями передавання інформації. Перший напрям пов’язаний із передаванням команд керування від користувача до мобільної платформи. Користувач формує команду через Web-інтерфейс, після чого вона передається до серверної частини. Серверна частина обробляє отриману команду та передає її на керуючий модуль. Керуючий модуль формує керуючі сигнали для драйвера двигунів, а драйвер двигунів забезпечує роботу виконавчих механізмів мобільної бази.

Другий напрям пов’язаний із передаванням інформації від мобільної платформи до користувача. Камера формує відеозображення, яке через серверну частину передається до Web-інтерфейсу. Сенсорний модуль надає додаткові дані про стан платформи, а серверна частина обробляє ці дані та передає їх користувачу. Також серверна частина формує службову інформацію про стан системи, режим роботи та події, що виникають під час роботи платформи.

Структурна схема поєднання блоків системи повинна відображати взаємодію таких основних елементів: користувача, клієнтського Web-інтерфейсу, серверної частини, керуючого модуля, драйвера двигунів, мобільної бази, камери та сенсорного модуля. Окремо на схемі доцільно показати передавання команд керування, відеозображення, службових даних і зворотної інформації про стан системи.

Структурну схему поєднання блоків програмно-апаратної системи наведено на рисунку 2.1.

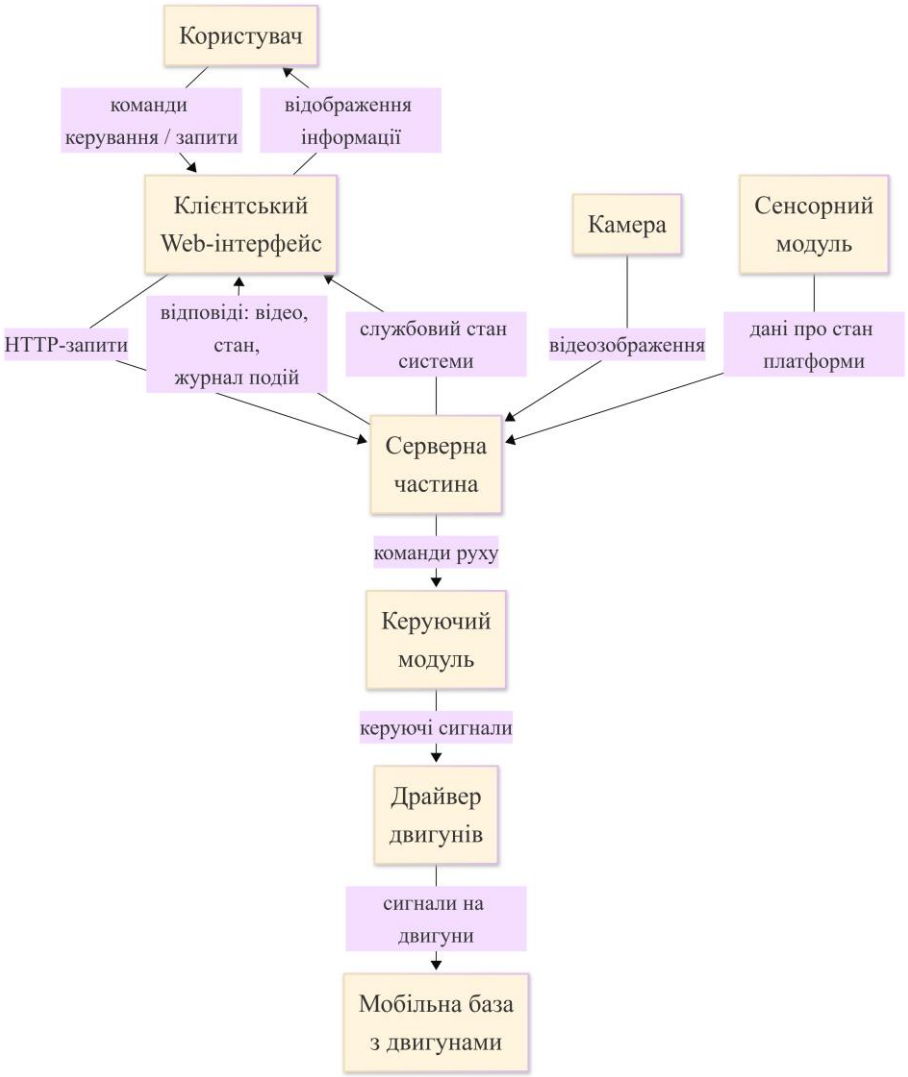


Рисунок 2.1 – Структурна схема поєднання блоків програмно-апаратної системи

На рисунку 2.1 показано, що центральною ланкою взаємодії є серверна частина, яка забезпечує обмін даними між Web-інтерфейсом користувача та апаратною частиною мобільної платформи. Команди керування передаються від користувача до виконавчих механізмів, а у зворотному напрямку надходять відеозображення, службова інформація та дані про стан системи.

Структурна схема відображає основні зв’язки між блоками системи та напрямки передавання інформації. Команди керування передаються від користувача до виконавчих механізмів мобільної платформи, а у зворотному напрямку надходять відеозображення, службові дані та інформація про стан системи.

2.3 Алгоритм роботи системи

Алгоритм роботи програмно-апаратної системи мобільної платформи побудований за подієво-циклічним принципом.

Після запуску система виконує ініціалізацію основних функціональних блоків, переходить у режим очікування запитів від Web-інтерфейсу та надалі обробляє дії користувача або службові події системи.

На початковому етапі виконується підготовка системи до роботи: ініціалізація серверної частини, перевірка доступності апаратних блоків, підготовка камери, сенсорного модуля та виконавчих механізмів. Після цього система переходить у режим очікування запитів від клієнтського Web-інтерфейсу.

Під час роботи користувач через Web-інтерфейс може подавати команди руху, надсилати запити на отримання відеозображення, створення знімка, зміну режиму роботи камери або отримання інформації про стан системи. Серверна частина приймає відповідний запит, визначає його тип і виконує необхідну дію. Якщо надходить команда руху, серверна частина обробляє її та передає керуючий сигнал до апаратної частини системи. Далі керуючий модуль формує сигнали для драйвера двигунів, після чого мобільна база виконує відповідну дію: рух уперед, рух назад, поворот або зупинку. Після виконання команди стан системи оновлюється та відображається у Web-інтерфейсі.

Якщо користувач запитує відеозображення, камера формує потік кадрів, який передається через серверну частину до Web-інтерфейсу. У разі створення знімка система фіксує поточний кадр і зберігає його як окреме зображення. Якщо користувач змінює режим роботи камери, серверна частина обробляє відповідну команду та оновлює параметри отримання відеозображення.

Окремо в алгоритмі передбачено отримання службової інформації про стан системи. До неї належать дані сенсорного модуля, стан режиму виявлення руху, повідомлення журналу подій та загальна інформація про працездатність основних блоків. Якщо певний додатковий модуль не передає дані, система не

припиняє роботу, а відображає відповідний стан у Web-інтерфейсі.

Після обробки кожної команди або службової події система повертається до режиму очікування наступного запиту. Такий принцип роботи дозволяє постійно підтримувати взаємодію між користувачем, Web-інтерфейсом, серверною частиною та апаратними блоками мобільної платформи.

Алгоритм роботи програмно-апаратної системи наведено на рисунку 2.2.

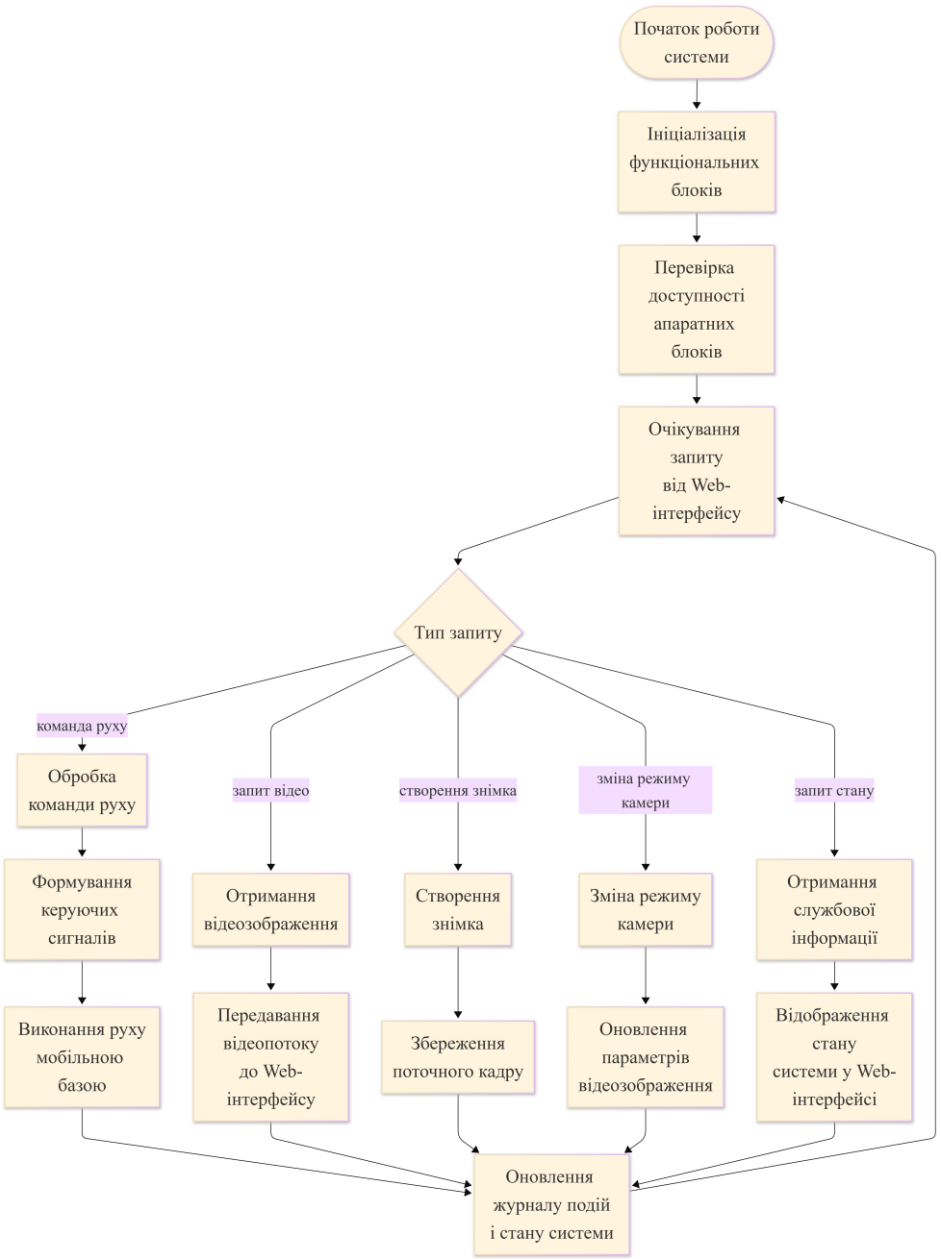


Рисунок 2.2 – Алгоритм роботи програмно-апаратної системи мобільної платформи

2.4 Застосування елементів теорії автоматичного управління

Теорія автоматичного управління використовується для опису принципів побудови систем керування, визначення основних елементів системи та аналізу взаємодії між задавальним впливом, пристроєм керування, об’єктом керування і зворотною інформацією [13, 14]. У межах розробленої мобільної платформи елементи теорії автоматичного управління доцільно використати для опису загальної структури керування рухом платформи.

У даній системі об’єктом керування є мобільна база з виконавчими механізмами, яка змінює своє положення в просторі відповідно до команд користувача. Задавальним впливом є команда оператора, сформована через Web-інтерфейс. Такою командою може бути рух уперед, рух назад, поворот ліворуч, поворот праворуч або зупинка. Пристроєм керування виступає сукупність серверної частини, керуючого модуля та драйвера двигунів, які приймають команду, обробляють її та формують керуючі сигнали для виконавчих механізмів.

Виконавчими механізмами системи є двигуни мобільної платформи. Вони перетворюють керуючі сигнали у механічний рух платформи. У результаті цього змінюється положення мобільної бази відносно навколишнього середовища. Збурювальними впливами для такої системи можуть бути нерівності поверхні, прослизання коліс, зміна заряду джерела живлення, затримки передавання команд або перешкоди під час руху.

Зворотна інформація в системі передається користувачу через відеозображення, службові повідомлення та дані про стан системи у Web-інтерфейсі. На основі цієї інформації оператор оцінює положення платформи, наявність перешкод і результат виконання попередньої команди. Після цього користувач формує наступну команду керування. Таким чином, у системі реалізується контур дистанційного керування з участю оператора.

Відповідність елементів розробленої системи основним поняттям теорії автоматичного управління наведено у таблиці 2.2.

Таблиця 2.2 – Відповідність елементів системи поняттям теорії автоматичного управління

Поняття ТАУ	Відповідний елемент	Пояснення
Задавальний вплив	Команда користувача у Web-інтерфейсі	Визначає бажану дію платформи
Пристрій керування	Серверна частина, керуючий модуль, драйвер двигунів	Обробляє команду та формує керуючий сигнал
Об’єкт керування	Мобільна база з двигунами	Змінює положення відповідно до керуючого впливу
Виконавчий механізм	Двигуни мобільної платформи	Перетворюють керуючий сигнал у механічний рух
Керуючий вплив	Сигнали, що подаються на двигуни	Забезпечують рух або зупинку платформи
Зворотна інформація	Відеозображення, стан системи, журнал подій	Дозволяє оператору оцінити результат керування
Збурення	Нерівності поверхні, прослизання, затримки, зміна живлення	Можуть впливати на точність і стабільність руху

Узагальнену структуру контуру дистанційного керування мобільною платформою наведено на рисунку 2.3.

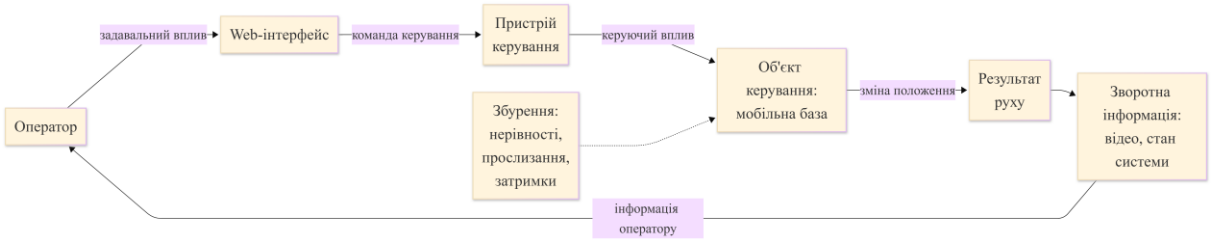


Рисунок 2.3 – Узагальнена структура контуру дистанційного керування мобільною платформою

Структурно процес керування можна подати у вигляді послідовності: користувач задає команду, пристрій керування формує керуючий вплив, мобільна база виконує рух, а інформація про стан системи повертається користувачу через Web-інтерфейс. Такий підхід відповідає загальній структурі системи управління, у якій результат дії об’єкта оцінюється за допомогою зворотної інформації.

Для розробленої мобільної платформи основною регульованою величиною можна вважати напрямок і стан руху платформи. Керуючим впливом є сигнали, що визначають роботу двигунів. Оскільки система орієнтована на дистанційне керування оператором, остаточне рішення щодо зміни напрямку руху або зупинки приймає користувач на основі отриманого відеозображення та службової інформації.

Застосування елементів теорії автоматичного управління дозволяє описати розроблену систему не тільки як набір апаратних і програмних компонентів, а як систему керування з визначеними входами, виходами, об’єктом керування, керуючим впливом, збуреннями та зворотною інформацією.

3 АПАРАТНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вибір апаратної платформи керування

Апаратна платформа керування визначає, які функції може виконувати мобільна система: обробляти команди руху, працювати з датчиками, керувати виконавчими пристроями, приймати дані з камери та передавати інформацію користувачу. Для розроблення мобільної платформи можна розглядати як мікроконтролерні плати, так і одноплатні комп'ютери. У цьому випадку важливо враховувати не тільки кількість входів і виходів, а й можливість роботи з камерою, мережею та серверним програмним забезпеченням.

Arduino UNO є мікроконтролерною платою на базі ATmega328P. Вона має 14 цифрових входів/виходів, 6 аналогових входів, 32 КБ Flash-пам'яті, 2 КБ SRAM та працює з тактовою частотою 16 МГц [24]. Така плата добре підходить для простого керування виконавчими пристроями, роботи з датчиками та реалізації базової логіки. Водночас Arduino UNO Rev3 не має вбудованого бездротового зв'язку, має обмежений обсяг пам'яті та не призначена для задач, пов'язаних із передаванням відеозображення або запуском Web-сервера. Тому для системи, у якій потрібно поєднати керування рухом, відео та Web-інтерфейс, можливостей Arduino UNO недостатньо [24], [28].

ESP32 є мікроконтролером з інтегрованими засобами Wi-Fi та Bluetooth, тому він краще підходить для мережевих і IoT-пристроїв, ніж класична Arduino UNO [25]. Окремі модулі на базі ESP32-CAM можуть використовуватися для задач потокового відео з мобільного робота, що розглядається у роботі [27]. Також існують рішення, у яких ESP32-CAM застосовується разом з Web-сторінкою керування мобільним роботом через протокол MQTT [28]. Водночас ESP32 залишається мікроконтролерною платформою, тому для системи, у якій потрібно одночасно реалізувати Web-інтерфейс, передавання відеозображення, керування рухом і додаткову логіку роботи, доцільніше використовувати продуктивнішу платформу з операційною системою.

Raspberry Pi Zero 2 WH належить до одноплатних комп'ютерів. За офіційними характеристиками він має чотириядерний 64-бітний процесор Arm Cortex-A53 з частотою 1 ГГц, 512 МБ SDRAM, Wi-Fi, Bluetooth, microSD, 40-контактний GPIO та CSI-2 роз'єм для підключення камери [26]. Крім того, Raspberry Pi може працювати під керуванням Raspberry Pi OS, яка є офіційною операційною системою для комп'ютерів Raspberry Pi, базується на Debian Linux та підтримує встановлення програмних пакетів [29]. Це дозволяє використовувати Raspberry Pi не тільки для роботи з входами та виходами, а й для запуску серверної частини, роботи з камерою, мережевими службами та бібліотеками для обробки даних.

Для порівняння розглянутих апаратних платформ основні можливості та обмеження наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння апаратних платформ керування

Платформа	Основні можливості	Основні обмеження
Arduino UNO	Просте керування входами/виходами, робота з датчиками та виконавчими пристроями, доступність і простота програмування	Немає вбудованого Wi-Fi у версії UNO Rev3, малий обсяг пам'яті, неможливість повноцінної роботи з відеопотоком і Web-сервером
ESP32	Вбудований Wi-Fi і Bluetooth, можливість мережевої взаємодії, придатність для IoT-пристроїв і простих вбудованих систем	Обмежені ресурси для одночасної роботи з відеопотоком, Web-інтерфейсом, керуванням рухом і додатковою логікою системи
Raspberry Pi Zero 2 W	Одноплатний комп'ютер з Linux, Wi-Fi, Bluetooth, GPIO, можливість запуску серверного програмного забезпечення	Більше енергоспоживання порівняно з простими мікроконтролерами, потреба у налаштуванні операційної системи

З урахуванням поставленої задачі найбільш придатною апаратною платформою є Raspberry Pi Zero 2 WH. На відміну від Arduino UNO, він має достатні ресурси для роботи з мережею, камерою та серверною частиною без використання додаткового основного контролера. Також у порівнянні з ESP32 Raspberry Pi краще підходить для системи, у якій потрібно одночасно реалізувати Web-інтерфейс, передавання відеозображення, обробку команд керування та взаємодію з периферійними пристроями. Тому Raspberry Pi Zero 2 WH обрано як основу системи керування мобільною платформою [26], [29].

3.2 Підключення апаратних компонентів мобільної платформи

Після визначення структури програмно-апаратної системи необхідно виконати підключення її апаратних компонентів. Апаратна частина мобільної платформи складається з чотириколісного шасі з DC-двигунами, одноплатного комп’ютера Raspberry Pi Zero 2 WH, драйвера двигунів L298N, камери Raspberry Pi NoIR Camera v2, інерціального модуля MPU-9250 та джерел живлення. Raspberry Pi використовується як центральний керуючий модуль, до якого підключаються компоненти для керування рухом, отримання відеозображення та зчитування додаткових даних про стан платформи.

3.2.1 Загальне компонування апаратної частини

Мобільна платформа (рис. 3.1) побудована на базі чотириколісного шасі з DC-двигунами. На корпусі платформи розміщено керуючий модуль Raspberry Pi, драйвер двигунів L298N, макетну плату для підключення допоміжних ліній та камеру. Таке компонування дозволяє розмістити основні елементи системи без зміни конструкції шасі та забезпечити доступ до дротів і роз’ємів під час налаштування.

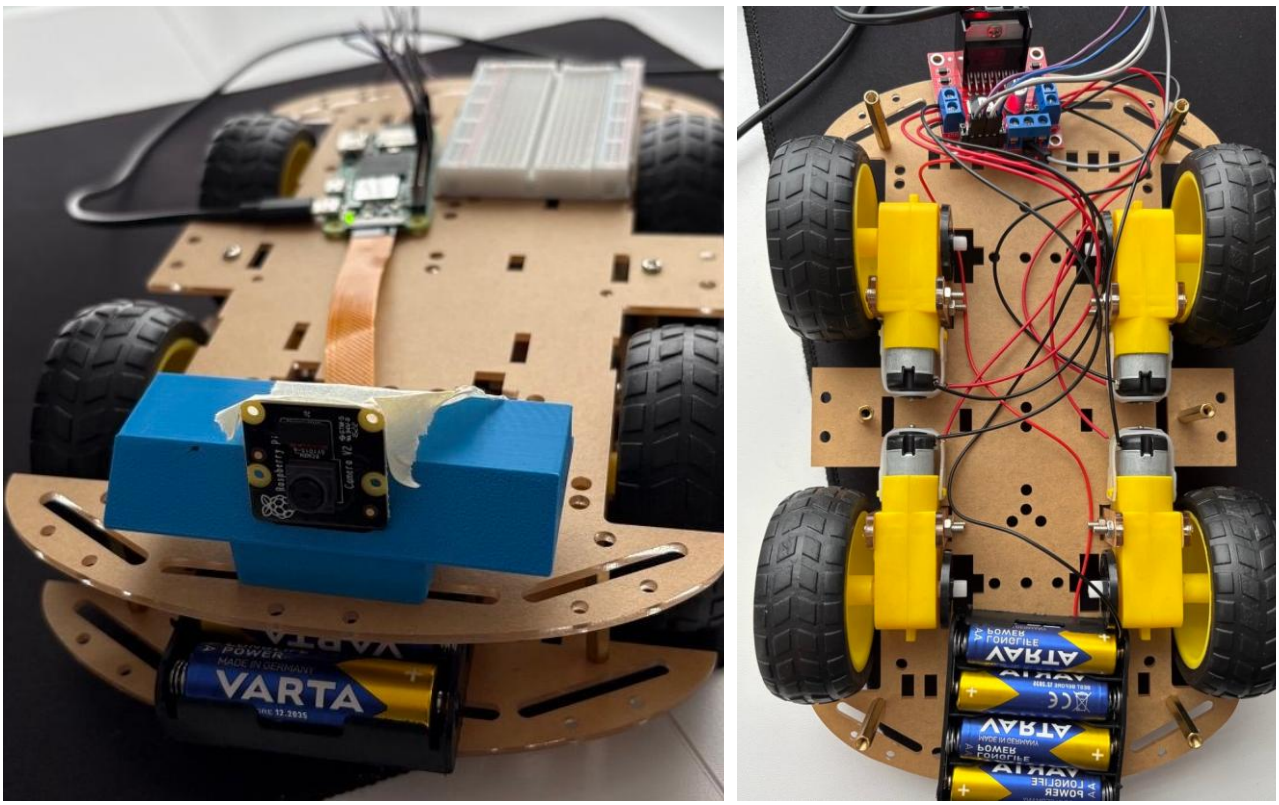


Рисунок 3.1 – Загальний вигляд мобільної платформи

Основним керуючим модулем системи є Raspberry Pi Zero 2 WH (рис. 3.2). До нього підключаються сигнальні лінії драйвера двигунів, камера через CSI-інтерфейс та інерціальний модуль через I2C. Raspberry Pi також виконує запуск серверної частини системи, обробляє команди користувача та передає сигнали керування до виконавчих елементів.

Для підключення периферійних компонентів використовуються виводи GPIO Raspberry Pi (рис. 3.3). Частина виводів застосовується для передавання логічних сигналів на драйвер двигунів, а окремі виводи використовуються для інтерфейсу I2C. При підключенні компонентів важливо враховувати нумерацію GPIO, оскільки фізичний номер контакту на роз’ємі та номер GPIO у програмному коді не є однаковими.



Рисунок 3.2 – Розміщення Raspberry Pi на мобільній платформі

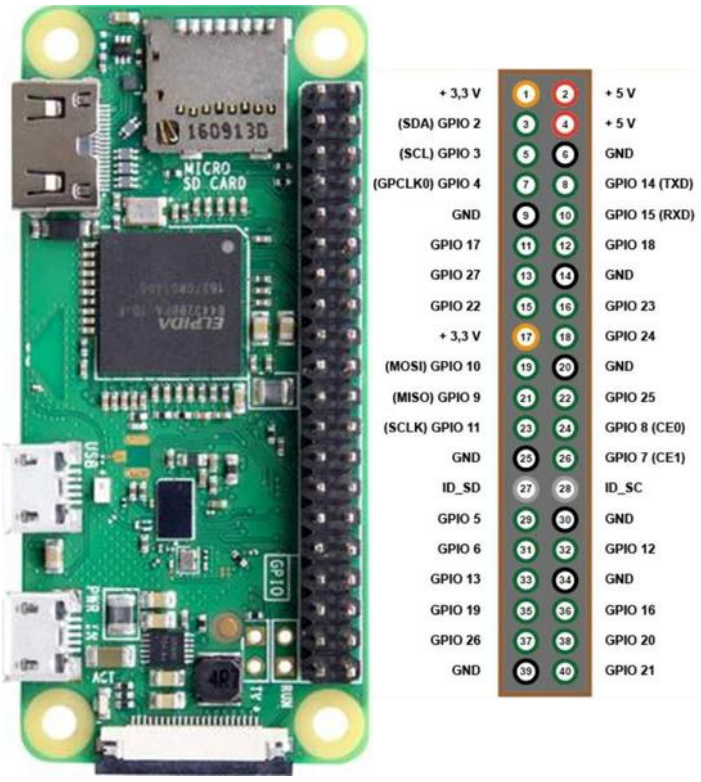


Рисунок 3.3 – Призначення виводів GPIO Raspberry Pi Zero 2 WH

3.2.2 Підключення драйвера двигунів L298N

Для керування DC-двигунами мобільної платформи використовується драйвер L298N. Його застосування необхідне тому, що виводи GPIO Raspberry Pi призначені для формування логічних сигналів, але не можуть безпосередньо живити двигуни. Драйвер L298N виконує роль проміжного елемента між Raspberry Pi та двигунами: він приймає логічні сигнали керування з GPIO та подає живлення на відповідні виходи двигунів.

Модуль L298N має дві групи виходів для підключення двигунів: вихід A та вихід B. Також на модулі є входи керування IN1-IN4, контакти увімкнення каналів ENA та ENB, вхід живлення двигунів, контакт GND та контакт 5 В логіки. Основні контакти драйвера L298N показано на рис. 3.4.

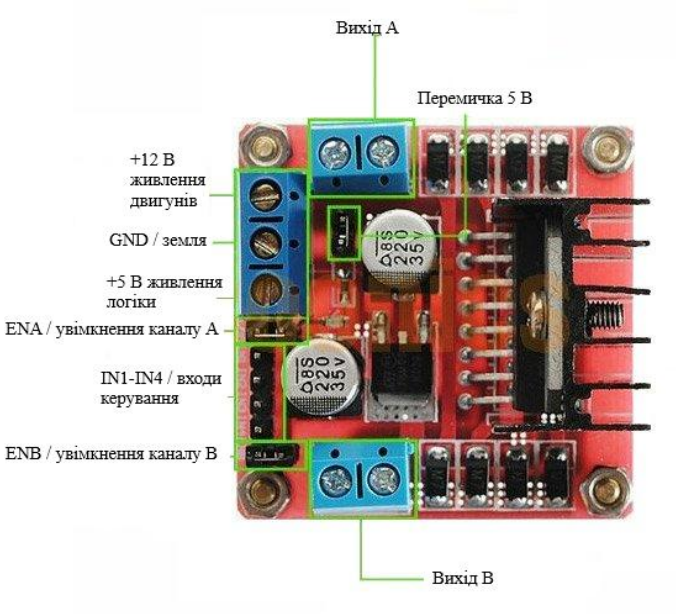


Рисунок 3.4 – Основні контакти драйвера двигунів L298N

У розроблюваній платформі вихід A драйвера використовується для підключення однієї пари двигунів, а вихід B – для підключення іншої пари двигунів. Сигнальні входи IN1-IN4 підключаються до GPIO Raspberry Pi. Через ці входи серверна частина системи формує команди руху: рух вперед, рух назад, поворот ліворуч, поворот праворуч та зупинку платформи.

Підключення драйвера L298N до Raspberry Pi наведено у таблиці 3.2.

Таблиця 3.2 – Підключення драйвера L298N до Raspberry Pi

Контакт L298N	Підключення	Призначення
IN1	GPIO17 Raspberry Pi	Керування напрямком обертання першої групи двигунів
IN2	GPIO27 Raspberry Pi	Керування напрямком обертання першої групи двигунів
IN3	GPIO22 Raspberry Pi	Керування напрямком обертання другої групи двигунів
IN4	GPIO23 Raspberry Pi	Керування напрямком обертання другої групи двигунів
ENA	Перемичка або активний рівень	Увімкнення каналу A
ENB	Перемичка або активний рівень	Увімкнення каналу B
Вихід A	Перша пара DC-двигунів	Живлення та керування першою парою двигунів
Вихід B	Друга пара DC-двигунів	Живлення та керування другою парою двигунів
+12 В живлення двигунів	Плюс батарейного блока 4xAA	Живлення двигунів через драйвер
GND	Мінус батарейного блока та GND Raspberry Pi	Спільна земля живлення і логічних сигналів

Позначення "+12 В" на модулі L298N використовується як вхід живлення двигунів, на цей вхід подається живлення від батарейного блока 4xAA. Raspberry Pi при цьому живиться окремо через власний роз’єм живлення. Важливо, що мінус батарейного блока, GND драйвера L298N та GND Raspberry Pi мають бути з’єднані між собою. Це потрібно для того, щоб логічні сигнали з Raspberry Pi

коректно розпізнавалися драйвером. На рисунку 3.5 можна побачити приклад підключення драйвера L298N до апаратної частини платформи.

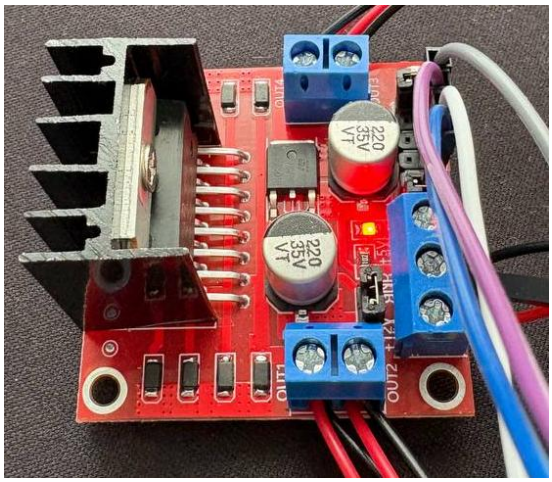


Рисунок 3.5 – Підключення драйвера L298N до апаратної частини платформи

Під час підключення двигунів необхідно враховувати конструкцію чотириколісного шасі. Двигуни з лівого та правого боку платформи встановлені дзеркально один відносно одного (рис 3.6). Через це однакове електричне підключення різних сторін може призвести до того, що колеса будуть обертатися у протилежних напрямах. Тому під час монтажу та програмної реалізації руху потрібно перевіряти фактичний напрям обертання кожної пари двигунів.

Для руху вперед обидві сторони платформи повинні створювати переміщення в одному напрямку. Якщо після подання команди руху одна сторона платформи обертається у протилежний бік, необхідно змінити порядок підключення двигуна до виходу драйвера або скоригувати логіку керування у програмному коді. У розроблюваній системі цей момент враховується у функціях керування рухом, де для кожної команди задається відповідна комбінація логічних рівнів на входах IN1-IN4.

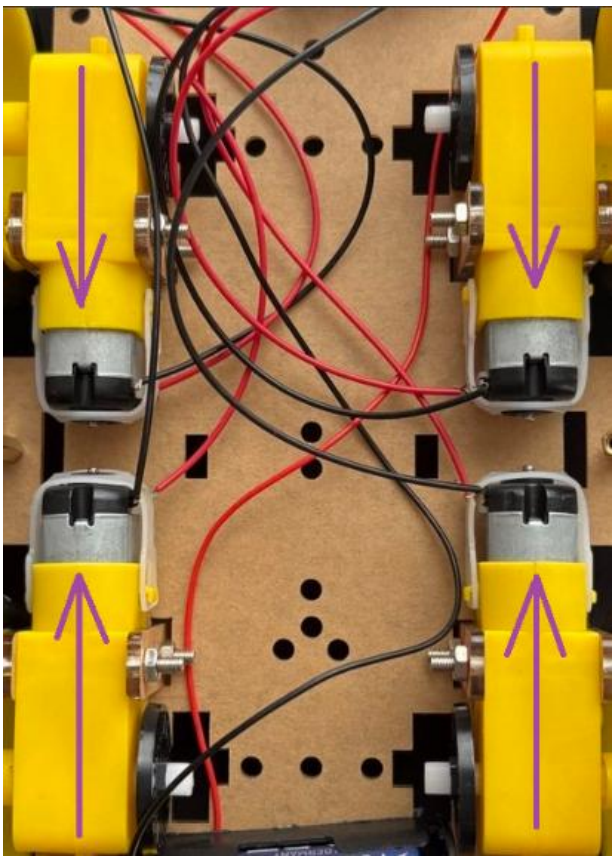


Рисунок 3.6 – Розміщення DC-двигунів на чотириколісному шасі

Таким чином, драйвер L298N забезпечує зв’язок між керуючим модулем Raspberry Pi та двигунами мобільної платформи. Raspberry Pi формує логічні сигнали керування, драйвер комутує живлення двигунів, а батарейний блок забезпечує необхідне живлення виконавчої частини. Окрему увагу під час підключення потрібно приділяти спільній землі, правильному підключенню входів IN1-IN4 та перевірці напрямку обертання двигунів.

3.2.3 Підключення камери Raspberry Pi NoIR Camera v2

Для отримання відеозображення у системі використовується камера Raspberry Pi NoIR Camera v2 (рис. 3.7). Вона встановлюється у передній частині мобільної платформи, щоб напрям огляду збігався з основним напрямом руху. Таке розміщення дозволяє користувачу бачити простір перед платформою під час дистанційного керування.



Рисунок 3.7 – Розміщення камери Raspberry Pi NoIR Camera v2 на мобільній платформі

Камера підключається до Raspberry Pi через CSI-інтерфейс за допомогою гнучкого шлейфа (рис 3.8). Такий спосіб підключення є зручним для мобільної платформи, оскільки камера не займає USB-порт і підключається безпосередньо до плати. Один кінець шлейфа підключається до роз'єму камери, а другий – до CSI-роз'єму Raspberry Pi.

Під час підключення потрібно враховувати, що Raspberry Pi Zero 2 WH має інший розмір роз'єму камери, ніж повнорозмірні моделі Raspberry Pi. Через це стандартний шлейф, який часто постачається з камерою, не підходить безпосередньо до Raspberry Pi Zero. Для підключення використовується окремий шлейф, сумісний саме з Raspberry Pi Zero 2 WH.

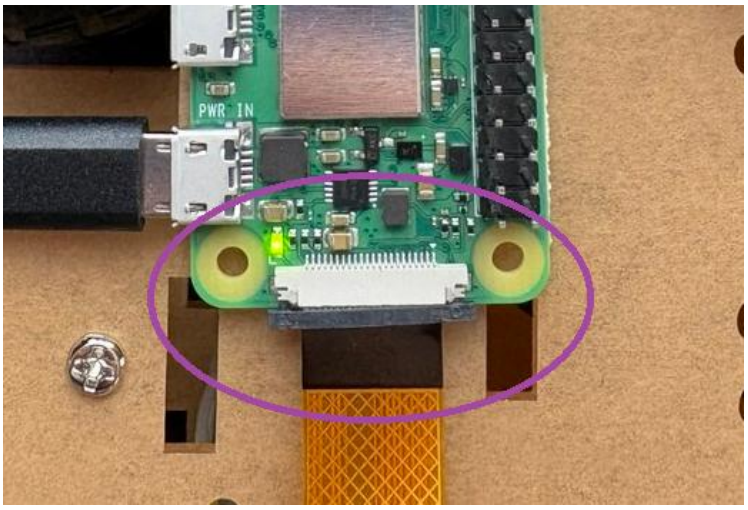


Рисунок 3.8 – Підключення шлейфа камери до Raspberry Pi

У програмній частині камера використовується для передавання відеозображення у Web-інтерфейс, створення знімків та роботи режиму виявлення руху. Відеопотік формується на стороні Raspberry Pi та передається користувачу через серверну частину. Завдяки цьому користувач може спостерігати за простором перед платформою без встановлення додаткового програмного забезпечення на клієнтському пристрої.

Оскільки використовується NoIR-версія камери, у звичайному режимі зображення може мати фіолетовий або рожевий відтінок. Це пов’язано з особливістю NoIR-камер, у яких відсутній інфрачервоний фільтр. Така особливість не заважає виконанню основної задачі, оскільки камера використовується для дистанційного спостереження, а не для точної передачі кольорів.

У Web-інтерфейсі передбачено декілька режимів відображення відеозображення – звичайний, нічний та монохромний режим. Звичайний режим використовується для базового перегляду зображення з камери. Нічний режим призначений для покращення видимості в умовах недостатнього освітлення. Монохромний режим дозволяє відображати зображення у чорно-білому вигляді, що може бути зручним для кращого візуального сприйняття контурів об’єктів.

Отже, камера Raspberry Pi NoIR Camera v2 забезпечує отримання відеозображення з мобільної платформи та є одним з основних елементів системи дистанційного спостереження. Її підключення через CSI-шлейф дозволяє безпосередньо інтегрувати камеру з Raspberry Pi, а підтримка декількох режимів відображення розширює можливості роботи Web-інтерфейсу.

3.2.4 Підключення інерціального модуля MPU-9250

Для отримання додаткової інформації про стан мобільної платформи у системі передбачено використання інерціального модуля MPU-9250. Цей модуль поєднує акселерометр, гіроскоп і магнітометр, що дозволяє отримувати дані про прискорення, кутову швидкість та орієнтацію платформи [32]. У межах розроблюваної системи ці дані можуть використовуватися для відображення

стану платформи у Web-інтерфейсі.

Підключення MPU-9250 до Raspberry Pi виконується через інтерфейс I2C. Для цього використовуються чотири основні лінії: живлення, земля, лінія передавання даних SDA та лінія синхронізації SCL. Живлення модуля підключається до відповідного виводу Raspberry Pi, GND модуля з'єднується із землею Raspberry Pi, а SDA та SCL підключаються до виводів інтерфейсу I2C.

Приклад підключення інерціального модуля MPU-9250 наведено на рисунку 3.9.

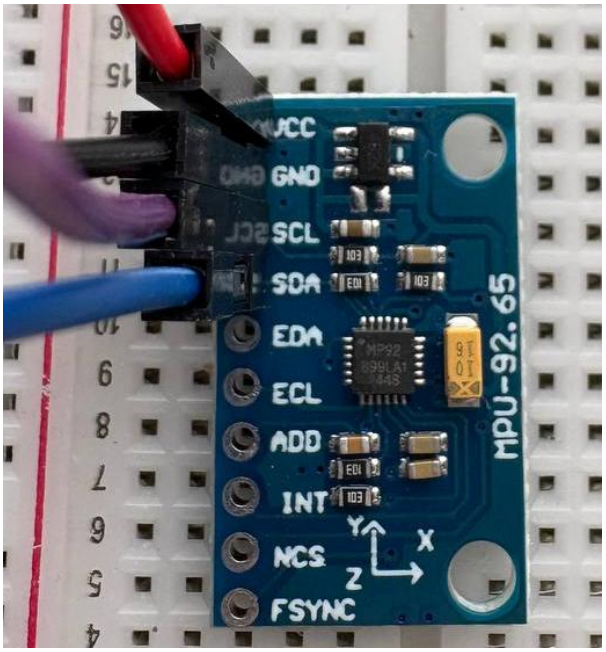


Рисунок 3.9 – Підключення інерціального модуля MPU-9250

Для коректної роботи модуля важливо правильно підключити лінії SDA та SCL, оскільки саме через них виконується обмін даними між Raspberry Pi та датчиком. Також необхідно враховувати, що інтерфейс I2C має бути активований у налаштуваннях Raspberry Pi OS. Після підключення датчика його наявність можна перевірити за допомогою сканування I2C-шини. Якщо модуль підключений коректно, він повинен визначатися за відповідною адресою на шині.

У програмній частині системи передбачено зчитування даних MPU-9250 та передавання їх до Web-інтерфейсу. Цей модуль може відображати дані

акселерометра, гіроскопа та магнітометра, а також розраховані параметри орієнтації платформи, зокрема нахил, крен та напрям. Це дозволяє користувачу отримувати додаткову інформацію про положення та стан мобільної платформи.

Окремо у програмній частині передбачено обробку ситуації, коли інерціальний модуль не передає дані або не визначається на шині I2C. У такому випадку система не завершує роботу аварійно, а відображає у Web-інтерфейсі стан Offline та повідомлення про відсутність підключення модуля. Це дозволяє зберегти працездатність основних функцій платформи, зокрема керування рухом, передавання відеозображення, створення знімків та роботу режиму виявлення руху.

3.2.5 Живлення апаратних компонентів

Живлення апаратної частини мобільної платформи розділено на дві основні частини: живлення керуючого модуля Raspberry Pi та живлення двигунів. Такий поділ використовується тому, що двигуни під час роботи споживають більший струм і можуть створювати просідання напруги, які небажано передавати безпосередньо на керуючий модуль.

Raspberry Pi Zero 2 WH живиться окремо через USB-роз’єм живлення. Це дозволяє забезпечити стабільну роботу операційної системи, серверної частини та підключених периферійних модулів. Двигуни мобільної платформи живляться від батарейного блока 4xAA, який підключається до входу живлення двигунів на драйвері L298N.

Важливою умовою правильної роботи системи є наявність спільної землі. Мінус батарейного блока, контакт GND драйвера L298N та контакт GND Raspberry Pi мають бути з’єднані між собою. Це необхідно для того, щоб драйвер L298N коректно розпізнавав логічні сигнали, які надходять з GPIO Raspberry Pi. Якщо спільну землю не підключити, команди керування можуть працювати нестабільно або взагалі не розпізнаватися драйвером.

Зовнішній вигляд батарейного блока наведено на рисунку 3.10.



Рисунок 3.10 – Живлення двигунів мобільної платформи від батарейного блока

Raspberry Pi не живиться від виходу 5 В драйвера L298N, оскільки такий спосіб може бути нестабільним для роботи одноплатного комп’ютера. У системі Raspberry Pi має окреме живлення, а драйвер L298N використовується для живлення та керування двигунами. Такий підхід зменшує ризик перезавантаження Raspberry Pi під час запуску або різкої зміни напрямку обертання двигунів. Окреме живлення керуючого модуля Raspberry Pi наведено на рисунку 3.11.



Рисунок 3.11 – Живлення керуючого модуля Raspberry Pi

Недоліком використаного варіанта живлення є наявність зовнішнього кабелю живлення Raspberry Pi. Через це платформа не є повністю автономною під час роботи. Для усунення цього недоліку можна використовувати компактний powerbank або спеціальний акумуляторний модуль для Raspberry Pi Zero, який встановлюється разом із платою та забезпечує її живлення без зовнішнього кабелю. Водночас під час вибору такого рішення потрібно враховувати масу акумулятора, оскільки невелике шасі не призначене для перевезення важких джерел живлення.

Отже, Raspberry Pi отримує живлення окремо через USB, батарейний блок 4xAA подає живлення на драйвер L298N, драйвер L298N живить DC-двигуни, а GND усіх частин системи об’єднується в одну спільну землю. Це забезпечує розділення силового живлення двигунів і живлення керуючого модуля, але зберігає коректну передачу логічних сигналів між Raspberry Pi та драйвером двигунів.

3.2.6 Узагальнення підключення апаратних засобів

Після підключення основних апаратних компонентів мобільної платформи Raspberry Pi Zero 2 WH виконує роль центрального керуючого модуля. До нього підключаються сигнальні лінії драйвера двигунів, камера та інерціальний модуль. Драйвер L298N використовується для керування DC-двигунами, а живлення виконавчої частини подається окремо від батарейного блока. Камера забезпечує отримання відеозображення, а MPU-9250 використовується як додатковий сенсорний модуль для отримання даних про стан платформи.

Основні підключення апаратних компонентів наведено у таблиці 3.3.

Таблиця 3.3 – Узагальнення підключення апаратних компонентів мобільної платформи

Компонент	Підключення	Призначення
Батарейний блок 4xAA	Підключений до входу живлення двигунів на L298N	Живлення виконавчої частини платформи

Продовження таблиці 3.3

Компонент	Підключення	Призначення
Драйвер L298N	Сигнальні входи IN1-IN4 підключені до GPIO Raspberry Pi	Керування напрямком обертання DC-двигунів
ENA, ENB драйвера L298N	Активовані для роботи каналів A та B	Увімкнення каналів керування двигунами
DC-двигуни	Підключені до виходів A та B драйвера L298N	Забезпечення руху мобільної платформи
GND Raspberry Pi, L298N та батарейного блока	Об’єднані між собою	Формування спільної землі для коректної передачі логічних сигналів
Raspberry Pi NoIR Camera v2	Підключена до Raspberry Pi через CSI-шлейф	Отримання відеозображення та створення знімків
MPU-9250	Підключення через I2C: VCC, GND, SDA, SCL	Отримання даних про прискорення, кутову швидкість, магнітне поле та орієнтацію платформи
Живлення Raspberry Pi	Окреме живлення через USB-роз’єм	Стабільна робота керуючого модуля та серверної частини
Raspberry Pi Zero 2 WH	Центральний керуючий модуль системи	Обробка команд, запуск серверної частини, робота з камерою, GPIO та периферійними модулями

Загальне підключення апаратних засобів побудовано таким чином, щоб розділити силову частину та частину керування. Двигуни отримують живлення через драйвер L298N від батарейного блока, а Raspberry Pi Zero 2 WH живиться

окремо та формує тільки логічні сигнали керування. Такий підхід дозволяє зменшити вплив роботи двигунів на стабільність керуючого модуля.
Окрему увагу під час монтажу було приділено підключенню спільної землі, правильній нумерації GPIO, підключенню двигунів до відповідних виходів драйвера та сумісності шлейфа камери з Raspberry Pi Zero 2 WH.

3.3 Реалізація серверної частини

Серверна частина програмно-апаратної системи реалізована мовою Python із використанням фреймворку Flask. Вона виконує роль проміжної ланки між Web-інтерфейсом користувача та апаратними компонентами мобільної платформи. Через серверну частину Web-інтерфейс отримує відеопотік, стан системи, журнал подій, дані додаткових модулів, а також передає команди керування рухом.

Серверна частина працює безпосередньо на Raspberry Pi Zero 2 WH. Завдяки цьому вона має доступ до апаратних інтерфейсів плати та може взаємодіяти з GPIO, камерою та I2C-модулем. Команди, які надходять від користувача через Web-інтерфейс, обробляються Flask API, після чого відповідні функції серверної частини керують драйвером двигунів, камерою, модулем виявлення руху або сенсорним модулем.

3.3.1 Структура серверної частини

Серверна частина має модульну структуру. Основна логіка API розміщена у файлі app.py, а робота з окремими частинами системи винесена в окремі програмні модулі. Такий підхід спрощує супровід коду, оскільки керування двигунами, робота з камерою, обробка даних IMU, виявлення руху та автоматична зупинка реалізовані окремо один від одного.

Структуру файлів серверної частини наведено у таблиці 3.4.

Таблиця 3.4 – Структура файлів серверної частини

Файл	Призначення
app.py	Основний файл Flask-додатка. Містить API-маршрути, запуск серверної частини, обробку запитів від Web-інтерфейсу та ведення журналу подій
config.py	Файл налаштувань системи. Містить параметри запуску сервера, номери GPIO, параметри камери, motion detection, IMU та журналу подій
motor.py	Модуль керування рухом платформи. Формує логічні сигнали GPIO для драйвера двигунів L298N
camera.py	Модуль роботи з камерою. Запуск камери, створення знімків та зміну режиму зображення
motion_detector.py	Модуль виявлення руху. Аналізує кадри з камери та визначає наявність руху за зміною зображення
imu.py	Модуль роботи з інерціальним сенсором MPU-9250. Виконує зчитування даних через I2C та обробляє ситуацію відсутності відповіді від модуля
watchdog.py	Модуль автоматичної зупинки. Зупиняє двигуни, якщо протягом визначеного часу не надходять актуальні команди керування
.env	Файл середовищних змінних, який може використовуватися для винесення окремих параметрів налаштування за межі програмного коду
requirements	Файл із переліком Python-залежностей, необхідних для запуску серверної частини

У файлі app.py створюється Flask-додаток, підключається CORS для взаємодії з клієнтською частиною, запускається watchdog, створюється фоновий потік для перевірки руху та описуються API-маршрути. Саме цей файл є центральною точкою серверної частини, через яку Web-інтерфейс взаємодіє з

іншими модулями.

Файл config.py використовується для зберігання основних параметрів системи. До них належать адреса та порт сервера, номери GPIO для керування драйвером L298N, час автоматичної зупинки, параметри камери, параметри виявлення руху, налаштування I2C та обмеження кількості повідомлень у журналі подій. Винесення цих значень в окремий файл дозволяє змінювати налаштування системи без редагування основної логіки роботи серверної частини.

Модуль motor.py відповідає за формування сигналів керування рухом. Він використовує GPIO Raspberry Pi для передавання логічних рівнів на входи IN1-IN4 драйвера L298N. У цьому модулі реалізовано функції руху вперед, назад, повороту ліворуч, повороту праворуч, зупинки та очищення стану GPIO після завершення роботи програми.

Модуль camera.py забезпечує роботу з камерою Raspberry Pi NoIR Camera v2. У ньому реалізовано запуск камери, формування відеопотоку, збереження знімка у папку snapshots та перемикання режимів зображення. Завдяки цьому Web-інтерфейс може відображати Live Camera, змінювати режим камери та отримувати окремий snapshot.

Модуль motion_detector.py використовується для виявлення руху на основі кадрів з камери. Він отримує актуальний кадр, перетворює його у відтінки сірого, порівнює з попереднім кадром і визначає, чи перевищує частка змінених пікселів заданий поріг. Якщо зміни достатні, система встановлює стан Motion Detected.

Модуль imu.py відповідає за роботу з інерціальним модулем MPU-9250. Він виконує підключення до I2C-шини, зчитує дані акселерометра, гіроскопа та магнітометра, а також формує відповідь для Web-інтерфейсу.

Модуль watchdog.py реалізує механізм автоматичної зупинки двигунів. Він працює у фоновому режимі та контролює час останньої команди руху. Якщо нові команди не надходять протягом заданого інтервалу, модуль викликає зупинку двигунів. Це підвищує безпечність роботи платформи під час дистанційного керування.

Серверна частина побудована за модульним принципом: app.py приймає запити від Web-інтерфейсу, а окремі модулі виконують роботу з конкретними апаратними або програмними функціями системи. Така структура дозволяє окремо описувати, перевіряти та змінювати логіку керування рухом, роботи з камерою, виявлення руху, обробки IMU та автоматичної зупинки.

3.3.2 Налаштування параметрів системи

Для зручного налаштування серверної частини основні параметри системи винесено в окремий файл config.py. Це дозволяє не змінювати логіку роботи основних модулів під час коригування номера порту, GPIO-виводів, параметрів камери, часу автоматичної зупинки, параметрів виявлення руху або налаштувань інерціального модуля.

У файлі config.py використовується завантаження змінних середовища за допомогою python-dotenv. Завдяки цьому окремі параметри можуть задаватися через файл .env. Якщо параметр не заданий у середовищі, у config.py використовується стандартне значення, визначене у програмному коді. Такий підхід є зручним для тестування системи та перенесення її між різними умовами запуску.

Фрагмент програмного коду з основними параметрами налаштування серверної частини наведено у додатку А у лістингу А.1.

У лістингу А.1 визначено параметри запуску сервера, GPIO-виводи Raspberry Pi Zero 2 WH для передавання логічних сигналів на входи IN1-IN4 драйвера L298N, параметри камери, параметри виявлення руху, інтервал повторної перевірки IMU та обмеження кількості повідомлень у журналі подій. Також задано параметр WATCHDOG_TIMEOUT, який визначає інтервал часу, після якого система автоматично зупиняє двигуни, якщо нові актуальні команди керування не надходять.

Основні параметри, винесені у файл config.py, наведено у таблиці 3.5.

Таблиця 3.5 – Основні параметри налаштування серверної частини

Група параметрів	Параметри	Призначення
Параметри запуску сервера	HOST, PORT	Визначають адресу та порт запуску Flask-сервера
GPIO для драйвера L298N	IN1, IN2, IN3, IN4	Визначають GPIO-виводи Raspberry Pi, підключені до входів керування драйвера двигунів
Автоматична зупинка	WATCHDOG_TIMEOUT	Визначає час очікування актуальної команди руху перед автоматичною зупинкою двигунів
Параметри камери	CAMERA_WIDTH, CAMERA_HEIGHT, CAMERA_FRAME_RATE, CAMERA_DEFAULT_MODE	Визначають роздільну здатність відеопотоку, частоту кадрів та початковий режим камери
Параметри виявлення руху	MOTION_DIFF_THRESHOLD, MOTION_PIXEL_RATIO, MOTION_COOLDOWN, MOTION_INTERVAL, MOTION_SAMPLE_STEP	Визначають чутливість алгоритму виявлення руху, інтервал перевірки кадрів та частоту оновлення стану

Продовження таблиці 3.5

Група параметрів	Параметри	Призначення
Журнал подій	MAX_LOGS	Обмежує максимальну кількість повідомлень, які зберігаються у журналі подій

Параметри HOST і PORT визначають, на якій адресі та порту запускається серверна частина. Для доступу до сервера з інших пристроїв у локальній мережі використовується значення HOST, яке дозволяє приймати підключення не лише з самого Raspberry Pi, а й з клієнтського пристрою, на якому відкрито Web-інтерфейс.

Параметри IN1, IN2, IN3 та IN4 відповідають GPIO-виводам, через які Raspberry Pi Zero 2 WH керує драйвером L298N. Саме ці значення використовуються у модулі motor.py для формування комбінацій логічних рівнів під час руху вперед, назад, повороту ліворуч, повороту праворуч та зупинки.

Параметри камери визначають розмір кадру, частоту оновлення та початковий режим роботи камери. Завдяки винесенню цих значень у config.py можна змінювати характеристики відеопотоку без редагування модуля camera.py.

Окрема група параметрів відповідає за режим виявлення руху. Значення MOTION_DIFF_THRESHOLD визначає поріг зміни яскравості пікселя, MOTION_PIXEL_RATIO задає мінімальну частку змінених пікселів для фіксації руху, а MOTION_SAMPLE_STEP дозволяє аналізувати не кожен піксель кадру, а вибірку зображення. Це зменшує навантаження на Raspberry Pi Zero 2 WH під час роботи з відеокадрами.

Файл .env використовується як додатковий спосіб винесення окремих параметрів за межі програмного коду. У поточній реалізації основні значення мають стандартні параметри у config.py, однак за потреби вони можуть бути перевизначені через змінні середовища. Це робить серверну частину більш гнучкою під час налаштування та запуску на Raspberry Pi.

Таким чином, файл config.py виконує роль централізованого місця зберігання параметрів серверної частини. Завдяки цьому основні модулі системи не містять жорстко заданих значень, а використовують налаштування, які можна змінити в одному місці.

3.3.3 Реалізація API-маршрутів Flask

Взаємодія між Web-інтерфейсом користувача та серверною частиною реалізована за допомогою API-маршрутів Flask. Кожен маршрут відповідає за окрему дію або за отримання певного стану системи. Завдяки цьому клієнтська частина не взаємодіє з апаратними компонентами безпосередньо, а надсилає HTTP-запити до серверної частини, яка вже виконує необхідні дії на Raspberry Pi Zero 2 WH.

У серверній частині використовуються GET- та POST-запити. GET-запити застосовуються для отримання даних від системи: стану сервера, відеопотоку, журналу подій, стану виявлення руху та даних IMU. POST-запити використовуються для передавання команд, які змінюють стан системи: керування рухом, зупинка платформи, створення знімка або зміна режиму камери. Основні API-маршрути серверної частини наведено на рисунку 3.12.

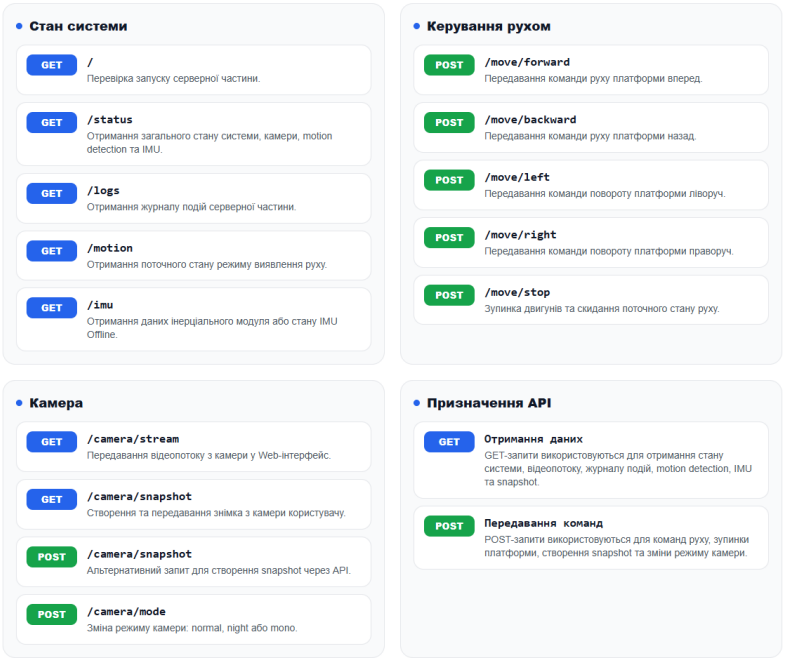


Рисунок 3.12 – API-маршрути серверної частини

Маршрут /status використовується для отримання загального стану системи. Він повертає інформацію про доступність backend, стан камери, поточний режим камери, доступні режими, стан виявлення руху та дані інерціального модуля. Приклад реалізації GET-запиту /status наведено нижче.

```
@app.route("/status", methods=["GET"])
def status():
    return jsonify({
        "status": "online",
        "camera_started": camera.is_camera_started(),
        "camera_mode": camera.get_camera_mode(),
        "camera_modes": camera.get_camera_modes(),
        "camera_crop": camera.get_camera_crop(),
        "motion": motion_detector.motion_detected(),
        "imu": imu.read(),
    })
```

У цьому прикладі серверна частина формує JSON-відповідь для Web-інтерфейсу. Значення "status": "online" показує, що backend працює. Поля camera_started, camera_mode та camera_modes описують стан камери. Поле motion передає поточний стан режиму виявлення руху, а поле imu містить дані інерціального модуля або інформацію про його недоступність.

Для керування рухом використовуються POST-запити. Наприклад, маршрут /move/forward відповідає за команду руху платформи вперед. Після отримання цього запиту серверна частина викликає функцію руху з модуля motor.py, оновлює watchdog та записує поточний стан руху до журналу подій.

```
@app.route("/move/forward", methods=["POST"])
def move_forward():
    motor.forward()
    watchdog.touch()
    set_move_state("forward")
    return jsonify({"status": "forward"})
```

У цьому фрагменті функція motor.forward() формує необхідну комбінацію GPIO-сигналів для драйвера L298N. Виклик watchdog.touch() оновлює час останньої актуальної команди руху, що потрібно для механізму автоматичної зупинки. Функція set_move_state("forward") використовується для фіксації стану руху та запису події до журналу. Після цього сервер повертає JSON-відповідь зі статусом виконаної команди.

Аналогічно реалізовані маршрути /move/backward, /move/left, /move/right та /move/stop. Кожен із них викликає відповідну функцію модуля motor.py, після чого Web-інтерфейс отримує відповідь про виконану дію. Такий підхід дозволяє розділити логіку обробки HTTP-запитів і логіку безпосереднього керування двигунами.

Окрему групу API-маршрутів становлять маршрути для роботи з камерою. Маршрут /camera/stream передає відеопотік у Web-інтерфейс, /camera/snapshot використовується для створення знімка, а /camera/mode дозволяє змінювати режим роботи камери. Це дає змогу керувати відеомодулем через браузер без додаткового програмного забезпечення на стороні користувача.

Також у серверній частині передбачено маршрути /logs, /motion та /imu. Маршрут /logs повертає журнал подій, /motion - поточний стан режиму виявлення руху, а /imu - дані інерціального модуля або стан його недоступності. Ці маршрути використовуються Web-інтерфейсом для оновлення службових блоків System Status, Event Log та IMU Sensor.

API-маршрути Flask забезпечують основну взаємодію між клієнтською частиною та серверною логікою. Web-інтерфейс формує запити, серверна частина обробляє їх, викликає відповідні модулі та повертає результат у вигляді JSON-відповіді, зображення або відеопотоку.

3.3.4 Реалізація керування рухом платформи

Керування рухом мобільної платформи реалізовано у модулі motor.py. Цей модуль відповідає за формування логічних сигналів на GPIO-виводах Raspberry Pi Zero 2 WH, які підключені до входів IN1-IN4 драйвера двигунів L298N. Таким

чином, серверна частина не керує двигунами безпосередньо, а формує керуючі сигнали для драйвера, який уже забезпечує подачу живлення на відповідні DC-двигуни.

У модулі motor.py використовуються GPIO-виводи, задані у файлі config.py. Після імпорту параметрів IN1, IN2, IN3 та IN4 виконується налаштування цих виводів як вихідних. Для роботи з GPIO використовується бібліотека RPi.GPIO. При цьому у коді передбачено перевірку доступності цієї бібліотеки. Якщо серверна частина запускається не на Raspberry Pi, а в середовищі без доступу до GPIO, замість реального керування виводами у консоль виводиться інформація про сформовані логічні сигнали. Це спрощує тестування окремих частин коду без підключення апаратної частини.

Основною допоміжною функцією модуля є _write(a, b, c, d). Вона приймає чотири логічні значення та передає їх відповідно на входи IN1-IN4 драйвера L298N. На основі цієї функції реалізовано окремі команди руху: forward(), backward(), left(), right() та stop().

Логіку формування сигналів керування наведено у таблиці 3.6.

Таблиця 3.6 – Формування логічних сигналів для керування рухом

Команда	IN1	IN2	IN3	IN4	Результат
forward()	1	0	1	0	Рух платформи вперед
backward()	0	1	0	1	Рух платформи назад
left()	1	0	0	1	Поворот платформи ліворуч
right()	0	1	1	0	Поворот платформи праворуч
stop()	0	0	0	0	Зупинка двигунів

Під час виконання команди руху серверна частина викликає відповідну функцію з модуля motor.py. Наприклад, при отриманні POST-запиту /move/forward викликається функція motor.forward(), після чого на входи драйвера L298N подається комбінація сигналів 1, 0, 1, 0. У результаті обидві групи двигунів обертаються таким чином, щоб платформа рухалась вперед.

Рух вперед і назад виконується тоді, коли обидві групи двигунів створюють переміщення платформи в одному напрямку. Для руху вперед на обидва канали драйвера подається однакова логіка напрямку, а для руху назад – протилежна. Через дзеркальне розміщення двигунів на лівій та правій стороні шасі фактичний напрям обертання коліс перевіряється під час налаштування, щоб обидві сторони платформи працювали узгоджено.

Поворот платформи реалізовано за диференціальним принципом керування. У цьому випадку одна група двигунів обертається в одному напрямку, а інша група – у протилежному. Такий принцип подібний до бортового керування у гусеничних платформах: окремий рульовий механізм не використовується, а зміна напрямку руху відбувається за рахунок різниці обертання лівої та правої сторони. Для чотириколісного шасі це дозволяє виконувати поворот на місці або з малим радіусом.

Окрему роль у керуванні рухом відіграє команда stop(). Вона встановлює нульові логічні рівні на всіх входах IN1-IN4, унаслідок чого двигуни припиняють обертання. Ця команда використовується не лише при натисканні кнопки зупинки, а й після відпускання кнопки керування у Web-інтерфейсі.

У Web-інтерфейсі керування рухом реалізовано таким чином, що платформа рухається тільки під час активного натискання кнопки або клавіші. Наприклад, при натисканні клавіші W надсилається команда руху вперед, а після відпускання клавіші надсилається команда зупинки. Серверна частина при цьому не зчитує стан клавіатури напряму, а лише обробляє HTTP-запити, які надходять від клієнтської частини.

Для підвищення безпечності роботи у системі також використовується механізм watchdog. Після кожної актуальної команди руху серверна частина оновлює час останньої команди. Якщо нові команди керування не надходять протягом заданого інтервалу, watchdog автоматично викликає зупинку двигунів. Це потрібно для ситуацій, коли Web-інтерфейс, браузер або мережеве з’єднання перестали передавати команди, і команда stop() не була отримана сервером.

Керування рухом у серверній частині побудовано як зв’язок між Web-

інтерфейсом, Flask API, модулем motor.py та драйвером L298N. Web-інтерфейс формує команди користувача, Flask API приймає ці команди, motor.py перетворює їх у комбінації GPIO-сигналів, а драйвер L298N забезпечує керування групами DC-двигунів.

3.3.5 Реалізація роботи з камерою

Робота з камерою у серверній частині реалізована у модулі camera.py. Цей модуль відповідає за запуск камери, отримання відеокадрів, формування відеопотоку для Web-інтерфейсу, створення знімків та зміну режиму зображення. Для взаємодії з камерою Raspberry Pi NoIR Camera v2 використовується бібліотека Picamera2, яка призначена для роботи з камерами Raspberry Pi у Python-середовищі [40].

Камера ініціалізується серверною частиною під час звернення до функцій роботи з відео. У модулі camera.py для цього використовується допоміжна функція _ensure_camera(). Вона перевіряє, чи створено об’єкт камери, виконує її налаштування та запускає запис кадрів у форматі JPEG. Завдяки цьому камера підключається до роботи тоді, коли серверній частині потрібно передати відеопотік або створити знімок.

Для збереження актуального кадру використовується клас StreamingOutput. Він приймає JPEG-кадр від камери та зберігає його у змінній frame. Після отримання нового кадру інші частини програми можуть використати його для передавання відеопотоку, створення snapshot або аналізу в модулі виявлення руху.

Передавання відеопотоку у Web-інтерфейс реалізовано через маршрут /camera/stream. Сервер повертає відповідь типу multipart/x-mixed-replace, у якій послідовно передаються JPEG-кадри. Завдяки цьому браузер може відображати зображення з камери як безперервний потік.

Приклад маршруту передавання відеопотоку наведено нижче.

```
@app.route("/camera/stream", methods=["GET"])
def camera_stream():
```

```
    return Response(
        camera.stream_generator(),
        mimetype="multipart/x-mixed-replace; boundary=FRAME"
    )
```

У цьому фрагменті функція camera.stream_generator() формує послідовність JPEG-кадрів, а Flask передає їх клієнтській частині як потокову відповідь. На стороні Web-інтерфейсу цей маршрут використовується для відображення блоку Live Camera.

Окрім відеопотоку, у системі реалізовано створення окремого знімка з камери. Для цього використовується маршрут /camera/snapshot, який може обробляти GET- та POST-запити. Після звернення до цього маршруту серверна частина викликає функцію camera.save_snapshot(), зберігає актуальний кадр у папку snapshots та повертає користувачу файл зображення.

Приклад реалізації маршруту створення знімка наведено нижче.

```
@app.route("/camera/snapshot", methods=["GET", "POST"])
def camera_snapshot():
    try:
        path = camera.save_snapshot()
        add_log(f"Manual snapshot saved: {path}")
        return send_file(path, mimetype="image/jpeg")
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

У цьому фрагменті після створення знімка подія записується до журналу подій за допомогою функції add_log(). Якщо кадр з камери недоступний або під час створення знімка виникає помилка, сервер повертає JSON-відповідь з описом помилки та кодом 500.

У модулі camera.py також реалізовано перемикання режимів зображення: normal, night та mono. Режим normal використовується для звичайного відображення зображення з камери. Режим night змінює параметри яскравості та контрастності для покращення видимості в умовах недостатнього освітлення.

Режим mono зменшує насиченість зображення та використовується для монохромного відображення відео.

Зміна режиму камери виконується через маршрут /camera/mode. Web-інтерфейс надсилає POST-запит із назвою потрібного режиму, після чого серверна частина перевіряє його коректність і застосовує відповідні параметри камери. Якщо вказаний режим не підтримується, сервер повертає повідомлення про помилку.

3.3.6 Реалізація виявлення руху та обробки даних IMU

Виявлення руху реалізовано у модулі motion_detector.py. У системі не використовується складна обробка зображень або розпізнавання об’єктів, оскільки для роботи платформи достатньо визначати сам факт помітної зміни кадру. Для цього використовується порівняння поточного кадру з попереднім.

JPEG-кадр, отриманий з камери, декодується за допомогою бібліотеки simplejpeg, яка надає функції для кодування та декодування JPEG-зображень у Python [41]. Після декодування кадр перетворюється у відтінки сірого, а потім порівнюється з попереднім кадром. Якщо частка пікселів, зміна яскравості яких перевищує заданий поріг, більша за встановлене значення, система фіксує рух.

Приклад основної логіки визначення руху наведено нижче.

```
diff = abs(current_gray - _previous_gray)
changed_ratio = (diff > MOTION_DIFF_THRESHOLD).mean()

if changed_ratio >= MOTION_PIXEL_RATIO:
    _motion = True
```

У цьому фрагменті diff містить різницю між поточним і попереднім кадром, а changed_ratio визначає частку пікселів, для яких ця різниця перевищує поріг MOTION_DIFF_THRESHOLD. Якщо отримане значення більше або дорівнює MOTION_PIXEL_RATIO, у системі встановлюється стан Motion Detected.

Обробка даних інерціального модуля реалізована у файлі imu.py. Серверна частина звертається до MPU-9250 через I2C та формує відповідь для Web-інтерфейсу. Дані IMU передаються через окремий API-маршрут /imu.

```
@app.route("/imu", methods=["GET"])
def get_imu():
    return jsonify({"imu": imu.read()})
```

Якщо MPU-9250 не відповідає або не визначається на шині I2C, серверна частина повертає стан недоступності модуля. Завдяки цьому помилка з додатковим сенсором не зупиняє роботу всієї системи, а у Web-інтерфейсі відображається стан IMU Offline.

3.3.7 Механізм автоматичної зупинки та журнал подій

Для підвищення безпечності керування у серверній частині реалізовано механізм автоматичної зупинки двигунів. Він розміщений у модулі watchdog.py та використовується для контролю надходження актуальних команд руху. Це потрібно для того, щоб платформа не продовжувала рух у випадку, якщо Web-інтерфейс, браузер або мережеве з’єднання перестали передавати команди.

Під час штатної роботи користувач натискає кнопку або клавішу керування у Web-інтерфейсі, після чого клієнтська частина надсилає відповідний POST-запит до серверної частини. Після отримання команди руху сервер викликає функцію з модуля motor.py та оновлює watchdog. Якщо користувач відпускає кнопку або клавішу, Web-інтерфейс надсилає команду stop, і двигуни зупиняються.

Механізм watchdog використовується як додаткова перевірка. Якщо нові команди руху не надходять протягом заданого інтервалу часу, watchdog автоматично викликає зупинку двигунів. Приклад перевірки часу останньої команди наведено нижче.

```
if not _stopped and time.time() - _last_command_time > WATCHDOG_TIMEOUT:
```

```
            _stopped = True
            should_stop = True

    if should_stop:
        stop()
```

У цьому фрагменті порівнюється поточний час із часом останньої отриманої команди руху. Якщо різниця перевищує значення WATCHDOG_TIMEOUT, система переходить у стан зупинки та викликає функцію stop(). Це дозволяє уникнути ситуації, коли платформа продовжує рух без актуальної команди користувача.

Окрім механізму автоматичної зупинки, у серверній частині реалізовано журнал подій. Він використовується для відображення важливих дій системи у Web-інтерфейсі. До журналу можуть додаватися повідомлення про запуск backend, зміну режиму камери, створення snapshot, команди руху, зупинку платформи та виявлення руху.

Для додавання подій використовується допоміжна функція add_log(). Вона формує повідомлення з часовою міткою та зберігає його у списку журналу. Кількість повідомлень обмежується параметром MAX_LOGS, щоб журнал не зростав безконтрольно під час тривалої роботи системи.

```
def add_log(message):
    logs.insert(0, {
        "time": datetime.now().strftime("%H:%M:%S"),
        "message": message,
    })
    del logs[MAX_LOGS:]
```

Журнал подій доступний через API-маршрут /logs. Web-інтерфейс використовує цей маршрут для оновлення блоку Event Log, у якому користувач може бачити останні дії системи. Це спрощує перевірку роботи окремих функцій і дозволяє швидко зрозуміти, які події відбувалися під час керування платформою.

Механізм watchdog та журнал подій доповнюють основну логіку серверної частини. Watchdog забезпечує автоматичну зупинку двигунів у випадку втрати актуальних команд, а журнал подій дає змогу відстежувати роботу системи під час дистанційного керування.

3.4 Реалізація клієнтської частини

Клієнтська частина системи реалізована на основі React.js. Вона відповідає за відображення Web-інтерфейсу користувача, передавання команд керування до серверної частини, отримання стану системи та відображення відеопотоку з камери. Через клієнтську частину користувач взаємодіє з мобільною платформою без прямого доступу до серверного коду або апаратних компонентів.

Для оформлення інтерфейсу використовується Tailwind CSS та додаткові стилі, описані у файлі index.css. У клієнтській частині передбачено світлий і темний режими відображення, блок відеопотоку, панель керування рухом, блок стану системи, блок IMU Sensor та журнал подій. Основна логіка інтерфейсу розподілена між окремими компонентами, API-файлом та користувацькими React-хуками.

3.4.1 Структура клієнтської частини

Клієнтська частина має компонентну структуру. Основні елементи інтерфейсу винесені в окремі React-компоненти, робота із серверною частиною реалізована в окремому API-файлі, а логіка керування рухом і оновлення стану системи винесена в користувацькі хуки. Такий підхід дозволяє розділити відображення сторінки, обробку дій користувача та взаємодію з Flask API.

Структуру файлів клієнтської частини наведено на рисунку 3.13.

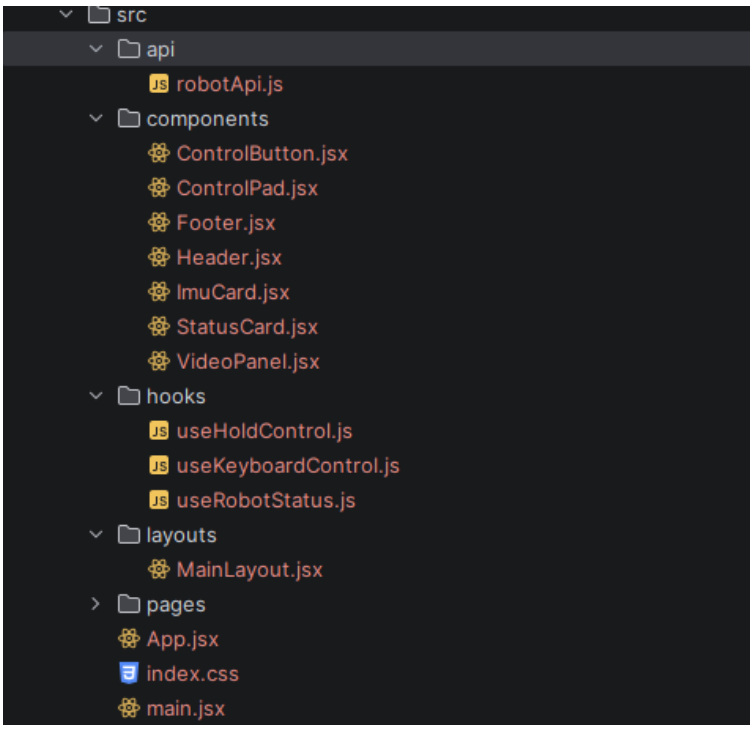


Рисунок 3.13 – Структура файлів клієнтської частини

Основні елементи структури клієнтської частини наведено у таблиці 3.7.

Таблиця 3.7 – Структура клієнтської частини

Елемент структури	Призначення
api	Містить файл robotApi.js, який відповідає за взаємодію Web-інтерфейсу з Flask API
components	Містить основні компоненти інтерфейсу: панель відео, панель керування рухом, блок стану системи, блок IMU, Header та Footer
hooks	Містить користувацькі React-хуки для утримання команди руху, обробки клавіатури та періодичного оновлення стану системи
layouts	Містить загальний макет сторінки MainLayout.jsx
pages	Містить основну сторінку DashboardPage.jsx, на якій розміщуються блоки керування та моніторингу

Продовження таблиці 3.7

Елемент структури	Призначення
App.jsx	Головний компонент застосунку, який об’єднує тему, хуки керування, макет і сторінку інтерфейсу
main.jsx	Точка входу клієнтської частини
index.css	Файл стилів, у якому підключено Tailwind CSS та описано додаткове оформлення інтерфейсу

Папка components є основною для побудови Web-інтерфейсу. У ній розміщено компоненти, що відповідають за окремі функціональні блоки сторінки. Компонент VideoPanel використовується для відображення відеопотоку з камери, перемикання режимів зображення та створення snapshot. Компонент ControlPad відповідає за панель керування рухом, а ControlButton використовується як окрема кнопка керування. Компонент StatusCard відображає стан backend, камери та режиму виявлення руху, а ImuCard показує стан інерціального модуля. Компоненти Header та Footer формують верхню і нижню частини інтерфейсу.

Папка hooks використовується для логіки, яка не пов’язана безпосередньо з розміткою сторінки. Хук useHoldControl відповідає за утримання активної команди руху, useKeyboardControl обробляє клавіші W, A, S, D та Space, а useRobotStatus періодично отримує стан системи та журнал подій від серверної частини. Завдяки цьому логіка керування не змішується з JSX-розміткою компонентів.

Файл robotApi.js розміщено у папці api. У ньому описано функції для звернення до Flask API: отримання стану системи, отримання журналу подій, передавання команд руху, зупинка платформи, формування посилань на відеопотік і snapshot. Це дозволяє не дублювати адреси API-маршрутів у різних компонентах.

Файл DashboardPage.jsx відповідає за основну сторінку Web-інтерфейсу. На ній об’єднуються блок відеопотоку, панель керування, картки стану системи, блок IMU та журнал подій. Файл MainLayout.jsx задає загальний макет сторінки, а App.jsx підключає тему, основну сторінку та хуки керування. Файл main.jsx використовується як точка входу React-застосунку.

Файл index.css містить підключення Tailwind CSS та додаткові стилі для оформлення інтерфейсу. У ньому описані кольори, панелі, кнопки, а також стилі для світлого і темного режимів. Завдяки цьому зовнішній вигляд інтерфейсу винесений окремо від логіки компонентів.

3.4.2 Взаємодія клієнтської частини із сервером

Взаємодія клієнтської частини із серверною частиною реалізована у файлі robotApi.js. У цьому файлі задається базова адреса Flask API та описуються функції для надсилання GET- і POST-запитів. Завдяки цьому компоненти Web-інтерфейсу не формують адреси API-маршрутів самостійно, а використовують готові функції з одного файлу.

Базова адреса серверної частини може задаватися через змінну середовища VITE_API_BASE_URL. Якщо така змінна не задана, адреса backend формується автоматично на основі поточного хоста та порту 5000. Це дозволяє запускати клієнтську частину без жорсткого зазначення IP-адреси Raspberry Pi у кожному компоненті.

Фрагмент коду взаємодії клієнтської частини з Flask API наведено нижче.

```
const rawApiBaseUrl = import.meta.env.VITE_API_BASE_URL ||
`${window.location.protocol}//${window.location.hostname}:5000`

const API_BASE_URL = rawApiBaseUrl.startsWith('http')
  ? rawApiBaseUrl.replace(/\/$/, '')
  : `http://${rawApiBaseUrl}`.replace(/\/$/, '')

const sendGet = async (path) => {
  const response = await fetch(`${API_BASE_URL}${path}`)
```

```
  if (!response.ok) {
    throw new Error(`GET ${path} failed`)
  }

  return response.json()
}

const sendPost = async (path, body, options = {}) => {
  const response = await fetch(`${API_BASE_URL}${path}`, {
    method: 'POST',
    headers: {
      ...(body ? { 'Content-Type': 'application/json' } : {}),
    },
    ...(body ? { body: JSON.stringify(body) } : {}),
    signal: options.signal,
  })

  if (!response.ok) {
    throw new Error(`POST ${path} failed`)
  }

  return response.json()
}
```

У цьому фрагменті функція sendGet() використовується для отримання даних від серверної частини, а функція sendPost() - для передавання команд або параметрів. Якщо сервер повертає помилку, у коді формується виняток, який може бути оброблений у відповідному компоненті або хуку.

Основні функції для роботи з API об’єднані в об’єкті robotApi. Через нього клієнтська частина отримує стан системи, журнал подій, стан motion detection, дані IMU, надсилає команди руху та формує посилання на відеопотік і snapshot.

```
export const robotApi = {
  moveForward: (options) => sendPost('/move/forward', undefined, options),
```

```
  moveBackward: (options) => sendPost('/move/backward', undefined, options),
  moveLeft: (options) => sendPost('/move/left', undefined, options),
  moveRight: (options) => sendPost('/move/right', undefined, options),
  stop: () => sendPost('/move/stop'),

  getStatus: () => sendGet('/status'),
  getLogs: () => sendGet('/logs'),
  setCameraMode: (mode) => sendPost('/camera/mode', { mode }),

  getStreamUrl: () => `${API_BASE_URL}/camera/stream`,
  getSnapshotUrl: () => `${API_BASE_URL}/camera/snapshot`,
}
```

GET-запити використовуються для отримання інформації від серверної частини. Наприклад, функція getStatus() звертається до маршруту /status, а getLogs() отримує журнал подій з маршруту /logs. Ці дані використовуються для оновлення блоків System Status, IMU Sensor та Event Log.

POST-запити використовуються для команд, які змінюють стан системи. Наприклад, функція moveForward() надсилає запит на маршрут /move/forward, а stop() – на маршрут /move/stop. Функція setCameraMode() передає на сервер назву обраного режиму камери.

Окремо у файлі robotApi.js формуються посилання на відеопотік і snapshot. Функція getStreamUrl() повертає адресу маршруту /camera/stream, яка використовується як джерело зображення у компоненті VideoPanel. Функція getSnapshotUrl() формує адресу маршруту /camera/snapshot, через який користувач може отримати окремий знімок з камери.

Такий підхід дозволяє зберігати логіку взаємодії з API в одному файлі. Якщо адреса серверної частини або API-маршрути зміняться, достатньо буде змінити файл robotApi.js, не переписуючи окремі компоненти Web-інтерфейсу.

3.4.3 Побудова компонентів Web-інтерфейсу

Побудова Web-інтерфейсу виконана на основі окремих React-компонентів. Основна сторінка інтерфейсу реалізована у файлі DashboardPage.jsx. У цьому

компоненті об’єднуються основні функціональні блоки: панель відеопотоку, панель керування рухом, блок стану системи, блок IMU Sensor та журнал подій.

Фрагмент побудови основної сторінки наведено нижче.

```
export const DashboardPage = ({ start, stop, status }) => {
  return (
    <div className="grid w-full grid-cols-1 gap-6 xl:grid-cols-[1.2fr_0.8fr]">
      <div className="space-y-6">
        <VideoPanel
          cameraMode={status.camera_mode}
          cameraModes={status.camera_modes}
          onModeChange={status.refresh}
        />
        <ControlPad start={start} stop={stop} />
      </div>
      <div className="space-y-6">
        <StatusCard
          online={status.online}
          cameraStarted={status.camera_started}
          motion={status.motion}
        />
        <ImuCard imu={status.imu} />
      </div>
    </div>
  )
}
```

У цьому фрагменті видно, що сторінка не описується одним великим блоком коду, а складається з окремих компонентів. Компонент VideoPanel відповідає за відображення відеопотоку та роботу з режимами камери, ControlPad – за керування рухом, StatusCard – за відображення стану системи, а ImuCard – за стан інерціального модуля.

Компонент VideoPanel використовує адресу відеопотоку, сформовану у файлі robotApi.js. Відео відображається через елемент img, для якого джерелом є маршрут /camera/stream.

```
<img
  src={robotApi.getStreamUrl()}
  alt="Robot stream"
  className="aspect-[4/3] w-full object-contain"
/>
```

Таке рішення дозволяє відображати потік з камери без окремого відеоплеєра. Браузер отримує послідовність JPEG-кадрів від серверної частини та показує їх у блоці Live Camera.

Керування рухом реалізовано у компонентах ControlPad та ControlButton. Компонент ControlPad формує розташування кнопок руху, а ControlButton відповідає за обробку натискання окремої кнопки. Для кожної кнопки задаються функції onStart та onStop.

```
<ControlButton
  label="Forward"
  icon={<ArrowUp size={24} />}
  onStart={() => start('forward')}
  onStop={stop}
/>
```

Функція onStart викликається під час натискання кнопки, а onStop – після її відпускання. Завдяки цьому платформа рухається тільки тоді, коли користувач утримує кнопку керування. Аналогічна логіка використовується і для керування з клавіатури.

Для обробки клавіатури використовується хук useKeyboardControl. Він обробляє клавіші W, A, S, D та Space. Клавіші W, A, S, D відповідають за рух вперед, ліворуч, назад і праворуч, а Space використовується для зупинки. При натисканні клавіші надсилається команда руху, а після відпускання - команда stop.

У головному компоненті App.jsx підключаються основна сторінка, хуки керування та перемикання теми інтерфейсу. Обрана тема зберігається у

localStorage, тому після перезавантаження сторінки інтерфейс відкривається в останньому вибраному режимі.

```
const { start, stop } = useHoldControl()
const status = useRobotStatus()

useKeyboardControl({ start, stop })

return (
  <MainLayout theme={theme} onThemeChange={setTheme}>
    <DashboardPage
      start={start}
      stop={stop}
      status={status}
    />
  </MainLayout>
)
```

У цьому фрагменті useHoldControl() забезпечує керування рухом під час утримання кнопки або клавіші, useRobotStatus() отримує актуальний стан системи від backend, а useKeyboardControl() підключає керування з клавіатури. Після цього всі потрібні дані та функції передаються на сторінку DashboardPage.

Компонентна структура клієнтської частини дозволяє окремо реалізувати відображення відео, керування рухом, стан системи, IMU та журнал подій. Завдяки цьому Web-інтерфейс залишається зрозумілим за структурою, а окремі частини сторінки можна змінювати без повної переробки клієнтської частини.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОБОТИ СИСТЕМИ

У попередніх розділах було розглянуто вибір апаратних компонентів, структуру програмно-апаратної системи, підключення основних модулів, алгоритм роботи, серверну частину та клієнтський Web-інтерфейс. Після реалізації цих елементів було отримано готову мобільну платформу з дистанційним керуванням, передаванням відеозображення.

У цьому розділі наведено результат практичної реалізації системи.

4.1 Загальний вигляд розробленої системи

Розроблена система складається з мобільної платформи та Web-інтерфейсу користувача. Мобільна платформа містить Raspberry Pi Zero 2 WH, драйвер двигунів L298N, камеру Raspberry Pi NoIR Camera v2, колісне шасі з двигунами та джерела живлення. Користувач взаємодіє із системою через Web-інтерфейс, який відкривається у браузері та не потребує встановлення окремого програмного забезпечення.

Загальний вигляд розробленої мобільної платформи наведено на рисунку 4.1.

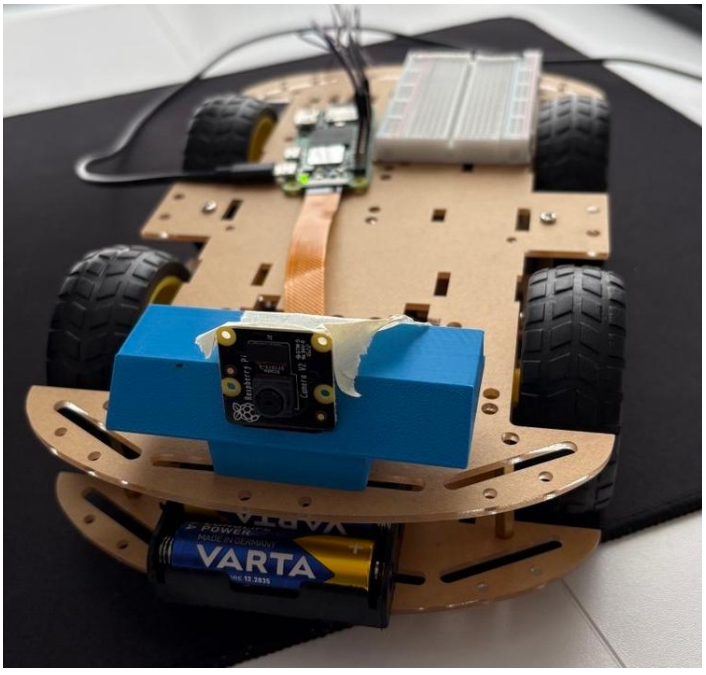


Рисунок 4.1 – Загальний вигляд розробленої мобільної платформи

Web-інтерфейс системи побудований у вигляді однієї сторінки, на якій розміщено основні блоки керування та моніторингу. У лівій частині інтерфейсу відображається відеопотік з камери та панель керування рухом. У правій частині розміщено блок стану системи, блок IMU Sensor та журнал подій.

Загальний вигляд Web-інтерфейсу у темному режимі наведено на рисунку 4.2.

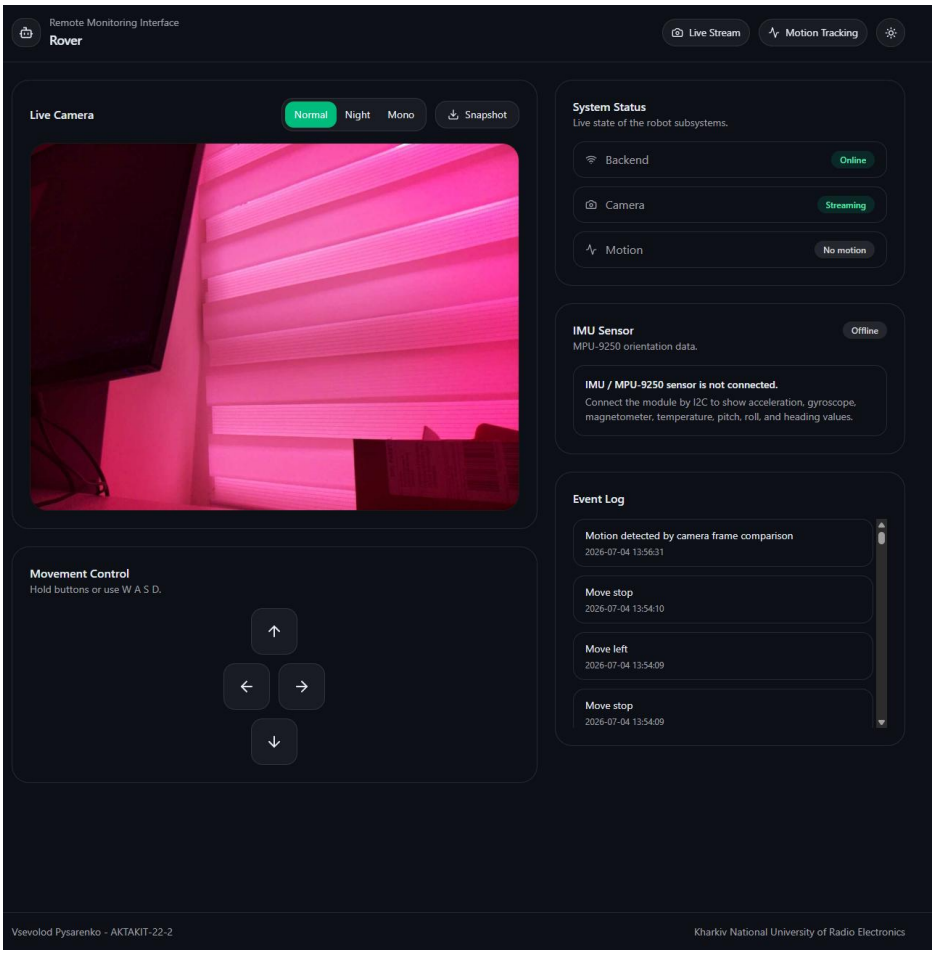


Рисунок 4.2 – Загальний вигляд Web-інтерфейсу у темному режимі

Для зручності користувача в інтерфейсі передбачено перемикання теми оформлення. Світлий режим Web-інтерфейсу наведено на рисунку 4.3.

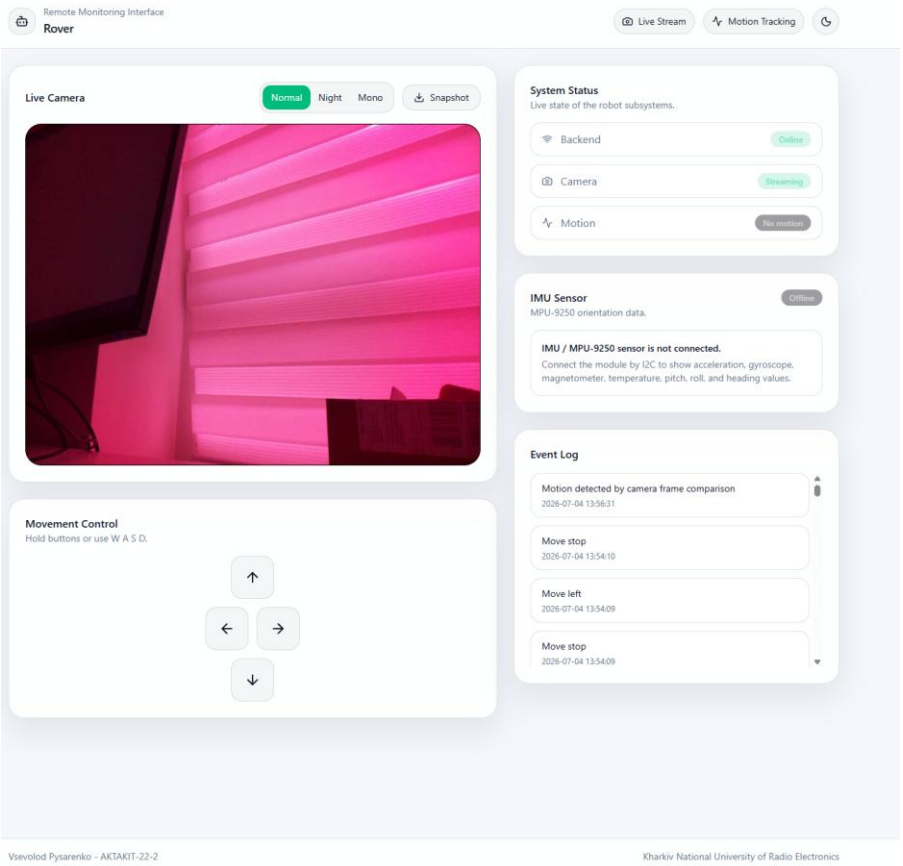


Рисунок 4.3 – Загальний вигляд Web-інтерфейсу у світлому режимі

Основні функціональні блоки Web-інтерфейсу відповідають можливостям розробленої системи. Блок Live Camera використовується для відображення відеозображення з камери. Блок Movement Control призначений для керування рухом мобільної платформи за допомогою кнопок або клавіш W, A, S, D. Блок System Status показує стан backend, камери та режиму виявлення руху. Блок IMU Sensor відображає стан інерціального модуля, а Event Log містить останні події системи.

Такий вигляд інтерфейсу дозволяє користувачу одночасно спостерігати відеопотік, керувати платформою та контролювати стан основних функцій системи.

4.2 Робота Web-інтерфейсу та відеопотоку

Основна взаємодія користувача з мобільною платформою виконується через Web-інтерфейс. Після запуску серверної та клієнтської частин користувач відкриває сторінку керування у браузері. На сторінці відображається відеопотік з камери, панель керування рухом, стан системи, стан IMU Sensor та журнал подій.

Відеопотік з камери відображається у блоці Live Camera. Для цього клієнтська частина використовує маршрут /camera/stream, через який сервер передає послідовність JPEG-кадрів. Завдяки цьому користувач може спостерігати простір перед мобільною платформою у реальному часі.

Відображення відеопотоку у звичайному режимі наведено на рисунку 4.4.

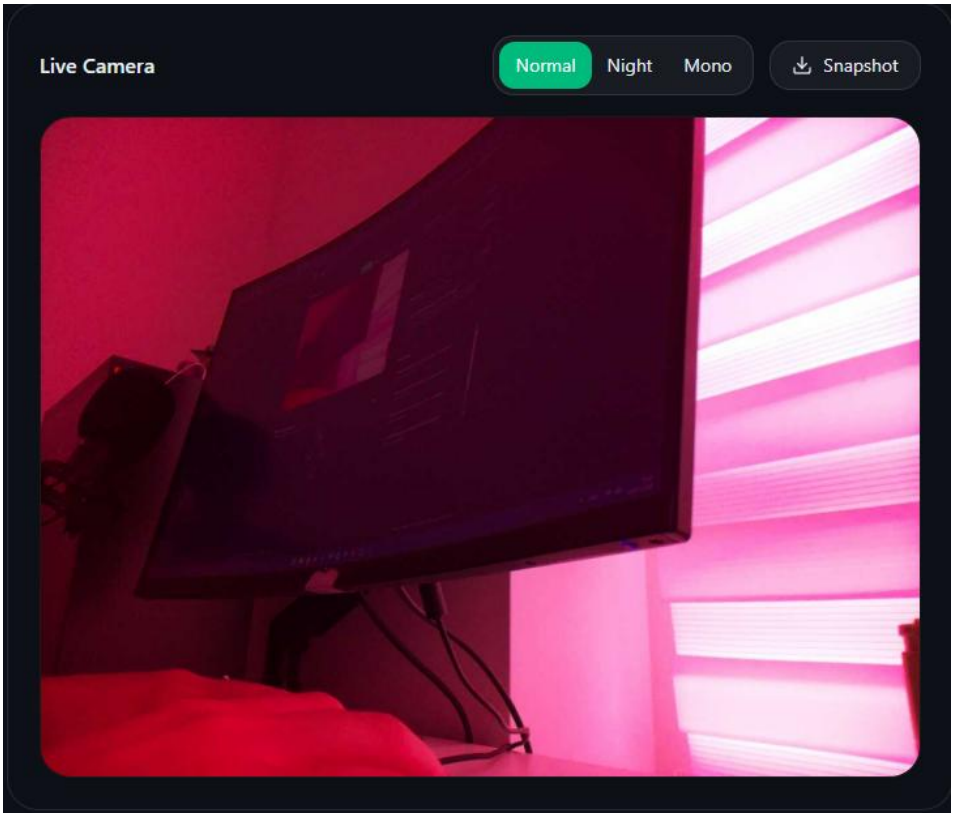


Рисунок 4.4 – Відображення відеопотоку у звичайному режимі камери

Оскільки у системі використовується Raspberry Pi NoIR Camera v2, зображення у звичайному режимі може мати характерний фіолетовий відтінок. Це пов’язано з особливістю NoIR-камери, яка не має стандартного

інфрачервоного фільтра. Такий модуль було обрано, оскільки він може використовуватися не тільки при звичайному освітленні, але й у задачах спостереження за умов недостатнього освітлення.

У Web-інтерфейсі передбачено перемикання режимів зображення. Користувач може обрати один із доступних режимів камери за допомогою кнопок у блоці Live Camera. У практичній реалізації було перевірено звичайний та монохромний режими відображення.

Відображення відеопотоку у монохромному режимі наведено на рисунку 4.5.

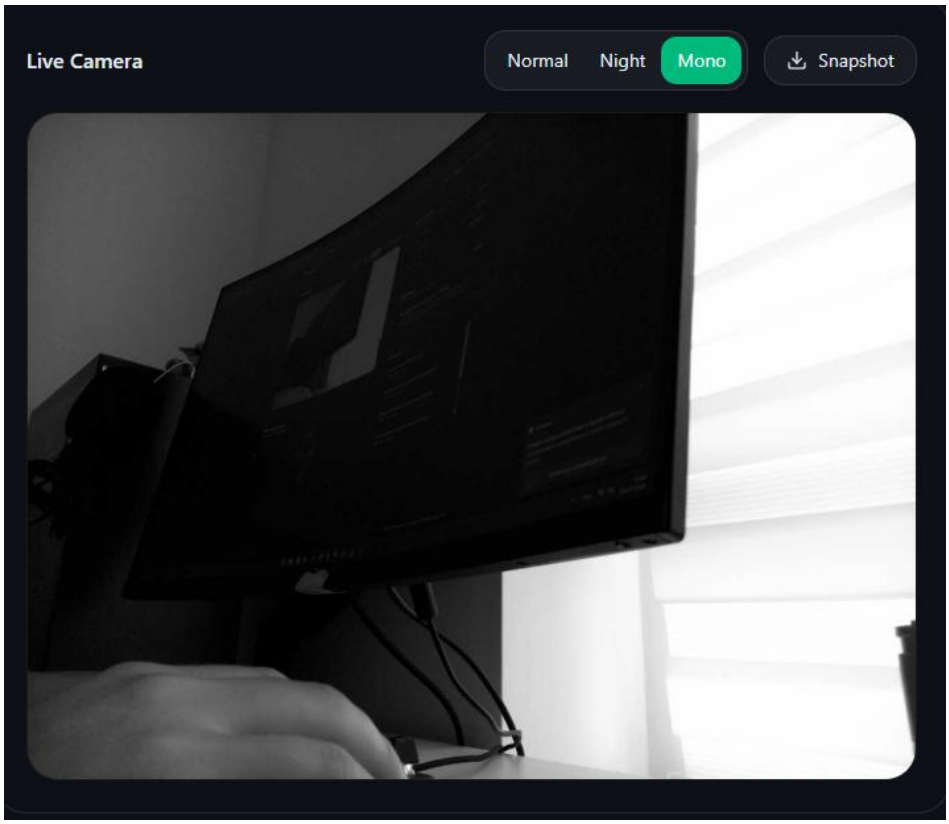


Рисунок 4.5 – Відображення відеопотоку у монохромному режимі

Монохромний режим використовується для відображення зображення без кольорової складової. Він може бути корисним у випадках, коли важливішою є загальна форма об’єктів у кадрі, а не їх колір.

Окрім перегляду відеопотоку, у системі реалізовано створення окремого знімка з камери. Для цього у Web-інтерфейсі використовується кнопка Snapshot,

після натискання якої клієнтська частина звертається до маршруту /camera/snapshot. Серверна частина зберігає актуальний кадр і повертає користувачу зображення у форматі JPEG.

Приклад отримання snapshot через серверний маршрут наведено на рисунку 4.6.

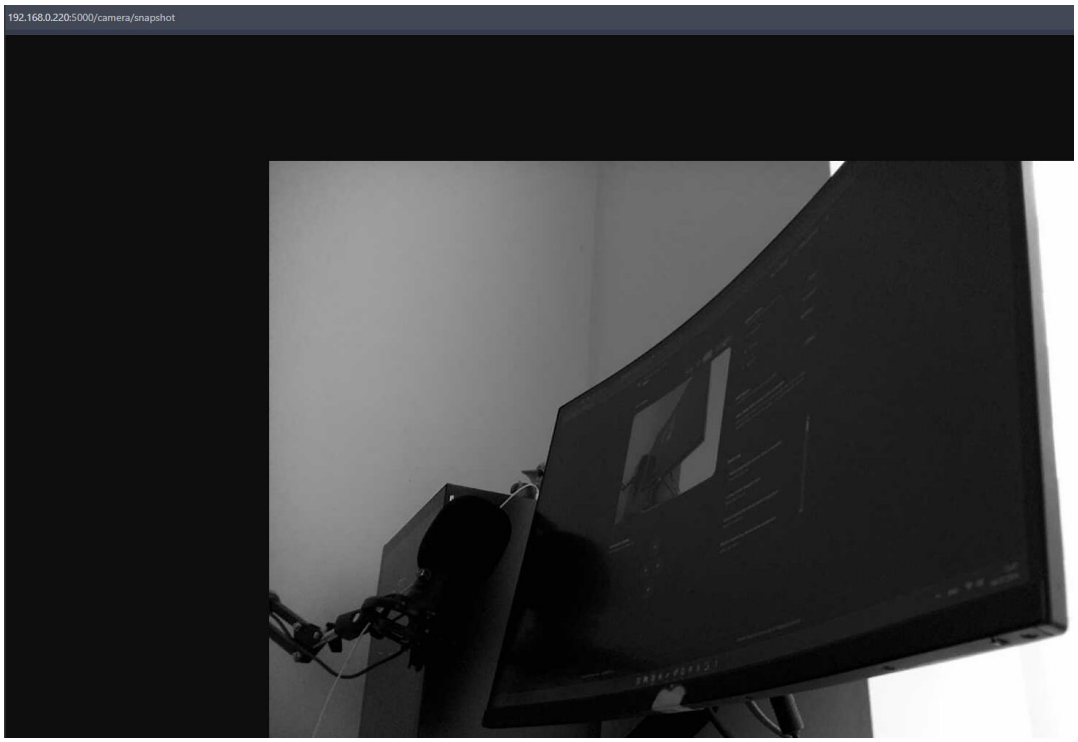


Рисунок 4.6 – Отримання snapshot через маршрут /camera/snapshot

Web-інтерфейс забезпечує не тільки безперервне відображення відеопотоку, але й можливість перемикання режимів камери та створення окремих знімків. Це дозволяє використовувати мобільну платформу для дистанційного спостереження та фіксації окремих кадрів під час роботи системи.

4.3 Робота системи керування рухом мобільної платформи

Керування рухом мобільної платформи виконується через блок Movement Control у Web-інтерфейсі. У цьому блоці розміщено кнопки для руху вперед, назад, ліворуч, праворуч та зупинки. Користувач може керувати платформою за допомогою кнопок на сторінці або клавіш W, A, S, D та Space на клавіатурі.

Панель керування рухом мобільної платформи наведено на рисунку 4.7.

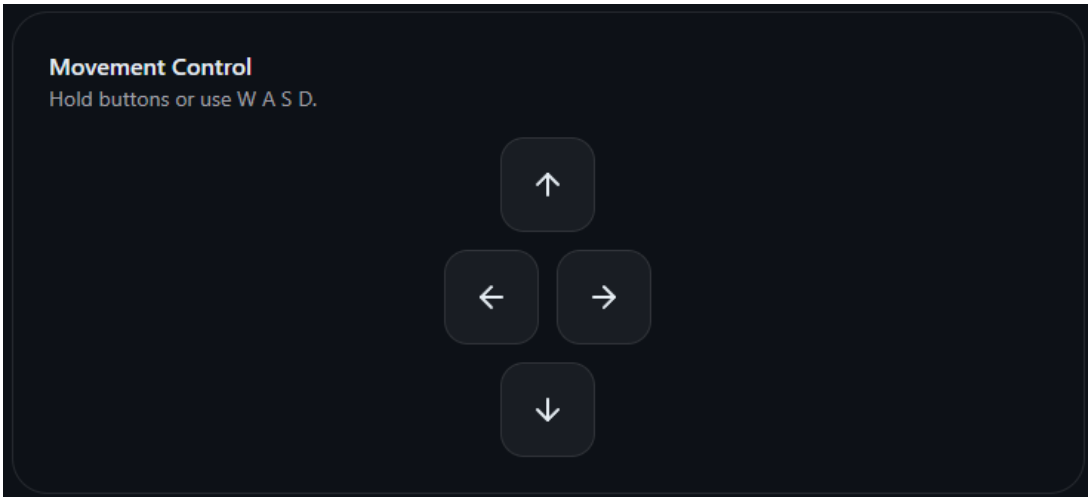


Рисунок 4.7 – Панель керування рухом мобільної платформи

Під час натискання кнопки або відповідної клавіші клієнтська частина надсилає POST-запит до серверної частини. Наприклад, для руху вперед використовується маршрут /move/forward, для руху назад – /move/backward, для повороту ліворуч – /move/left, для повороту праворуч – /move/right. Після відпускання кнопки або клавіші надсилається команда /move/stop, у результаті чого двигуни зупиняються.

Основні команди керування рухом наведено у таблиці 4.1.

Таблиця 4.1 – Команди керування рухом мобільної платформи

Дія користувача	Команда	Результат
Натискання кнопки Forward або клавіші W	/move/forward	Рух платформи вперед
Натискання кнопки Backward або клавіші S	/move/backward	Рух платформи назад
Натискання кнопки Left або клавіші A	/move/left	Поворот платформи ліворуч
Натискання кнопки Right або клавіші D	/move/right	Поворот платформи праворуч

Продовження таблиці 4.1

Дія користувача	Команда	Результат
Відпускання кнопки або клавіші Space	/move/stop	Зупинка платформи

Під час практичної перевірки було підтверджено реакцію платформи на основні команди руху. Після натискання кнопок у Web-інтерфейсі або клавіш на клавіатурі серверна частина передавала відповідні сигнали на драйвер L298N, після чого змінювався напрямок обертання груп двигунів.

Окремо було перевірено логіку утримання команди руху. Платформа виконує рух лише під час активного утримання кнопки або клавіші. Після припинення введення надсилається команда зупинки, і двигуни вимикаються.

Роботу мобільної платформи під час виконання команд руху наведено у демонстраційних матеріалах додатка В.

4.4 Робота інформаційних блоків Web-інтерфейсу

Окрім відеопотоку та керування рухом, у Web-інтерфейсі передбачено інформаційні блоки для контролю стану системи. Вони дозволяють користувачу бачити, чи працює серверна частина, чи запущена камера, який режим зображення активний, чи виявлено рух у кадрі та який стан має інерціальний модуль.

Блок System Status показує основні стани системи: доступність сервера, стан камери, активний режим камери та результат роботи motion detection. Завдяки цьому користувач може швидко перевірити, чи доступна система та чи працюють основні функції.

Блок System Status показує основні стани системи: доступність сервера, стан камери, активний режим камери та результат роботи motion detection. Завдяки цьому користувач може швидко перевірити, чи доступна система та чи працюють основні функції.

Блоки System Status та IMU Sensor наведено на рисунку 4.8.

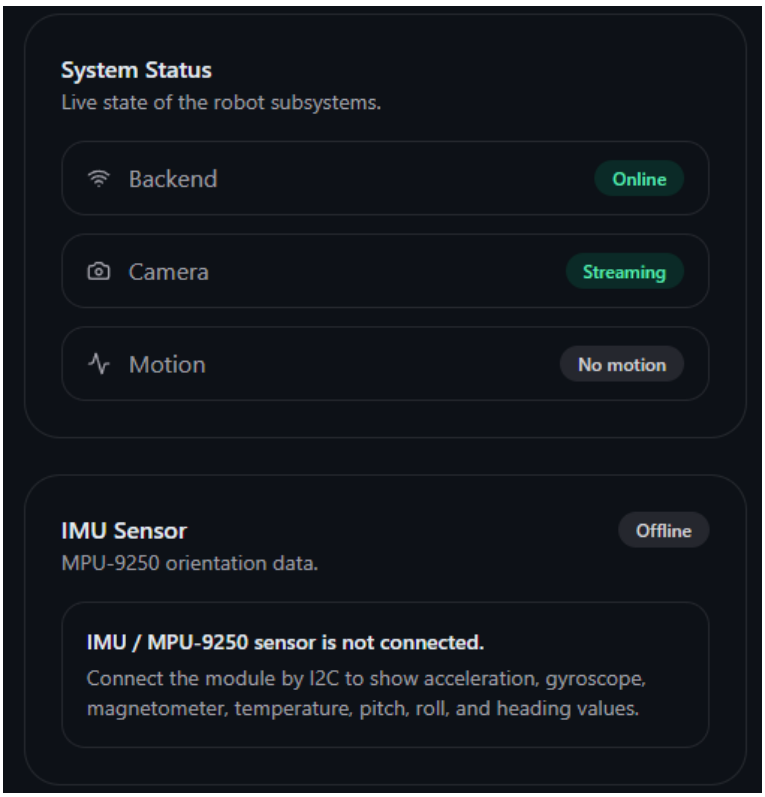


Рисунок 4.8 – Відображення блоків System Status та IMU Sensor

Блок IMU Sensor використовується для відображення стану інерціального модуля MPU-9250. Під час реалізації практичної частини було виявлено дефект модуля, через який датчик не передавав дані на серверну частину. У такому випадку Web-інтерфейс відображає стан IMU Offline. Це показує, що програмна частина коректно обробляє відсутність відповіді від додаткового сенсорного модуля і не блокує роботу основних функцій системи.

Окремо в інтерфейсі відображається журнал подій Event Log. У ньому фіксуються останні дії системи: запуск сервера, зміна режиму камери, створення snapshot, команди руху, зупинка платформи та виявлення руху.

Блок Event Log наведено на рисунку 4.9.

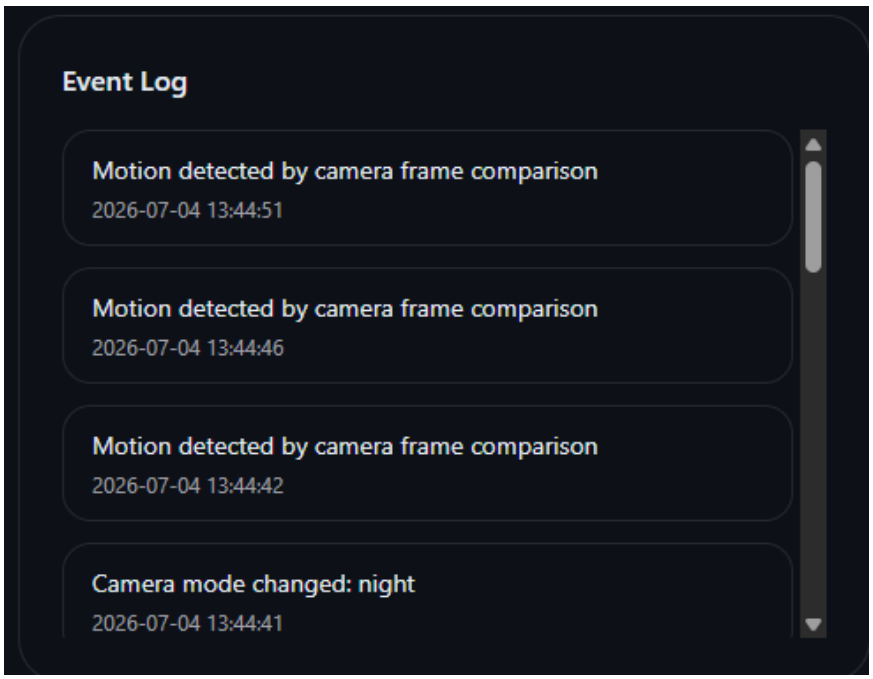


Рисунок 4.9 – Відображення журналу подій Event Log

Також було перевірено роботу функції motion detection. У звичайному стані система відображає відсутність руху в кадрі. Після появи помітної зміни зображення стан змінюється на Motion Detected.

Приклад відображення стану motion detection наведено на рисунках 4.10 та 4.11.

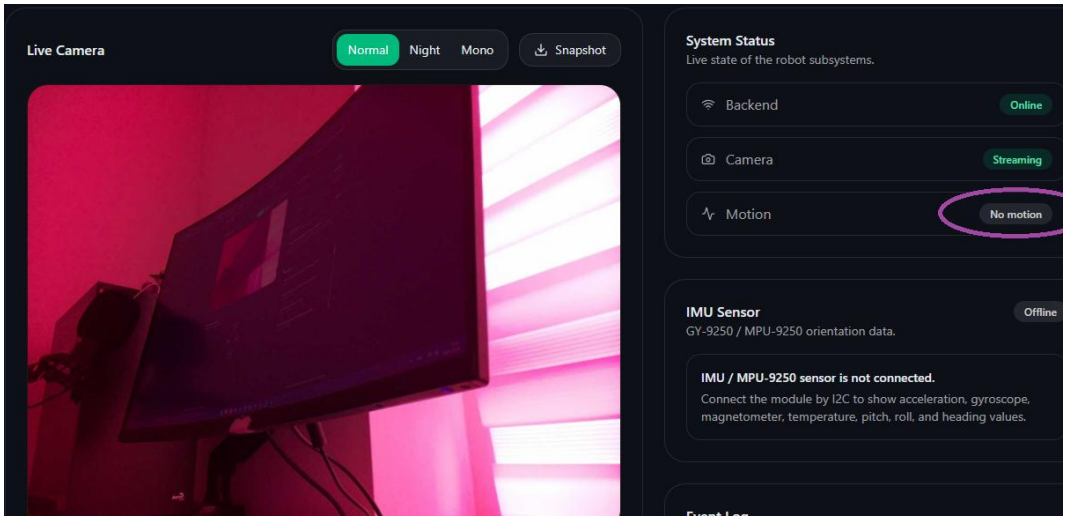


Рисунок 4.10 – Відображення стану Motion Detection без виявленого руху

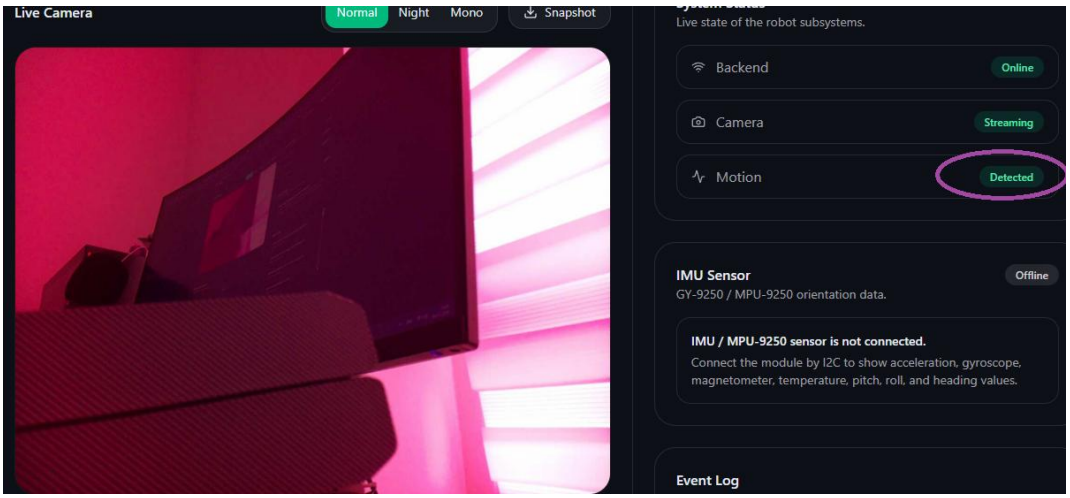


Рисунок 4.11 – Відображення стану Motion Detection з виявленим рухом

5 ОХОРОНА ПРАЦІ

5.1 Загальні положення

Охорона праці передбачає створення безпечних умов під час виконання технічних робіт та зменшення впливу можливих небезпечних факторів. У цій роботі це пов’язано з використанням комп’ютерної техніки, електронних компонентів, джерел живлення та рухомої мобільної платформи.

Основні вимоги у сфері охорони праці визначаються Законом України «Про охорону праці» та нормативними документами щодо безпеки роботи з екранними пристроями [42, 43]. У межах цієї роботи потрібно враховувати безпеку під час роботи за комп’ютером, а також під час підключення низьковольтних електронних компонентів, джерел живлення, камери, драйвера двигунів та рухомих частин мобільної платформи.

Розроблена система не використовує промислову силову напругу, однак під час її складання та перевірки можливі ризики короткого замикання, неправильного підключення компонентів, перегріву окремих елементів і контакту з рухомими колесами. Тому під час виконання роботи необхідно дотримуватися правил безпечної роботи з комп’ютерною технікою, електронними пристроями та джерелами живлення.

5.2 Аналіз небезпечних та шкідливих факторів

Під час виконання роботи основні небезпечні та шкідливі фактори пов’язані з роботою за персональним комп’ютером, підключенням електронних компонентів та перевіркою руху мобільної платформи. Під час програмування і налагодження Web-інтерфейсу можливе зорове навантаження, втома, тривале перебування у статичній позі та напруження під час роботи з екранним пристроєм [43, 45].

Під час складання апаратної частини можуть виникати ризики, пов’язані з неправильним підключенням проводів, коротким замиканням, перегрівом окремих електронних компонентів або пошкодженням джерел живлення.

Окремим фактором є наявність рухомих частин мобільної платформи. Під час перевірки руху можливий контакт з колесами або зміщення платформи зі столу чи робочої поверхні. Тому тестування руху потрібно виконувати на вільній рівній поверхні та не торкатися коліс під час роботи двигунів.

5.3 Вимоги безпеки під час роботи з комп’ютерною технікою та електронними компонентами

Під час роботи за комп’ютером необхідно правильно організувати робоче місце: розмістити монітор на зручній відстані, забезпечити достатнє освітлення та робити короткі перерви під час тривалої роботи. Це зменшує навантаження на зір, спину та кисті рук під час програмування, налагодження Web-інтерфейсу й оформлення пояснювальної записки [43, 45].

Під час підключення електронних компонентів потрібно перевіряти правильність з’єднань перед подачею живлення. Зміна підключення проводів, камери, драйвера, Raspberry Pi або інших модулів має виконуватися лише після вимкнення живлення.

Не допускається використання пошкоджених проводів, нестабільних джерел живлення або випадкове з’єднання контактів живлення між собою. Проводи мають бути закріплені так, щоб вони не потрапляли в рухомі частини платформи та не заважали обертанню коліс.

Під час перевірки руху мобільної платформи її потрібно розміщувати на рівній вільній поверхні. Не слід торкатися коліс або двигунів під час роботи, а також залишати платформу увімкненою без нагляду.

5.4 Пожежна безпека

Під час роботи з комп’ютерною технікою та електронними компонентами необхідно дотримуватися правил пожежної безпеки [44]. Робоче місце має бути вільним від зайвих предметів, легкозаймистих матеріалів та пошкоджених кабелів.

Не допускається перевантаження розеток, використання несправних блоків живлення або залишення увімкненої системи без нагляду. Перед підключенням Raspberry Pi, драйвера двигунів та інших модулів потрібно перевіряти стан проводів і правильність з’єднань.

У разі появи запаху гару, нагріву проводів або нестабільної роботи живлення необхідно одразу вимкнути систему та припинити перевірку до усунення причини несправності.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено мобільну платформу з дистанційним керуванням та передачею відеозображення через Web-інтерфейс. У результаті виконання роботи отримано програмно-апаратну систему, яка поєднує рухому платформу, камеру, модуль керування двигунами, серверну частину та клієнтський Web-інтерфейс.

Під час роботи було проаналізовано технічні рішення для побудови мобільних роботизованих платформ, способи дистанційного керування, засоби передавання відеозображення та варіанти реалізації Web-інтерфейсу. На основі цього було обрано наземну колісну платформу, оскільки вона має просту конструкцію, достатню керованість і підходить для переміщення в межах приміщень.

Для реалізації системи було обрано Raspberry Pi Zero 2 WH як керуючий модуль, драйвер L298N для керування DC-двигунами, Raspberry Pi NoIR Camera v2 для отримання відеозображення та інерціальний модуль MPU-9250 для відображення додаткових параметрів стану платформи. Такий набір компонентів дозволив побудувати компактну систему з можливістю дистанційного керування та відеоспостереження.

Серверну частину реалізовано на основі Flask API. Вона забезпечує приймання команд від Web-інтерфейсу, керування напрямком руху платформи, передавання відеопотоку, створення snapshot, перемикання режимів камери, виявлення руху в кадрі, ведення журналу подій та автоматичну зупинку двигунів у разі відсутності актуальних команд.

Клієнтська частина реалізована на основі React.js. Через Web-інтерфейс користувач може переглядати відеопотік, керувати рухом платформи, створювати знімки, змінювати режим камери та контролювати стан системи. В інтерфейсі також відображаються службові блоки System Status, IMU Sensor, Motion Detection та Event Log.

Під час практичної перевірки було підтверджено роботу основних функцій системи: передавання відеозображення, створення snapshot, перемикання режимів камери, керування рухом платформи, фіксацію подій у журналі та виявлення руху в кадрі. Під час реалізації практичної частини було виявлено дефект інерціального модуля MPU-9250, через який датчик не передавав дані на серверну частину. При цьому програмна частина коректно обробила відсутність відповіді від модуля та відобразила стан IMU Offline без порушення роботи інших функцій.

Розроблена система може використовуватися для дистанційного спостереження, огляду приміщень і технічних зон, де користувачу потрібно отримувати відеозображення та керувати платформою без безпосередньої присутності біля об’єкта. Поставлену мету роботи було досягнуто, оскільки мобільна платформа реалізує дистанційне керування, передавання відеозображення та взаємодію з користувачем через Web-інтерфейс.

Отримані результати кваліфікаційної роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», оскільки розроблена система поєднує апаратні та програмні засоби, Web-технології, робототехнічну платформу та елементи автоматизації для вирішення задач дистанційного керування і спостереження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про організацію проведення практики здобувачів вищої освіти Харківського національного університету радіоелектроніки [Електронний ресурс] : Наказ ХНУРЕ від 03.05.2019 р. № 222. Режим доступу : https://nure.ua/wp-content/uploads/222-vid-03.05.2019-pro-vvedennja-v-diju-rishennja-vchenoi-radi-universitetu.pdf (дата звернення: 24.06.2026).

2. Положення про академічну доброчесність [Електронний ресурс] : Наказ ХНУРЕ від 02.02.2021 р. № 50. Режим доступу : https://nure.ua/wp-content/uploads/Main_Docs_NURE/polozhennja-pro-akademichnu-dobrochesnist.pdf (дата звернення: 24.06.2026).

3. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми Автоматизація та комп'ютерно-інтегровані технології [Електронний ресурс] / упоряд.: І. Ш. Невлюдов, О. І. Филипенко, О. В. Токарєва, С. П. Новоселов, О. В. Сичова ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків : ХНУРЕ, 2023. 760 Kb.

4. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2016. 31 с.

5. Татарінцев О. В. Система керування мобільним роботом на основі IoT технологій : пояснювальна записка до атестаційної роботи здобувача вищої освіти на другому (магістерському) рівні, спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології / О. В. Татарінцев ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки, кафедра комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки. – Харків, 2019. – 74 с. [Електронний ресурс]. – Режим доступу: https://openarchive.nure.ua/entities/publication/b615b012-c6d0-408e-a9f5-e8883cd977fd (дата звернення: 06.07.2026).

6. Зарубін І. С. Особливості та різновиди сучасних роботизованих платформ / І. С. Зарубін // Automation and Development of Electronic Devices (ADED'2022) : collection of Students' Scientific Paper. Харків : ХНУРЕ, 2022. Part 2. P. 55-60. Режим доступу : https://openarchive.nure.ua/handle/document/21128 (дата звернення: 24.06.2026).

7. Зарубін І. С. Аналіз конструкції мобільної роботизованої платформи на колесах / І. С. Зарубін // Радіоелектроніка та молодь у XXI столітті : зб. матеріалів 27-го Міжнар. молодіж. форуму, 10-12 травня 2023 р. Харків : ХНУРЕ, 2023. Т. 2. С. 17-18. Режим доступу : https://openarchive.nure.ua/handle/document/23053 (дата звернення: 24.06.2026).

8. Крутовий А. Р. Розробка 3D-моделі системи переміщення робота з покращеними можливостями : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на першому (бакалаврському) рівні, спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології / А. Р. Крутовий ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2024. 52 с. Режим доступу : https://openarchive.nure.ua/handle/document/27873 (дата звернення: 24.06.2026).

9. Husky A300 Unmanned Ground Vehicle Robot [Електронний ресурс] // Clearpath Robotics. – Режим доступу : https://clearpathrobotics.com/husky-a300-unmanned-ground-vehicle-robot/ (дата звернення: 06.07.2026).

10. PackBot 525 [Електронний ресурс] // Teledyne FLIR Defense. – Режим доступу : https://defense.flir.com/defense-products/packbot-525/ (дата звернення: 06.07.2026).

11. Spot. The Agile Mobile Robot [Електронний ресурс] // Boston Dynamics. – Режим доступу : https://bostondynamics.com/products/spot/ (дата звернення: 06.07.2026).

12. Matrice 350 RTK [Електронний ресурс] // DJI Enterprise. – Режим доступу : https://enterprise.dji.com/matrice-350-rtk (дата звернення: 06.07.2026).

13. Вернигора Д. С. Розробка системи автоматизації віддаленого управління мобільним роботом : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на першому (бакалаврському) рівні / Д. С. Вернигора ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2024. Режим доступу : https://openarchive.nure.ua/handle/document/27600 (дата звернення: 24.06.2026).

14. Єрошенко О. А. Програмно-апаратний комплекс для дистанційного керування мобільним роботом / О. А. Єрошенко, А. С. Щербак // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доповідей 15-ї міжнародної науково-технічної конференції, 24-25 квітня 2025 р. Баку, Харків, Жиліна, 2025. С. 126. Режим доступу : https://openarchive.nure.ua/handle/document/32047 (дата звернення: 24.06.2026).

15. Назаренко В. Д. Розроблення радіочастотного пульта керування дальнього радіусу дії для мобільного робота : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на першому (бакалаврському) рівні / В. Д. Назаренко ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2024. Режим доступу : https://openarchive.nure.ua/handle/document/27022 (дата звернення: 24.06.2026).

16. Машталяр С. В. Дистанційно керована мобільна платформа з функцією відеомоніторингу : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на першому (бакалаврському) рівні / С. В. Машталяр ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2025. Режим доступу : https://openarchive.nure.ua/handle/document/32246 (дата звернення: 24.06.2026).

17. Українська DevDroid впровадила у свої НРК систему детекції цілей зі штучним інтелектом [Електронний ресурс] // Militarnyi. Режим доступу : https://militarnyi.com/uk/news/ukrayinska-devdroid-vprovadyla-u-svoyi-nrk-systemu-detektsiyi-tsilej-zi-shtuchnym-intelektom/ (дата звернення: 24.06.2026).

18. Introduction to the DJI Fly App Page [Електронний ресурс] // DJI Support. Режим доступу : https://support.dji.com/help/content?customId=en-us03400006562&documentType=artical&lang=en&paperDocType=paper&re=US&spaceId=34 (дата звернення: 24.06.2026).

19. Fly View | QGC Guide (v5.0) [Електронний ресурс] // QGroundControl Documentation. Режим доступу : https://docs.qgroundcontrol.com/Stable_V5.0/en/qgc-user-guide/fly_view/hud.html (дата звернення: 24.06.2026).

20. Go1 APP Download [Електронний ресурс] // Unitree Robotics. Режим доступу : https://www.unitree.com/app/go1/ (дата звернення: 24.06.2026).

21. Camera [Електронний ресурс] // Raspberry Pi Documentation. Режим доступу : https://www.raspberrypi.com/documentation/hardware/camera/picam/ (дата звернення: 24.06.2026).

22. Camera software [Електронний ресурс] // Raspberry Pi Documentation. Режим доступу : https://www.raspberrypi.com/documentation/computers/camera_software.html (дата звернення: 24.06.2026).

23. Media Capture and Streams API [Електронний ресурс] // MDN Web Docs. Режим доступу : https://developer.mozilla.org/en-US/docs/Web/API/Media_Capture_and_Streams_API (дата звернення: 24.06.2026).

24. Arduino UNO Rev3 [Електронний ресурс] // Arduino Official Store. Режим доступу : https://store.arduino.cc/products/arduino-uno-rev3 (дата звернення: 24.06.2026).

25. ESP32 Wi-Fi & Bluetooth SoC [Електронний ресурс] // Espressif Systems. Режим доступу : https://www.espressif.com/en/products/socs/esp32/specifications (дата звернення: 24.06.2026).

26. Raspberry Pi Zero 2 W [Електронний ресурс] // Raspberry Pi. Режим доступу : https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/ (дата звернення: 24.06.2026).

27. Yevsieiev V. Development of a Streaming Video Broadcast System About the Environmental Space of A Mobile Robot Based on ESP32-Cam / V. Yevsieiev // Автоматизація та комп'ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку : матеріали Всеукр. наук.-практ. Internet-конференції. Черкаси, 2023. С. 166-168. Режим доступу : https://openarchive.nure.ua/handle/document/22840 (дата звернення: 24.06.2026).

28. Лихо Т. А. Розроблення Веб-сторінки керування мобільним роботом через протокол MQTT / Т. А. Лихо // Computer-integrated technologies, automation and robotics 2024 : proceedings of the I st All-Ukrainian Conference, Kharkiv, May 16-17, 2024. Kharkiv, 2024. P. 66-71. Режим доступу : https://openarchive.nure.ua/handle/document/26174 (дата звернення: 24.06.2026).

29. Raspberry Pi OS [Електронний ресурс] // Raspberry Pi Documentation. Режим доступу : https://www.raspberrypi.com/documentation/computers/os.html (дата звернення: 24.06.2026).

30. L298 Dual Full-Bridge Driver [Електронний ресурс] // STMicroelectronics. Режим доступу : https://www.st.com/resource/en/datasheet/l298.pdf (дата звернення: 24.06.2026).

31. Гурін Д. В. Розробка структурної схеми підключення роботизованої платформи для розмінування / Д. В. Гурін // Manufacturing & Mechatronic Systems 2024 : матеріали VIII міжнародної науково-практичної конференції, 25-26 жовтня 2024 р., Харків. Харків, 2024. Режим доступу : https://openarchive.nure.ua/entities/publication/10f838fa-5153-4220-8a86-632cfa22922a (дата звернення: 24.06.2026).

32. MPU-9250 Product Specification [Електронний ресурс] // TDK InvenSense. Режим доступу : https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-9250-Datasheet.pdf (дата звернення: 24.06.2026).

33. Драйвер двох двигунів на L298N [Електронний ресурс] // Arduino.ua. Режим доступу : https://arduino.ua/prod406-draiver-dvyh-dvigatelei-na-l298n (дата звернення: 24.06.2026).

34. Welcome to Flask [Електронний ресурс] // Flask Documentation. Режим доступу : https://flask.palletsprojects.com/en/stable/ (дата звернення: 24.06.2026).

35. Django at a glance [Електронний ресурс] // Django Documentation. Режим доступу : https://docs.djangoproject.com/en/6.1/intro/overview/ (дата звернення: 24.06.2026).

36. Describing the UI [Електронний ресурс] // React Documentation. Режим доступу : https://react.dev/learn/describing-the-ui (дата звернення: 24.06.2026).

37. Tailwind CSS [Електронний ресурс] // Tailwind CSS Documentation. Режим доступу : https://tailwindcss.com/ (дата звернення: 24.06.2026).

38. Теорія автоматичного управління (збірник задач) [Текст] : навч. посіб. для студентів спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / І. Ш. Невлюдов, О. В. Токарєва ; Харків. нац. ун-т радіоелектроніки. Харків : Панов А. М., 2020. 240 с.

39. Рекомендації по застосуванню методів ТАУ при виконанні кваліфікаційної роботи бакалавра [Електронний ресурс]. Харків : ХНУРЕ.

40. The Picamera2 Library [Електронний ресурс] / Raspberry Pi Ltd. - Режим доступу: https://pip.raspberrypi.com/documents/RP-008156-DS-picamera2-manual.pdf (дата звернення: 04.07.2026).

41. simplejpeg [Електронний ресурс] // Python Package Index (PyPI). – Режим доступу: https://pypi.org/project/simplejpeg/ (дата звернення: 04.07.2026).

42. Комплекс навчально-методичного забезпечення навчальної дисципліни "Безпека праці в індустрії ІТ- технологій" підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [Електронний ресурс] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, І. І. Хондак. – Харків, 2024. – 171 с. (дата звернення: 05.07.2026).

43. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Наказ Міністерства соціальної політики України від 14.02.2018 № 207 [Електронний ресурс]. - Режим доступу: https://zakon.rada.gov.ua/go/z0508-18 (дата звернення: 05.07.2026).

44. Про затвердження Правил пожежної безпеки в Україні : Наказ Міністерства внутрішніх справ України від 30.12.2014 № 1417 [Електронний ресурс]. - Режим доступу: https://zakon.rada.gov.ua/go/z0252-15 (дата звернення: 05.07.2026).

45. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин : Правила МОЗ України від 10.12.1998 № 7 [Електронний ресурс]. - Режим доступу: https://zakon.rada.gov.ua/go/v0007282-98 (дата звернення: 05.07.2026).