МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Вбудована система контролю транспортування вантажів

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія»

Здобувач _____Нікіта ГЮЛЬМАМЕДОВ
підпис

«__» _____ 202__ р.

Керівник ст. викл.

_____ Іван БУРЛАЧЕНКО
підпис

«__» _____ 202__ р.

Миколаїв – 2026

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Гюльмамедова Нікіти Миколайовича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Вбудована система контролю транспортування вантажів

Затверджена наказом по ЧНУ ім. Петра Могили від 25.11.2025 № 294.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом виконання кваліфікаційної роботи є створення апаратного та програмного забезпечення для системи контролю транспортування вантажів

4. Перелік питань, що підлягають розробці:
1) дослідити існуючі системи транспортування вантажів
2) визначити необхідні вимоги для системи транспортування вантажів
3) створити принципову схему з підключенням комплектуючих апаратно програмної платформи
4) визначити необхідні засоби апаратної реалізації вбудованих систем транспортування вантажів
5) створити механізм фіксації вантажів та фотофіксації перешкод
провести дослідження створеної платформи

5. Перелік графічних матеріалів

Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Іван БУРЛАЧЕНКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Нікіта ГЮЛЬМАМЕДОВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Вбудована система контролю перевезень вантажів

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	04.12.2025	14.12.2025	Виконано
2	Огляд літератури за темою роботи	15.12.2025	31.12.2025	Виконано
3	Складання календарного плану КБР	03.01.2026	10.01.2026	Виконано
4	Аналіз предметної області	11.01.2026	26.01.2026	Виконано
5	Розробка проєктних рішень	27.01.2026	15.02.2026	Виконано
6	Моделювання та конструювання АПЗ	16.02.2026	28.02.2026	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	01.03.2026	02.03.2026	Виконано
8	Оформлення КБР та презентації	21.05.2026	21.05.2026	Виконано
9	Перший передній захист КБР	22.05.2026	22.05.2026	Виконано
10	Другий передній захист КБР	05.06.2026	05.06.2026	Виконано
11	Завершення оформлення КБР та презентації	06.06.2026	09.06.2026	Виконано
12	Перевірка на академічний плагіат, фальсифікацію та списування	10.06.2026	11.06.2026	Виконано
13	Відгук керівника КБР	11.06.2026	12.06.2026	Виконано
14	Подання КБР рецензенту та рецензування КБР	13.06.2026	14.06.2026	Виконано
15	Подання КБР, її електронної копії та інших документів (відгуку, рецензії)	15.06.2026	16.06.2026	Виконано
16	Захист кваліфікаційної бакалаврської роботи	24.06.2026	14.06.2026	Виконано

Керівник роботи _____ Іван БУРЛАЧЕНКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Нікіта ГЮЛЬМАМЕДОВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 2026__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Вбудована система контролю транспортування вантажів»
Здобувач: Гюльмамедов Нікіта Миколайович
Керівник: ст. викладач Бурлаченко Іван Сергійович

Автоматизація транспортних процесів є одним із ключових напрямків розвитку сучасної промисловості та логістики. Впровадження вбудованих систем контролю транспортування вантажів дозволяє суттєво знизити ризики для здоров'я персоналу, підвищити ефективність роботи підприємств. забезпечити виконання завдань у середовищах, небезпечних або недоступних для людини. Компактні дистанційно керовані транспортні засоби здатні замінити людину у вузьких складських приміщеннях, зонах із підвищеним рівнем задимлення або радіаційного забруднення, а також у важкодоступних місцях, куди велика вантажна техніка фізично не потрапляє. З розвитком ефективніших електродвигунів, ємніших акумуляторів та нових конструкційних матеріалів сфера застосування таких систем невпинно розширюється. Це можна побачити від автономної доставки посилок до морських і підземних операцій.

Актуальність: створити систему контролю транспортування вантажів набуває особливої важливості в умовах підвищених ризиків для персоналу.

Мета роботи: розробити прототип вбудованої системи дистанційного керування транспортним засобом із відеоспостереженням, здатної функціонувати в умовах, небезпечних або недоступних для людини.

Об'єкт дослідження: транспортування вантажів в умовах обмеженої інфраструктурної доступності.

Предмет дослідження: методи апаратної реалізації керування роботизованим транспортним засобом із дистанційним керуванням.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, переліку джерел посилань та 3 додатків.

У вступі визначається актуальність теми, формулюються мета роботи, об'єкт і предмет дослідження та надається короткий огляд поставленої задачі.

У першому розділі проведено аналіз сучасного стану розробки вбудованих систем контролю транспортування вантажів, розглянуто існуючі рішення та їхні характеристики, виявлено їх переваги та недоліки. Сформульовано вимоги до системи, що розробляється, та обґрунтовано план виконання роботи.

У другому розділі обґрунтовано вибір апаратної платформи, протоколів передачі даних та програмних інструментів. Розглянуто ключові компоненти системи: одноплатний комп'ютер, драйвер двигунів, ШІМ-контролер сервоприводів та ультразвуковий далекоміра. Обґрунтовано використання мови програмування Python.

У третьому розділі описано результати практичної реалізації системи: апаратну конфігурацію транспортного засобу, розробку програмного забезпечення серверної частини з багатопотоковою архітектурою, реалізацію FPV-відеострімінгу з можливістю перемикання профілів якості. Також продемонстровано вебінтерфейс оператора з керуванням рухом, сервоприводами та моніторингом даних сенсорів у реальному часі.

У висновках наведено аналіз виконаної роботи, сформульовано основні результати дослідження та окреслено перспективи подальшого розвитку системи.

В цілому, кваліфікаційна бакалаврська робота, містить 72 с. (без додатків), 30 рисунків, 11 таблиць, 37 джерел посилань та 3 додатків.

Ключові слова: *вбудована система, транспортування вантажів, дистанційне керування, Raspberry Pi, ультразвуковий сенсор, сервопривід, PCA9685, Python, роботизований транспортний засіб.*

ABSTRACT
of the Bachelor's Thesis
"Built-in cargo transportation control system"
Applicant: Nikita Hiulmamedov
Supervisor: Senior Lecturer Ivan Burlachenko

The automation of transportation processes is one of the key directions in the development of modern industry and logistics. The introduction of embedded cargo transportation control systems makes it possible to significantly reduce health risks for personnel, improve enterprise efficiency, and ensure task execution in environments that are hazardous or inaccessible to humans. Compact remotely controlled vehicles can replace humans in narrow warehouse aisles, areas with high levels of smoke or radiation contamination, and hard-to-reach locations where large cargo equipment simply cannot access. With the advancement of more efficient electric motors, higher-capacity batteries, and new structural materials, the scope of application of such systems continues to expand – from autonomous parcel delivery to maritime and underground operations.

Relevance: the development of a cargo transportation control system is of particular importance in conditions of elevated risks to personnel.

Purpose of the work: to develop a prototype embedded system for remote control of a transport vehicle with video surveillance, capable of operating in environments that are hazardous or inaccessible to humans.

Object of research: cargo transportation under conditions of limited infrastructural accessibility.

Subject of research: methods of hardware implementation of remote control of a robotic transport vehicle.

The bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references and 3 appendices.

The introduction defines the relevance of the topic, formulates the purpose of the work, the object and subject of research, and provides a brief overview of the problem statement.

The first chapter presents an analysis of the current state of embedded cargo transportation control systems, reviews existing solutions and their characteristics, and identifies their advantages and disadvantages. The requirements for the system under development are formulated and the work execution plan is justified.

The second chapter justifies the selection of the hardware platform, data transmission protocols, and software tools. The key system components are examined: the single-board computer, motor driver, servo PWM controller, and ultrasonic rangefinder. The use of the Python programming language is substantiated.

The third chapter describes the results of the practical system implementation: the hardware configuration of the transport vehicle, the development of server-side software

with a multithreaded architecture, and the implementation of FPV video streaming with switchable quality profiles. The operator web interface featuring motion control, servo management, and real-time sensor data monitoring is also demonstrated.

The conclusions present an analysis of the completed work, summarize the main research results, and outline the prospects for further development of the system.

In general, the bachelor's thesis, without appendices, contains 72 pages (excluding appendices), 34 figures, 11 tables, 37 references and 3 appendices.

Keywords: *embedded system, cargo transportation, remote control, Raspberry Pi, ultrasonic sensor, servo drive, PCA9685, Python, robotic vehicle.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПЗ – апаратно-програмне забезпечення

КБР – кваліфікаційна бакалаврська робота

ПЗ – програмне забезпечення

РТС – робототехнічна система

СКТВ – системи контролю транспортування вантажів

BMS – Battery Management System

FPS – Battery Management System

FPV – First Point View

IoT – Internet of Things

OSD – On-Screen Display

ВСТУП

У сьогоднішніх умовах активно розвиваються технології бездротового зв'язку та технології мікроконтролерів, що дає змогу створювати роботи системи контролю транспортування вантажів (СКТВ), що здатні виконувати завдання дистанційно, а в майбутньому, можливо, і автономно, без допомоги людини що керує роботизованою системою (РТС). Такі умови формують нові тренди на створення проєктів, а поліпшення якості та надійності акумуляторів з їх зменшенням розмірів до створення компактного та, відносно, потужного пристрою для перевезення вантажу у підприємствах, де потрібно це зробити швидко, та у невеликому приміщенні, але при цьому більше ваги і не навантажуючи людину занадто великою вагою. Інколи, можуть бути ситуації, коли людину не можна відправити для перенесення вантажу, тоді РТС можуть бути корисними і допомогти виправити ситуацію без використання людини у небезпечній ситуації.

Актуальність. Створення СКТВ для транспортування є досить важливим елементом для транспортування вантажу на відстані, що залежать від задачі, і при цьому зберегти життя людини при екстренних ситуаціях, з високим рівнем небезпеки. Також, РТС, що може доставляти товари до користувача в автономному режимі є доволі перспективним напрямком для підприємств з розвезення посилок на відносно великі дистанції. Інколи, людині буває важко підняти вантаж, але невелика система транспортування може трохи допомогти людині від ризиків носити важкий або складно підйомну систему з елементів на підприємстві. А з появою нових технологій, що дозволяють використовувати ефективніші електродвигуни, кращі акумулятори з можливістю використовувати більше енергії чи нові матеріали, з яких можна виготовити шестерні для редукторів і інших компонентів транспортного пристрою використовувати нові технології на морі, наприклад.

Мета роботи – створити прототип для покращення процесу використання серед підприємств систем контролю транспортування вантажів, що може зберегти здоров'я працівників або покращити рівень безпеки серед працівників, які

працюють в екримальній умовах. Полегшити використання РТС для перевезення вантажів на відстань в умовах, коли людина не зможе їх транспортувати вантаж без використання спеціальних систем для того, щоб прийти у невеликі місця, чи в місця де важко їх.

Об'єкт дослідження: транспортування вантажів в умовах обмеженої інфраструктурної доступності.

Предмет дослідження: методи апаратної реалізації First Point View (FPV) керування РТС з On-Screen Display (OSD) інформацією.

З теми випливає, що потрібно вирішити такі **задачі**:

– дослідити технічні характеристики існуючих СКТВ;

– проаналізувати нормативно-правові документи для розроблення системи контролю вантажоперевезення;

– сформувати технічні вимоги для створення СКТВ;

– обрати засоби апаратної реалізації вбудованих систем контролю транспортування вантажів;

– змоделювати принципову схему та схему підключення апаратних компонентів;

– створити механізм захоплення вантажів та фотофіксації перешкод;

– виконати експерементальні дослідження властивостей розробленої вбудованої системи.

Сфера застосування: в організаціях, що виконують транспортування вантажу у місцях з неможливістю доставлення навантаження за допомогою звичнойного керованого транспортного засобу та задопомогою людини, чи на роботах, пов'язаних з перенесенням важкого обладнання чи розвезення продукції по приміщенню чи ще деякої території приміщення.

Апробація результат кваліфікаційної роботи була проведена під час Всеукраїнської науково-практичної конференції «Могилянські читання – 2025: Досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» (м. Миколаїв 10–14 листопада 2025) [1].

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СИСТЕМИ ДЛЯ ТРАНСПОРТУВАННЯ ВАНТАЖІВ

1.1 Теоретичні відомості про вбудовані системи контролю перевезень вантажів

У порівнянні з класичними транспортними засобами, що можуть використовуватися для перевезень вантажів, роботизовані системи контролю транспортування використовуються у невеликих за площею місцях. Вони можуть перевозити вантажі на невеликі відстані у приміщеннях, складах, підприємствах чи на ділянках окремих невеликих населених пунктів. Але з розвитком сучасних технологій, як збільшення кількості циклів розряджання у акумуляторів, технологій передачі даних бездротового зв'язку їх використання збільшиться.

СКТВ мають декілька переваг над автомобілями, залізничним чи водним транспортом для перевезень вантажів. Однією з найголовніших переваг є компактність системи. Ця система зможе знаходитись у відносно вузькому приміщенні [2], де великий транспорт не зміг би ефективно функціонувати. Також невеликий розмір може допомогти платформі у роботі в важкодоступному місці. Така система маневреніша та може змінити маршрути за необхідності. Використання Internet of Things (IoT) передбачає зв'язок між СКТВ та користувачем через пульт. Це може бути інтерфейс з Wi-Fi чи Bluetooth. Саме така особливість може забезпечити безпечне виконання перевезень вантажів, без ризиків для людини у ситуаціях, коли відправляти людину не варто.

З точки зору доцільного використання ресурсів, така система може транспортувати вантажі, коли не доречно використовувати великі транспортні засоби. Вантажівка також може виконати це завдання, але людина має сісти за руль, в машину треба завантажити невелику кількість вантажу, і відправити цю машину лише для однієї точки. Таку задачу, інколи, доцільніше доручити РТС, що зможе зробити також перевезення, але швидше, і не маючи змоги відправляти грузову машину лише для відправки однієї невеликої посилки.

IoT-система для перевезень мають свої недоліки. Людину складно замінити у нестандартних ситуаціях. Кваліфікований працівник може самостійно приймати рішення в екстремальній ситуації, яку платформа мала б якось вирішувати. Це може бути механічне пошкодження, яке дуже складно полагодити на дистанційнокерованій платформі, що вже виїхала і проїхала майже всю відстань маршруту. Людина у разі критичної проблеми, може полагодити свою машину, або виконати базовий ремонт. Якщо ж транспорт не має змоги полагодити, людина сама може викликати спеціаліста з сфери обслуговування автомобілів і відправити свій транспортний засіб на ремонт.

Іншим недоліком СКТВ є комунікація з людьми. Така система не може ефективно передати інформацію іншій людині, чи коректно відповідати на всі питання. До того ж, РТС навіть з використанням інших наявних способів передачі інформації не можуть подивитися на ситуацію зі сторони людини. Сам працівник, може надати більше інформації і доступніше у випадку запитань, чи надати свої рекомендації за наявності у неї відповідної кваліфікації.

Людина обережніше поводиться з крихким вантажем, ніж система керування вантажем. Це зменшує ризик пошкодження корисного навантаження.

1.2 Наявні засоби транспортування вантажів

Є декілька видів СКТВ (рис. 1.1). Це можуть бутий наземні платформи – колісні та гусеничні роботи. Доволі швидко розвивається область перевезення цінних вантажів по повітрю. Також можуть зустрічатись роботизовані платформи для транспортування вантажів на поверхні води. Рідко можуть трапитись системи для транспортування вантжів під водою чи під поверхнею землі.

Колісні платформи є одними з найпопулярніших типів мобільних систем, що можуть використовуватись для транспортування вантажу. Цей спосіб транспортування є доволі поширеним, оскільки він компактний, може перевозити вантаж важчий, чим людина може підняти і має відносно простішу конструкцію. Може використовуватись у складах підприємств, виробничих приміщеннях, лабораторіях чи логістичних центрах. Інтеграція у виробничий процес,

передавання даних із датчиків та із використанням технологій IoT додають зручності у використанні платформи. Основою таких систем є двигуни і шасі, систему керування і вантажний відсік [3]. Є популярними двоколісні та чотириколісні системи. Таку систему набагато легше обслуговувати ніж інші типи через її простоту конструкції.

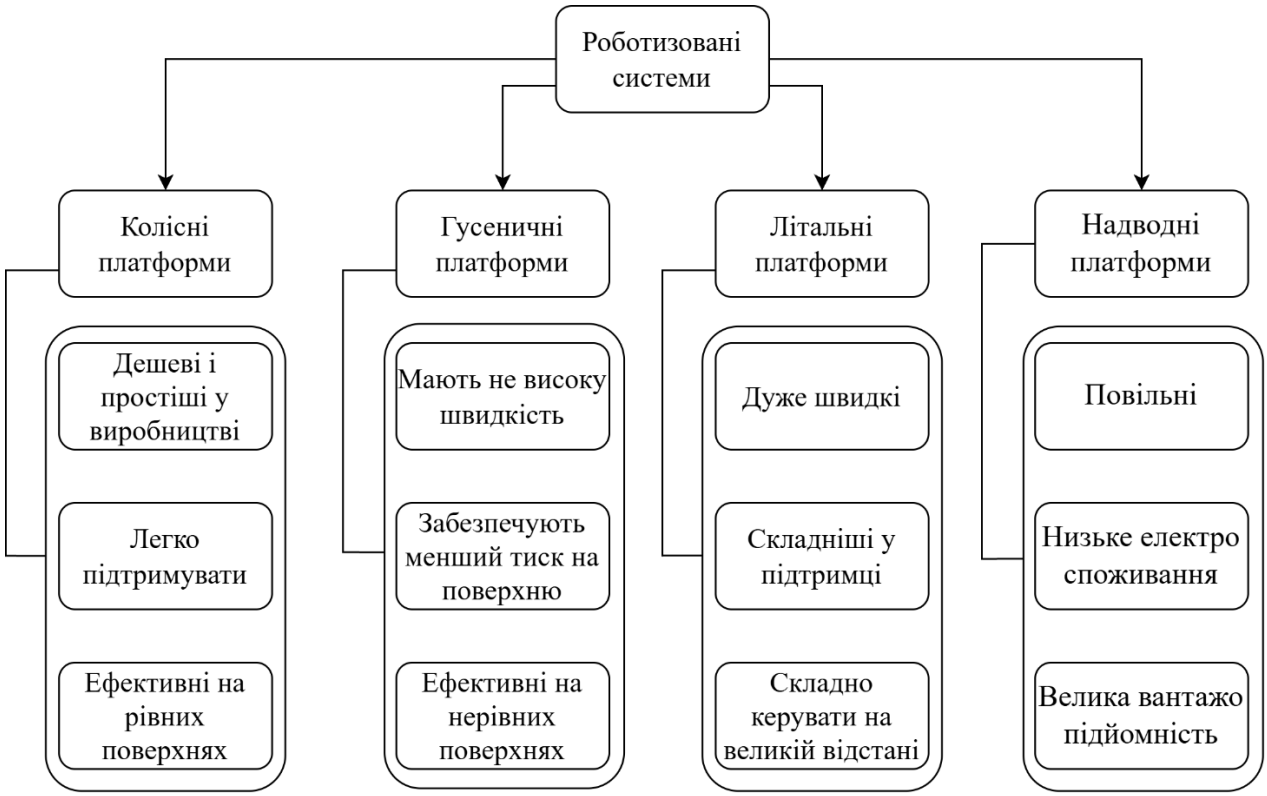

Рисунок 1.1 – Види РТС та їх ключові особливості

Колісні роботизовані платформи можуть працювати як в ручному режимі, так і дистанційного керування. Також, з'являються системи з автоновим керування, що можуть самостійно виконувати деякі завдання. У перших варіантах водій може керувати такою системою задопомогою пульта. Таким пультом може бути як складна система, так і програмне забезпечення, що може бути встановлене на телефон. В автономному режимі така платформа може пересуватися по заготовленим маршрутам [4].

Але у таких систем є і свої недоліки. Найнеприємнішим недоліком є знижена ефективність пересування по поверхням, що знаходяться під кутом. Також, слід

зважати на рівень прохідності місцевості. Наприклад, по вологій землі зчеплення буде набагато гірше, ніж на рівній гладкій поверхні підприємста чи на асфальті.

FOLO-4WD Automatic Following Car Robot (рис. 1.2) [5] – колісна платформа, що може демонструвати майже всі переваги колісних РТС. Це компактна платформа, що може перевозити до 150 кг. Маленькі габарити дозволяють йому виконувати завдання у місцях де транспорт не може зробити перевезення вантажу.

Рисунок 1.2 – Демонстрація роботи FOLO-4WD Automatic Following Car Robot [5]

Акумулятори можуть забезпечувати РТС протягом 6 годин із навантаженням 80 кг. Керується дистанційно з максимальною дальністю до 30 м. Має систему для уникнення перешкод, дистанція для зупинки при виявленні перешкоди 10 см. Особливим недоліком є швидкість платформи – 6 км/год, порівнюючи з такими ж колісними РТС, має відносну не високу швидкість. Слідувати за людиною можливість є, але виїхати швидше ніж за звичайний транспорт не зможе. Проходимість не висока, але для руху по прямій поверхності ця РТС може розігнатися до максимальної швидкості та виконувати свої завдання.

FOLO-300 4WD Automatic Wheeled Following Robot 300KG (далі FOLO-300 4WD) [6] схожа платформа. Має багато спільних рис. Середних це колісна база

платформи. Це дає їй переваги на рівній поверхні та на поверхнях у приміщеннях. Враховуючи свої габарити, така платформа може перевозити вантаж вагою до 300 кг. Це доволі багато для такої платформи, що також дає їй змогу пересуватися у приміщенні чи на території підприємства. Хоча розміри цієї платформи все ж більші, ніж у FOLO-4WD Automatic Following Car Robot. Акумулятори забезпечать роботу платформи приблизно 5 годин. Відстань керування 30 метрів.

Рис. 1.3 – Вигляд

Серед переваг такої платформи є наявність датчику перешкод. Можна забезпечити те, щоб платформа не врізалась з перешкодою. Велика перевага платформи у тому, що вона може їхати на відстань 45 км.

Недоліком такої платформи є її довга зарядка. Така зарядка відбувається приблизно 7–8 годин. Головний недолік платформи це її швидкість. Вона приблизно 4 км/год. Також серед недоліків платформи є пересування по нерівній поверхності, всього лише під кутом приблизно 10°, це варто враховувати під час складення маршрутів. Тому така платформа не є всюдихідною.

Mobile Ranger 4WS 4WD Robot (UGV) від Agelix [7] має схожі характеристики. Він має приблизно схожі розміри з FOLO-300 4WD Automatic Wheeled Following Robot 300KG. Але вагу, що може підняти, має меншу – 150 кг. Також, може бути використовуваний у різних місцях з рівною поверхнею. Це можуть бути приміщення чи склади з вантажами. Заряду може вистачити на 2–8 годин роботи.

Перевагою такої платформи є його швидкість. Значення швидкості може бути приблизно 9,36 км/год. До того ж є швидке заряджання батареї. В середньому цей процес займає 1 годину.

З недоліків можна вибрати те, що платформа може швидко розряджатися, мінімальна кількість часу тримання заряду зазначена в 2 години. Також поверхня залежить від поверні на якій їздить. Кут такої поверхні може бути не більше 10° на нерівних поверхнях.

Рисунок 1.4 – Вид Mobile Ranger 4WS 4WD Robot (UGV) від Agelix [7]

ot (далі FOLO-100 PRO) [8] – також колісна платформа. Перевозити вона може 100 кг. Розмірність такої платформи невисока. Власне такі характеристики дозволяють її використовувати у приміщенні чи складі, при цьому, можна використовувати спеціальний транспорт, що керується людиною у випадках з дійсно габаритними вантажами. Зберігає заряд впродовж до 8 годин.

Однією з головних переваг є те, що має можливість швидко реагувати на перешкоди та вчасно зупинитися. Дистанція виявлення перешкоди до 30 м. Керувати можна ведучи за собою чи з пульта. Також є швидкою зарядка – приблизно 4 години.

Недоліком є швидкість перевезень. Ця швидкість може сягати невисоких вимірів – 5 км/год. Також така платформа не може пересувати ефективно по нерівних поверхнях, всього лише кут ефективної роботи приблизно 10° на похилих поверхнях.

Рисунок 1.5 – Вигляд платформи FOLO-100 PRO [8]

Ще одним видом наземного пересування вантажів є використання гусеничних СКТВ. Такі системи мають замість колес гусениці. Вони ефективніші ніж колісні системи, бо використовують гусеничний механізм, що має більше площу котакту з поверхнею, ніж колеса. Завдяки цьому, такі системи мають кращу прохідність по нерівній місцевості, піску та землі [9]. Тому гусеничні платформи використовують набагато частіше у місцях, де не можуть впоратися колісні системи. Основою такої системи є шасі, електронні модулі та вантажна платформа. Звичайний рух здійснюється за допомогою двох гусеничних стрічок, які приводять в дію незалежні електричні двигуни.

Ще однією перевагою під час транспортування такої системи є стабілізація ходу [10]. Саме гусениці її дають. Це дасть перевану транспортуванні крихких вантажів або нестійнких вантажів.

Такі системи виконують майже такі самі задачі, що і колісні. Керуються також, або в ручному режимі, чи через дистанційне керування. Є і системи, що також можуть керуватися і в автономному режимі. Пультом може бути як сама система, що передає радіохвилі до платформи за допомогою натискання кнопок, так і вебінтерфейс.

З недоліків такої системи є складність конструкції. Її дорожче виготовити, так і підтримувати. Вони мають більшу вагу, що збільшує енергоспоживання. А гусениці мають самі механічних механізми, що потрібно перевіряти і обслуговувати частіше, бо вони піддаються великому навантаженню.

er Mini (рис. 1.3) [11] – гусенична платформа, що використовується для перевезень

вантажів. Його основою перевагою, є його можливість рухатися по нерівній поверхні з кутом нахилу до 30°. Це майже втричі більше, ніж у колісної платформи FOLO-4WD Automatic Following Car Robot при максимальному завантаженні. Має акумулятори ємності 30 АН, через що може працювати 80 хвилин. Має максимальну швидкість перевезення 1 м/с. Головним недоліком, є його невелика вантажопідйомність – всього 25 кг, майже в 6 разів менше ніж у колісної платформи, але при цьому має вагу у 56 кг. Має двигуни невеликої потужності – 2 двигуни по

Рисунок 1.6 – Вигляд Button Mini [12]

Таблиця 1.1 – Загальні характеристики платформ для перевезень вантажів

Назви платформ	**FOLO-4WD Automatic Following Car Robot**	**FOLO-300 4WD**	**Mobile Ranger 4WS 4WD Robot (UGV)**	**FOLO-100 PRO**	**Bunker Mini**
Тип платформи	Колісна	Колісна	Колісна	Колісна	Гусенична
Відстань роботи, м	30	30	Не вказано	30	200
Час роботи, хв	360	300	Від 120 до 480	Від 360 до 480	80
Напруга акумулятора, В	36 (при 15 Аг)	48 (при 60Аг)	Не вказано	24 (при 20Аг)	30
Максимальна швидкість перевезення вантажу, км/год	6	4	9,36	5	3,6
Вага натажу, кг	150	300	150	100	25
Дальність ходу, км	Не вказано	45	Не вказано	20	14

Колісні платформи ефективні для швидких перевезень вантажів, гусеничні для перевезень з непрямою поверхнею, літальні платформи дуже швидкі для легких вантажів, а надводна для максимальної ефективності усіх характеристик (табл. 1.1).

1.3 Нормативні документи системи контролю транспортування вантажів

Розробка РТС, де використовуються бездротові мережеві технології і базуючись на таких технологіях як IoT вимагають комплексного підходу. Це дуже важлива складова для проєктування невеликих за розміром систем для перевезення вантажу з урахуванням безпеки для людей та непорушності безпечних ситуацій у приміщеннях. Згідно з європейської та української бази нормативних документів, для таких пристроїв можна застосовувати вимоги технічних регламентів. Це важливо, бо така система використовує радіообладнання, низьковольтне обладнання та інші міжнародні стандарти для передачі даних.

Основним способом керування СКТВ є використання радіозв'язку. Тип такого радіозв'язку є технологія Wi-Fi з діапазоном частоти 2,4 ГГц. Використання такого роду пристроїв забезпечується Технічним регламентом радіообладнання, що затверджений Постановою КМУ від 15 березня 2022 року №355 [13]. За цим регламентом треба щоб радіообладнання розроблялось так, щоб робити менше завад іншим пристроям. Згідно закону України «Про електронні комунікації» [14], використання діапазону 2,4 ГГц, що є технологією IEEE 802.11 b/g/n підпадає під визначення для загального користування цим радіочастотним спектром. Цей спектр має не перевищувати потужність випромінювання 100 мВт або 20 дБм. Також така технологія повинна гарантувати, що рівень електромагнітного випромінювання є безпечним для людського здоров'я.

Двигуни використовують широтно-імпульсну модуляцію. Але двигуни у такому випадку є джерелом потужного електромагнітного шуму під навантаженням. Для контролю таких процесів є Технічний регламент з електромагнітної сумісності обладнання від постанови КМУ від 16 грудня 2015 року №1077 [15]. Він вимагає проєктування з урахуванням вимог до сучасного

рівня розвитку техніки. Це захист від емісії завад. Іскріння на електрощітках двигуна можуть зробити так, що WiFi-роутер просто не зможе передавати радіосигнал до пристроїв без завад. Тому паралельно до контактів двигуна треба спаяти конденсатор, як правило номіналом 0,1 мкФ . Частина з логікою самого мікроконтролера має бути із захистом від завад, тобто заземлена. Також можна використовувати електролітичні конденсатори, вони можуть допомогти зі згладжуванням пульсацій струмів.

Дуже важливим є і захист системи від високої напруги. Система живиться від літієвих акумуляторів, напруга в яких 3,7 В, 5 В, 12 В. З часом вона розряджається. І треба заряджати її знову. Саме це і є системою для контролювання технічним регламентом низьковольтного обладнання, що є постановою КМУ від 16 грудня 2015 року № 1067 [15]. Але регламент розповсюджується на системи, де постійний струм має напругу від 50 до 1000 В змінного струму і від 75 до 1500 В постійного струму. Тому така система має мати гальванічну розв’язку і ізоляцію. Треба звернути на імпульси, що можуть йти від мережевого адаптеру, тому модель має бути надійною до цього. Також має бути гальванічна розв'язка, або ізоляція на металевих частинах шасі, задля того, щоб заряд не потрапляв на ділянки шасі і не вивів з ладу мікроконтролер. Також варто використовувати Battery Management System (BMS) для того, щоб захистити систему від короткого замикання чи від перезаряду і розряду акумуляторів, що може їм нашкодити.

Так як у роботі використовується протокол HTTP, то має бути забезпечена і безпека. Цим займається система протоколів Закону України «Про захист інформації в інформаційно-комунікаційних системах» №80/94-ВР [16]. Отже, і Закон України «Про основні засади забезпечення кібербезпеки України» № 2163-VIII [17] визначає такі об'єкти як IoT потенційно тими, що потребують певний базовий захист. Для цього можна авторизуватись на рівні мережі. Точка доступу, що створює мікроконтролер, чи підключення до роутеру має бажано мати захист рівня WPA2-PSK. Прошивка мікроконтролера має включати алгоритм роботи оброки виключних ситуацій. Наприклад, система отримала пошкоджений

HTTP-запит чи формат передачі даних JSON, що не відповідає тому, що очікує для отримання мікроконтролер, мікроконтролер має зупинити рух.

1.4 Технічні вимоги для створення СКТВ

Для забезпечення успішної реалізації проєкту, а також для гарантування надійності, безпеки та масштабованості роботизованої платформи, необхідно сформувати чіткий перелік технічних вимог. Ці вимоги розділені на функціональні, апаратні, програмні, системи живлення, мережевої взаємодії, програмного забезпечення (ПЗ), та експлуатаційні підгрупи, що дозволяє формалізувати процес розробки на кожному етапі.

Функціональні вимоги:

- СКТВ повинна мати дистанційне ручне керування в режимі реального часу з використанням бездротового зв’язку;
- система має передавати дані про наявність з’єднання із клієнтом;
- архітектура повинна мати модульний характер, щоб мати можливість її підтримувати та модифікувати без повної переробки базової платформи.

Вимоги до апаратного забезпечення:

- мати обчислювальне ядро із вбудованим апаратним модулем Wi-Fi. Це дозволить об'єднати логіку обробки роботи сенсорів та мережевий стек на одному кристалі, якщо треба, зменшивши габарити;
- мати підсистему приводів: для забезпечення точного керування сервомоторами варто використовувати спеціальний ШІМ-контролер для роботи з сервомоторами. Він використовує передачу даних по шині I2C. Це допоможе розвантажити головний процесор та забезпечить стабільність керівних імпульсів;
- драйвери силової частини: це може бути Н-міст для керування електродвигунами, що мають вбудований тепловий захист та захист від зворотних струмів індукції;
- використовувати сенсори для контролю пересування по місцевості для виявлення перешкод у фронтальній частині шасі.

Вимоги до системи живлення:

− використання Li-Ion або Li-Pol акумуляторів для живлення пристроїв;

− захист акумулятора: це BMS при роботі з акумуляторами, вони можуть захистити від перевантаження системи чи короткого замикання. Системи з розбалансуванням можуть допомогти заряджати акумулятори;

− має бути розподіл напруги: треба живити мікроконтролер напругою 5 В або 3,3 В від стабілізаторів напруги. Це мають бути DC-DC перетворювачі і захисті його від перевантаження під час стрибків напруги;

− мати спеціальний зарядний модуль для системи живлення CC/CV з утриманням значення струму, що є безпечним для заряду акумулятора та напруги.

Вимоги до мережевої взаємодії:

− мати підключення через Wi-Fi стандарт 802.11 b/g/n при діапазоні 2,4 ГГц. При можливості, платформа повинна мати можливість створювати власну точку доступу чи підключатися до існуючої локальної мережі.

Вимоги до ПЗ:

− створити програмну частину для обробки команд з клієнтської частини;

− формат повідомлень: усі дані між вебінтерфейсом та платформою мають серіалізуватися та передаватися у легкому текстовому форматі JSON, що забезпечує швидкий парсинг на стороні мікроконтролера.

Вимога до клієнтського інтерфейсу – кросплатформність. Інтерфейс керування повинен бути реалізований у вигляді адаптивного вебзастосунка за допомогою HTML, CSS, JavaScript, який коректно відображається як на екранах ПК, так і на мобільних пристроях і може бути універсальним, навіть для різних контролерів.

Висновки до 1 розділу

У першому розділі було розглянуто аналіз сучасного стану, методів та засобів автоматизованого транспортування вантажів. Дослідження тенденцій розвитку логістики в контексті концепції Інтернету речей показало, що гнучкі мобільні роботизовані платформи є одним із розповсюджених фактором оптимізації сучасних виробничих та складських процесів. Такі системи можуть допомогти

людині зберегти життя та транспортувати вантаж у випадках, коли отримати транспортний засіб не може переправити навантаження. Огляд існуючих комерційних рішень виявив, що більшість з них з високою вартістю впровадження, що робить їх економічно недоцільними для локальних. Це обґрунтовує актуальність розробки доступної мобільної платформи з відкритою, модульною архітектурою.

Дослідження нормативно-технічної бази підтвердило необхідність системного підходу до проєктування IoT-пристроїв. Визначено, що апаратна та програмна реалізація розроблюваної системи повинна спиратися на актуальні українські та міжнародні стандарти. Зокрема, гарантування безпеки експлуатації вимагає дотримання норм електробезпеки низьковольтного та пожежної безпеки систем керування літієвими акумуляторами.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ СИСТЕМИ ДЛЯ ТРАНСПОРТУВАННЯ ВАНТАЖІВ

Для проєктування вбудованої системи контролю перевезення вантажів необхідно правильно підібрати усі комплектуючі проаналізувавши наявні на ринку та вибрати найоптимальніші. Це допоможе обрати оптимальне апаратне забезпечення з урахування їх можливостей та ціни. Серед таких комплектуючих треба визначити такі: мікроконтролер, двигуни, камера, датчик відстані, драйвер двигунів. Такі комплектуючі допоможуть створити необхідну концепцію використання такої платформи. Ключове місце займає вибір програмного забезпечення, з урахуванням властивостей мови та наявних бібліотек.

Окрім вибору комплектуючих необхідно і визначити з самим принципом роботи СКТВ. Описати її важливі механізми та процеси, що відбуваються під впливом різних чинників. Однією з ключових рішень є створення принципової схеми та схеми підключення. За допомогою них можна набагато легше візуалізувати роботу цієї платформи.

2.1 Вибір мікроконтролера та камери для СКТВ

Мікроконтролер у вбудованій системі контролю транспортування вантажів буде відігравати одразу декілька ролей. Спершу, він буде відповідати за зв'язок з людиною, що керує цією системою. За допомогою інтерфейсу будуть відсилатися команди до вбудованої системи транспортування, де вони будуть оброблятися та давати команди іншим компонентам платформи. Серед цих завдань буде контроль роботи двигунів, їх швидкість, напрямок роботи і керування самими моторами.

Іншим завданням буде збирання даних з датчика дистанції. Саме за допомогою цих даних буде визначатися відстань до найближчої перешкоди і вибору дії : зупинки чи продовження руху. Якщо дистанція до певної перешкоди буде 150 см буде зроблена зупинка платформи, якщо ж відстань більша, то зупинки не буде. Ключовою функцією для керування апаратної платформи є елемент керування за допомогою сервомотора вліво або вправо. Це треба робити за

допомогою ШІМ-контролера PCA-9685 [18]. Також цей ШІМ-контролер буде використовуватися для контролю роботи обертання самого датчика відстані та клешні-маніпулятора.

З можливостей, такий мікроконтролер або одноплатний комп'ютер повинен мати підтримку таких протоколів як SPI, I^2C і UART. Для програмного забезпечення варто, щоб були наявні бібліотеки або рекомендації від виробників для роботи з одноплатним комп'ютером чи мікроконтролером.

ESP32-CAM (таблиця 2.1) [19] двохядерний мікроконтролер, але ESP32-S-CAM може бути і одноядерним мікроконтролером. Такі мікроконтролери варті мати стабільне живлення 5 В зі струмом для забезпечення роботи як і самого мікроконтролера, так і камери для нього. Серед його переваг, це його низьке енергоспоживання. Сумарна споживана сила струму менше 1 А. Має найнижчі частоти процесора. Це 240 МГц або 160 МГц, якщо це ESP32-S-CAM. При двох ядрах може використовуватись одне ядро для самої роботи програми, а друге для роботи з мережею Wi-Fi.

З недоліків є малий обсяг оперативної пам'яті. Всього лише 4 Мбайт. Цього може не вистачити для роботи вебсторінки та камери одночасно. Також така система не підтримує повноцінно Linux, тому використовувати логи може бути незручно для іншого розробника. Окрім цього, проблемою може сама потужність. Не буде Frame Per Second (FPS) при нестабільному живленні і великій якості зйомки. Тому варто обирати перевірену систему живлення з надійною платою BMS та правильною схемою живлення. Також вбудована антена не має великої потужності та всього лише 25 дБ. Цього може не вистачати на великі відстані. Але це можна виправити, використовуючи окрему антену.

Таблиця 2.1 – Характеристики ESP32-CAM

Контролер	BL618
Кількість ядер, шт	2
Оперативна пам'ять	Вбудована: 520 кбайт, Зовнішня: 4 Мбайт
Підтримувані інтерфейси	UART, SPI, iec, PWM, ADC, DAC
Напруга живлення, В	5
Підтримка камер	OV2640, OV7670
Зберігання даних	micro-SD карта пам'яті
Розмір, мм	27 × 40
Вага, г	10

Raspberry Pi 3 (табл. 2.2) [20]– одноплатний комп'ютер з процесоорм на базі SoC Broadcom BCM2837B0 з 4 ядрами Cortex-A53, 64-bit, 1,4 ГГц. Серед переваг цього одноплатного комп'ютера можна виділити стабільність. Це дуже популярна плата для роботи з вбудованими системами і поширена серед користувачів, тому містить багато бібліотек. Підтримує багато протоколів периферії, таких як GPIO, I2C, SPI, UART на рівні офіційного ядра ОС. До того ж, для нього доступні Linux-дистритбутиви, з великим вибором потрібних для роботи. Має свій роз'єм для модуля камери, що забезпечує одночасну швидку передачу даних.

Серед недоліків можна виділити великий розмір одноплатного комп'ютера. Він доволі великий, порівняно з ESP32-CAM, тому у невеликі системи транспортного контролю його важко вмістити і слід це враховувати під час проєктування СКТВ. Також Raspberry Pi 3 має одразу 4 порти USB type-A, роз'єм RJ-45, що не використовуються під час роботи. Це створює проблеми, оскільки вони споживають електроенергію навіть в автономному режимі. Таким чином, це знижує коефіцієнт корисної дії платформи.

Таблиця 2.2 – Характеристики Raspberry Pi 3

Кількість ядер, шт	4
Частота процесора, ГГц	1,2
Контролер	BCM2837
Кількість оперативної пам'яті, Гбайт	1
Покоління оперативної пам'яті	LDDR2
Ethernet	100 Base Ethernet
Кількість пінів, шт	40
Кількість USB портів, шт	4
Кількість HDMI портів, шт	1
Роз'єм для камери	CSI
Кількість роз'ємів для камери, шт	1
Наявність SD-карти	Є
Споживаний струм, А	2,5

Orange Pi 3 (табл. 2.3) [21] – одноплатний комп'ютер, що має 4 ядра Cortex A-53 з частотою 1,8 ГГц з 2 Гбайт пам'яті LPDDR3 і інтегрованим модулем eMMC-пам'яті на 8 Гбайт. Це доволі серйозна перевага, оскільки не обов'язково для зберігання невеликого обсягу даних купувати SD-карту. Також eMMC-пам'ять трошки швидша за SD-карту.

Серед недоліків є вище енергоспоживання і низька стабільність апаратних таймерів. Для точного керування ШІМ-сигналами треба базова частота таймерів. В архітектурі Allwinner H6 системні таймери можуть мати схильність до початку мікросекундних дрейфувань. Це може викликати мікроскопічні коливання швидкості обертання коліс, що знизить точность позиціонування вже платформи.

Таблиця 2.3 – Характеристики Orange Pi 3

Кількість ядер, шт	4
Частота процесора, ГГц	1,8
Контролер	Allwinner H6
Кількість оперативної пам'яті, Гбайт	2
Покоління оперативної пам'яті	LPDDR3
Ethernet	10M, 100M, 1000M
eMMC-пам'ять, Гбайт	8
Кількість USB портів, шт	3
Кількість HDMI портів, шт	1
Роз'єм для камери	CSI
Кількість роз'ємів для камери, шт	1
Наявність SD-карти	Є
Споживаний струм, А	3

Orange Pi Zero 2W (табл. 2.4) [22] – одноплатний комп'ютер, що побудований на базі чотириядерного процесора Allwinner H618 з архітектурою ARM Cortex-A53. Тактова частота такого процесора 1,5 ГГц. Має пам'ять LPDDR4 з об'ємом 1 Гбайт. Як переваги, такий одноплатний комп'ютер має відносно маленькі розміри, і при цьому має високу обчислювальну потужність. Споживає трохи менше струму, ніж Orange Pi 3. Має більш сучасний тип ядра.

Головними недоліками можуть бути нестабільність самої програмної підтримки. Офіційні збірки ОС використовують модифіковані застарілі гілки ядра Linux. Також були знайдені помилки у роботі з конфігурацією самих ШІМ-сигналів, що може ускладнити роботу з двигунами напряму, якщо неможливо використати ШІМ-контролер.

Таблиця 2.4 – Характеристики Orange Pi Zero 2W

Кількість ядер, шт	4
Частота процесора, ГГц	1,5
Контролер	Allwinner H618
Кількість оперативної пам'яті, Гбайт	1
Покоління оперативної пам'яті	LPDDR4
Ethernet	100M при наявності Ethernet-хабу
Кількість пінів, шт	40
Кількість USB портів, шт	2
Кількість HDMI портів, шт	Відсутні
Роз'єм для камери	Відсутні
Кількість роз'ємів для камери, шт	Відсутні
Наявність SD-карти	Є
Споживаний струм, А	2

Raspberry Pi Zero 2W (рис. 2.1, табл. 2.5) [23] – одноплатний комп'ютер, що має 4 ядра побудованих на архітектурі ARM Cortex-A53. Вони мають частоту 1 ГГц. Сама архітектура має пам'ять LPDDR2 з розміром 1 Гбайт. Підтримує багато низькорівневих протоколів, таких як GPIO, PWM, I2C, SPI і UART. Серед переваг такого одноплатного комп'ютера можна виділити наявність CSI роз'єму для підключення камери напряму до Raspberry Pi Zero 2W. Це підвищує швидкість передачі даних, ніж це робити через USB, що має покоління USB 2.0. Також це доволі невелика плата, що може вміститися у невеликій платформі. Підтримує Wi-Fi 4 та Bluetooth.

Серед недоліків є USB з поколінням 2.0. Вони мають не високу швидкість передачі даних. Головним мінусом цього одноплатного комп'ютера є кількість оперативної пам'яті. Всього 512 Мбайт, що може викликати проблеми під час використання різного ПЗ. Також серед мінусів може бути споживання – 2 А.

Таблиця 2.5 – Характристики Raspberry Pi Zero 2W

Кількість ядер, шт	4
Частота процесора, ГГц	1
Контролер	BCM2710A1
Кількість оперативної пам'яті, ГБ	1
Покоління оперативної пам'яті	LPDDR2
Ethernet	10M, 100M
Кількість пінів, шт	40
Кількість USB портів, шт	2
Кількість HDMI портів, шт	Відсутні
Роз'єм для камери	CSI
Кількість роз'ємів для камери, шт	1
Наявність SD-карти	Є
Споживаний струм, А	2,5

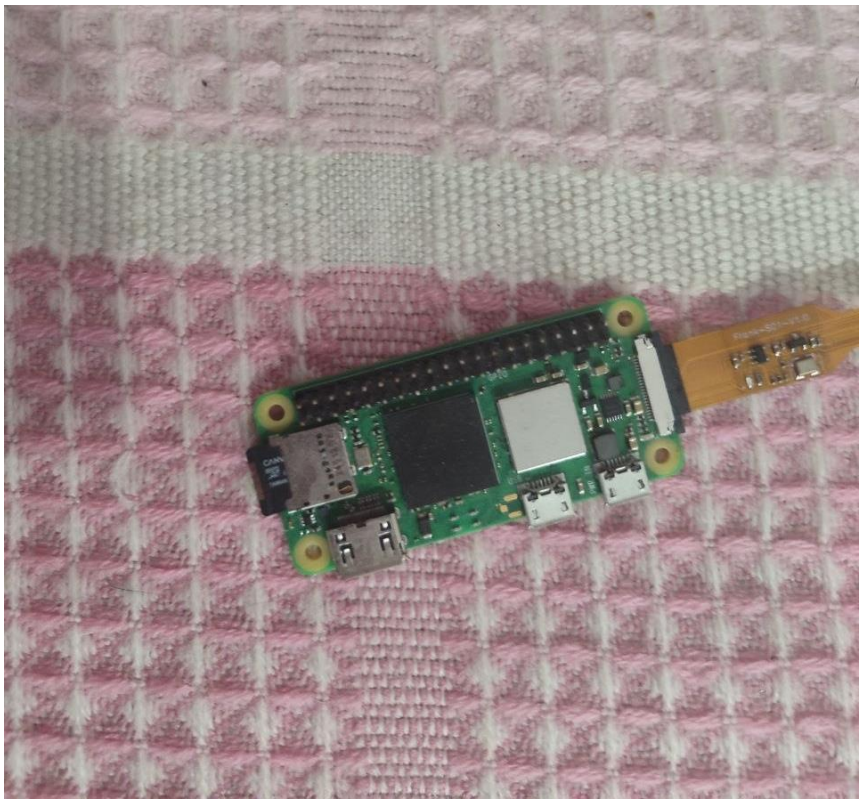

Рисунок 2.1 – Вигляд Raspberry Pi Zero 2W

Таблиця 2.6 – Основні характеристики усіх одноплатних комп'ютерів

Одноплатний комп'ютер	**ESP32-CAM**	**Raspberry Pi 3**	**Orange Pi 3**	**Orange Pi Zero 2W**	**Raspberry Pi Zero 2W**
Кількість ядер, шт	2	4	4	4	4
Обсяг оперативної пам'яті	512 кбайт і 4 Мбайт зовнішньої	1 Гбайт	2 Гбайт	1 Гбайт	512 Мбайт
Роз'єм для камери	FPC	CSI	CSI	відсутній	CSI
Споживаний струм, А	0,5	2,5	3	2,5	2,5
Ціна, грн.	537	1547,37	4700	2669	1326

Отже, вибір Raspberry Pi Zero 2W є найдоречнішим, оскільки ця плата невеликі розміри, при відносно невисокому енергоспоживанню. Orange Pi Zero 2W має схожі параметри, але коштує в рази 2 дорожче, при цьому немає роз'єму для камери. ESP32-CAM має менші розміри і невелике енергоспоживання, але її обчислювальна потужність буде меншою, ніж у Raspberry Pi Zero 2W, хоча вона і дешевша в 2 рази. Pasbperry Pi 3 і Orange Pi 3 мають схожі розміри між собою, але вони значно більші.

Для роботи системи контролю транспортування різних вантажів необхідно визначити можливі перешкоди, що можуть трапитись на шляху до точки відвезення вантажу. Також це використовується для моніторингу навколишнього середовища і забезпечує надійність дистанційного керування такою платформою. Для цього можна використати камеру, що можна під'єднати до Raspberry Pi Zero 2W. Для того щоб це зробити така камера має декілька значних відповідностей. По-перше це роз'єм типу CSI, що знаходиться на одноплатному комп'ютері. По-друге мати відповідне розширення. Без цього можна буде не розібрати що відбувається поза зоною бачення людини, що керує цією платформою. По-третє це вплив на обчислювальні можливості. Якщо камера буде мати доволі високу деталізацію, це може вплинути на FPS, що може бути одним з ключових факторів під час керування у серйозних умовах.

Raspberry Pi Camera Module v1.3 (рис. 2.2) [24] один з найпопулярніших модулів камери для Raspberry Pi Zero 2W. Він базується на сенсорі OmniVision

OV5647, що є доволі бюджетним і розповсюдженим серед користувачів. Роздільна здатність 5 Мп і фіксованим фокусом. Серед переваг низьке енергоспоживання, невеликі розміри і сумісність з усіма операційними системами. А недоліком є застаріла матриця. В умовах з поганим освітленням, інколи, можна навіть не помітити деякі важливі деталі, які б могли бути важливими. А шуми не дадуть використовувати доволі ефективно технології зі штучним інтелектом.

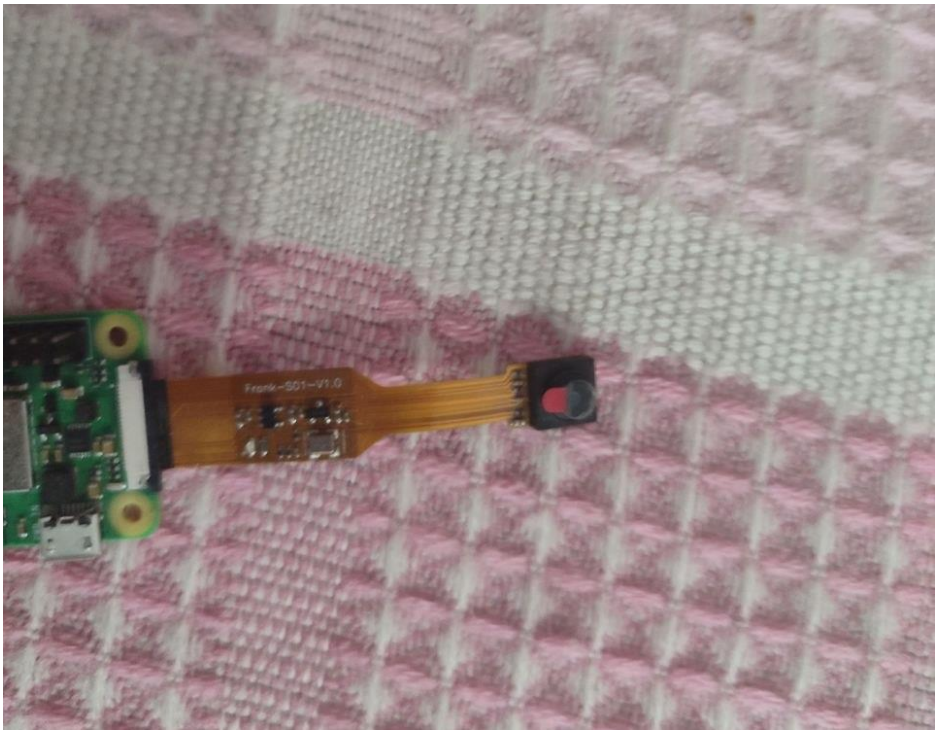

Рисунок 2.2 – Вигляд камери Raspberry Pi Camera Module V1.3

Raspberry Pi Camera Module v2 (рис. 2.3) [25] має матрицю 8 Мп і краще передає кольори, ніж Raspberry Pi Camera Module v1.3. Має оптимальний баланс між роздільною здатністю та швидкістю обробки інформації. Може активно працювати у форматі 720p при 60 FPS та 1080p при 30 FPS. Підтримує сучасні бібліотеки libcamera і має невелике навантаження на систему, завдяки апаратній обробці на ISP відеочіпа VideoCore IV. Недоліком є відсутність автофокусу. Це можна виправити, якщо налаштовувати цей параметр вручну.

Рисунок 2.3 – Вигляд Rasbperry Pi Camera Module V2.1 [26]

Raspberry Pi Camera Module 3 (рис. 2.4) [27] є однією з найдорожчих камер для Rasberry Pi Zero 2W. Вона базується на базі модуля Sony IMX708. Через це вона має роздільну здатність 12 Мп. Також має підтримку апаратного HDR та швидкісного фазового автофокусу. Має чудове зображення навіть в темних приміщеннях, миттєво фокусується на об'єкті, і на тому, що наближується і підтримує широкий динамічний діапазон. Головним недоліком є через технологію швидкісного діапазону, вимагає додаткових обчислювальних потужностей. Враховуючи, що Raspberry Pi Zero 2W не має їх багато, то це може бути інколи проблема.

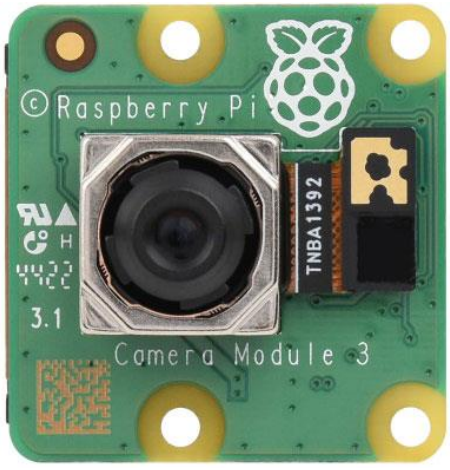

Рисунок 2.4 – Вигляд Raspberry Pi Camera Module v3.1 [28]

Для використання вибрана камера Raspberry Pi Camera Module v1.3, оскільки вона має декілька переваг. Найголовніший це її простота. Вона доволі розповсюджена і при цьому дешева. Іншою головною перевагою навантаження на

відеочіп. Воно не велике, бо не вимагає інших додаткових функцій, як у Raspberry Pi Camera Module 3 з автофокусом, але не і не має додаткового модуля, як у Raspberry Pi Camera Module V2, що може зменшити навантаження на графічне ядро. І перевагою є її якість зйомки, всього 5 Мп і воно зменшує саме навантаження на ядра процесора, даючи можливість використовувати ці ресурси в інших цілях.

2.2 Вибір моторів та драйвера двигунів для СКТВ

Електродвигун це компонент, що проводить в рух апаратну платформу. Саме за допомогою нього можна визначити з якою швидкістю буде переміщуватися платформа на поверхні. Скільки буде споживати при навантаженнях і під час поїздки без. І головне, скільки зможе підняти на собі і везти далі для виконання завдань з вантажоперевезень. Для тестувань роботи таких платформ було використані 3 щіткових двигуни, оскільки вони мають нижчу вартість за безщіткові двигуни. Недоліком таких електродвигунів є те, що вони можуть мати нижчий ККД та швидше виходити з ладу через потрапляння бруду і пилу до щіток. Безщіткові набагато надійніші, мають вищий ККД, але керуються через спеціальні контролери і мають вищу вартість, коли щітковий двигун може керуватися за допомогою окремого транзистора.

Мотор з металевим редуктором (рис. 2.5) [29] є доволі надійним вибором. Він має високу надійність через матеріали, з яких він зроблений. Він може витримувати велике навантаження на свій редуктор. Серед його переваг також є низьке електроспоживання, всього лише 120 мА при напрузі 6 В. Серед недоліків такого мотора є його крутний момент, це значення лише 0,2 кг/см. Але при цьому він може робити 1000 обертів за хвилину без редуктора.

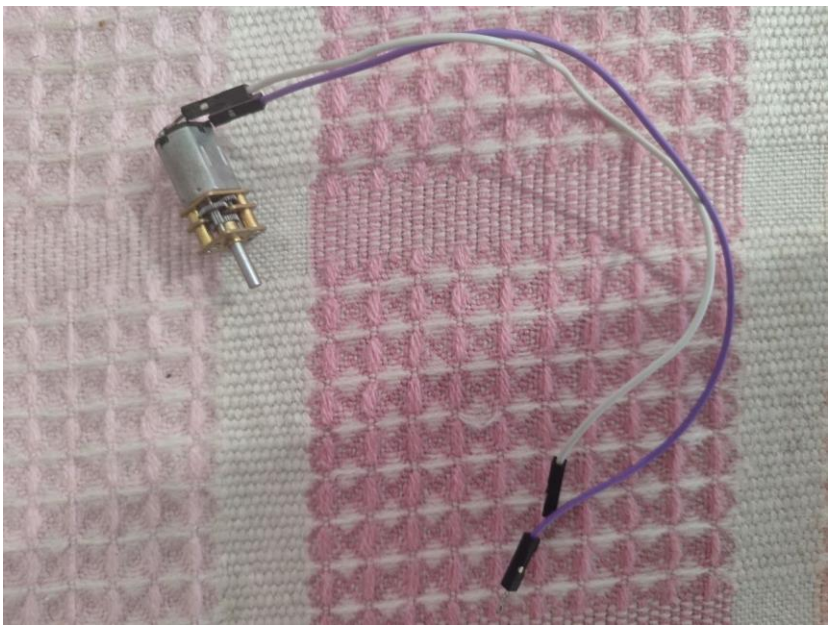

Рисунок 2.5 – Вигляд мотору з металевими редукторами

Мотор без редуктора ASM1117 (рис. 2.6) [30] набагато більший. Він не містить редуктора, тому його кількість обертів доволі висока. Заявлено, що він може давати 9000 обертів на хвилину. Але без редуктора, його крутний момент невеликий, тому везти вантажі, не створюючи нову систему підвищення крутного моменту буде незручно. Також він може споживати 300 мА при напрузі 3 В, це більше ніж у мотора з металевим редуктором.

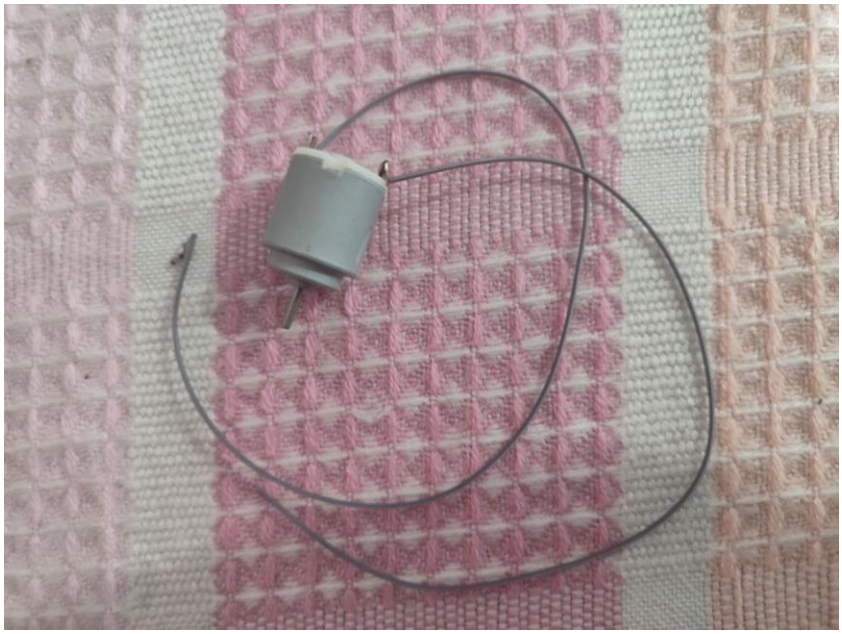

Рисунок 2.6 – Мотор ASM1117 без редуктора

Мотор з пластиковим редуктором (рис. 2.7) [31] є дуже поширеним прикладом компонентів для створення роботів на базі Arduino та ESP8266 чи ESP32. Він може переміщувати вантаж з більшою вагою, з крутним моментом 800 г/см. Споживає до 250 мА при напрузі 3,6 В. Може використовувати напругу 6-8 В. Головним недоліком є його кількість обертів за хвилину. Через використання редуктора вона не висока, тому розраховувати на швидкість самої платформи не сильно варто.

Рисунок 2.7 – Мотор з пластиковим редуктором [31]

Драйвер двигунів є Н-мостом для щіткових двигунів. Вони створені з транзисторів і дозволяють керувати таким контролер за допомогою слабших сигналів з одноплатного комп'ютера.

Драйвер L298N (рис. 2.10) [32] – класичний двоканальний Н-міст на біполярних транзисторах, що є стандартом завдяки апаратній надійності. Модуль оснащений масивним алюмінієвим радіатором, гвинтовими клемниками для жорсткої фіксації дротів та інтегрованим лінійним стабілізатором напруги на 5 В. Наявність стабілізатора дозволяє живити логічну периферію та датчики робота напряму від силового акумулятора. Попри падіння напруги на внутрішніх

переходах, велика теплова маса радіатора нівелює цей недолік, роблячи модуль стійким до тривалих навантажень та заклинювань редуктора.

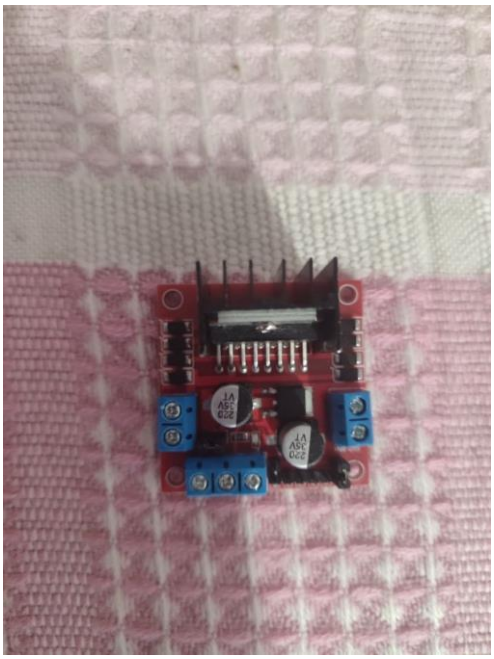

Рисунок 2.8 – L298N

Модуль L298N-mini [33] – компактна SMD-альтернатива для проектів із дефіцитом простору. Через відсутність радіатора він має мізерну вагу, але позбавлений теплового захисту та вбудованого регулятора на 5 В, що вимагає паяння та встановлення зовнішнього DC-DC конвертера. Головний ризик полягає в тому, що при блокуванні або різкому реверсі двигунів піковий пусковий струм може викликати миттєвий термічний пробою чіпа через неможливість швидкого розсіювання тепла.

Сучасний драйвер TB6612FNG [34] побудований на польових транзисторах, що забезпечує ККД понад 90 %, низький внутрішній опір та мінімальний нагрів. Проте чіп вкрай чутливий до статичної електрики, перенапруг та імпульсних завад від колекторних двигунів. Модуль не має вбудованого стабілізатора і потребує паяння, а найменша помилка у полярності або стрибок струму при перевантаженні шасі призводять до безповоротного виходу мікросхеми з ладу.

За результатами аналізу обрано повнорозмірний драйвер L298N. Його нижчий енергетичний ККД повністю компенсується конструктивною міцністю: гвинтові термінали стійкі до вібрацій шасі, вбудований перетворювач на 5 В

економить місце на платі, а висока стійкість до пікових струмів короткого замикання гарантує живучість системи в реальних експлуатаційних умовах.

2.3 Вибір датчиків відстані для СКТВ

Датчик для вимірювання відстані є важливим компонентом для уникнення перешкод під час руху. Саме цей датчик може виміряти відстань, і коли відстань буде менша ніж 1,5 метра до можливої перешкоди, дати команду апаратній платформі для зупинки і уникнення пошкоджень. Тому для роботи такої системи необхідно використати надійний датчик, що міг би виконувати завдання у відповідних умовах.

Ультразвукові датчики працюють за принципом вимірювання часу. Вони вимірюють час польоту звукової хвилі. Датчик може генерувати дуже багато імпульсів певної звукової частоти, що поширюється у повітрі. Такі імпульси повертаються назад до приймача, і саме за цей проміжок часу і розраховується відстань.

Лазерні датчики використовують схожий принцип роботи. Він дуже схожий тим, що лазерні датчики випромінюють світловий імпульс інфрачервоного спектра, а ультразвукові датчики звук з певною частотою. Швидкість світла дуже висока, тому датчик може швидко фіксувати зміщення фази чи точний час затримки сигналу.

HC-SR04 (рис. 2.8) [35] – ультразвуковий датчик відстані. Має класичний п'єзоелектричний випромінювач і приймач. Головною перевагою є конусна діаграма спрямування звукової хвилі. Кут такої конусної діаграми 15°. Це дозволяє отримати значення відстані до перешкоди, коли вона з'явилась у цьому спектрі, а не коли вона з'явилась рівно перед самим датчиком. Також такий датчик не залежить від кольору чи роботи на сонці, з відблисками.

Рисунок 2.9 – Вигляд HC-SR04

Недоліком є сліпа зона цього датчика. На відстані 2 см до датчика він не може визначити відстань. Це його сліпа зона. Також м'які матеріали добре поглинають звуковий сигнал, через що він спотворюється. Такі випадки призводять до хибних вимірів відстані. Має серйозну перешкоду, якщо звук буде сильним, щоб перебити сам механізм, що видає звукові хвилі.

VL53L0X v2 (рис. 2.9) [36] – лазерний датчик виконаний на базі інфрачервоного лазера. Має високу доволі точність, майже до 1 мм, та дуже високу швидкість вимірювання. Діапазон роботи до 2 м, що підходить під функціонал цієї СКТВ. Дуже невеликі габарити дозволяють розмістити його у більшості місць, навіть, у невеликих платформах. Серед недоліків є вузький кут огляду, промінь є точковим. Якщо перед роботом буде тонка вертикальна перешкода, яка зміщена на пару сантиметрів від осі променя, то датчик її може просто проігнорувати. Також його слабке місце це робота при зовнішньому інфрачервоному випромінюванні.

Рисунок 2.10 – Вигляд VL53L0X v2

Характеристики кращі у VL53L0X v2, але за використанням HC-SR04 буде зручнішим. Він не може так якісно визначити дистанцію, але і похибки на дистанції у декілька сантиметрів, навіть при великих шумах у будівлі, не зможуть змінити ситуацію, чи зробити так, щоб він врізався у об'єкт через цю причину. Також його очевидною перевагою є те, що можна використовувати його для пошуку перешкод у певному спектрі. Лазерний далекомір не зможе ефективно відстежити об'ємні перешкоди. До того ж, передавати дані через I2C може бути інколи доволі складно, якщо є декілька пристроїв з таким протоколом, також зростає і затримка. Контроль роботи HC-SR04 є доволі простим і складається з генерації звукових хвиль певної частоти та вимірюванням часу їх повернення.

2.4 Вибір мови програмування

Для розробки програмного забезпечення мобільної роботизованої платформи на базі мікрокомп’ютера Raspberry Pi Zero 2W необхідно обрати мову програмування, яка забезпечить швидку обробку даних від датчиків, ефективну роботу з відеопотоком та простоту реалізації вебінтерфейсу дистанційного керування.

Для системного аналізу було розглянуто три архітектурні підходи: компільовані мови наднизького рівня (C/C++), інтерпретовані високорівневі мови (Python).

C/C++ Історично є базовим інструментом для розробки вбудованих систем та систем реального часу. Серед переваг є максимально можлива швидкодія, відсутність додаткових інтерпретуючих шарів, повний контроль над виділенням та звільненням оперативної пам'яті. Програми компілюються безпосередньо в машинний код цільової архітектури ARMv8, що забезпечує прямий доступ до системних викликів ядра Linux та регістрів периферії через драйвери /dev/gpiomem або /dev/i2c-1.

Серед недоліків висока складність та тривалість процесу розробки. Відсутність автоматичного менеджменту пам'яті створює ризики появи критичних помилок: витоків пам'яті або помилок сегментації, що є неприпустимим для автономних систем. Створення легковагого вебсервера та реалізація асинхронної трансляції відеопотоку на C++ потребує підключення громіздких сторонніх бібліотек та значних часових витрат на налагодження. Крім того, компіляція важких проектів безпосередньо на мікрокомп'ютері є надто повільною через обмежену потужність CPU.

Високорівнева інтерпретована мова Python мультипарадигмальна мова програмування, що поєднує лаконічність синтаксису з потужною екосистемою готових модулів для інженерних та наукових задач. Серед переваг найвища швидкість розробки, гнучкість архітектури та простота інтеграції різнорідних компонентів. Наявність оптимізованих низькорівневих бібліотек, які офіційно підтримуються спільнотою Raspberry Pi Foundation gpiozero, smbus2, libcamera. Мова дозволяє в межах одного скрипта поєднати опитування датчиків, логіку керування приводами та вебсервер на базі легковагого фреймворку Flask або FastAPI, мінімізуючи складність міжпроцесної взаємодії.

Недоліком є Нижча швидкість виконання порівняно з компільованими мовами через динамічну типізацію та наявність глобального блокування

інтерпретатора , що обмежує чисту багатопотоковість на багатоядерних процесорах.

Обґрунтування вибору мови Python для розробки програмного забезпечення робота базується на швидкості створення мережевих сервісів та зручності взаємодії з периферією на базі Raspberry Pi Zero 2W. Ключовою властивістю самої мови є автоматичне керування ресурсами та вбудоване збирання сміття, що захищає автономну систему від критичних витоків пам'яті, тоді як базові апаратні вимоги інтерпретатора залишаються цілком прийнятними для доступних 512 МБ ОЗП. Нижча обчислювальна швидкість Python повністю нівелюється парадигмою мови-клею, оскільки ресурсомісткі модулі під капотом реалізовані на C/C++ і працюють на максимальній швидкості процесора. У підсумку, використання Python дозволяє швидко й надійно об'єднати роботу з шинами зв'язку та легковагий вебсервер в єдиний програмний стек без ускладнення архітектури.

2.5 Архітектура прототипа СКТВ

Головною апаратною особливістю схеми є повне фізичне розділення контурів живлення за допомогою двох незалежних імпульсних понижуючих перетворювачів (рис. 2.11 та 2.12). Це усуває перешкоди. Перший модуль Buck Converter стабілізує напругу для чутливої електроніки комп'ютера, тоді як другий виділено суто під потужні сервоприводи. Живлення драйвера двигунів L298N підключено напряму до акумуляторів 18650. Таке рішення економить енергію.

Для розвантаження процесора від рутинних завдань генерації ШІМ-сигналів у систему інтегровано спеціалізований 16-канальний драйвер PCA9685. Він керує сервоприводами. Зв'язок між головною платою та драйвером реалізовано через послідовну шину I2C. Вона використовує дві лінії. Ходовий двигун постійного струму підключений двома окремими дротами до силових виходів OUT1 та OUT2. Це забезпечує реверс. Первинний аналіз оточення та виявлення перешкод покладено на ультразвуковий датчик HC-SR04, підключений до GPIO. Усі компоненти мають спільну землю.

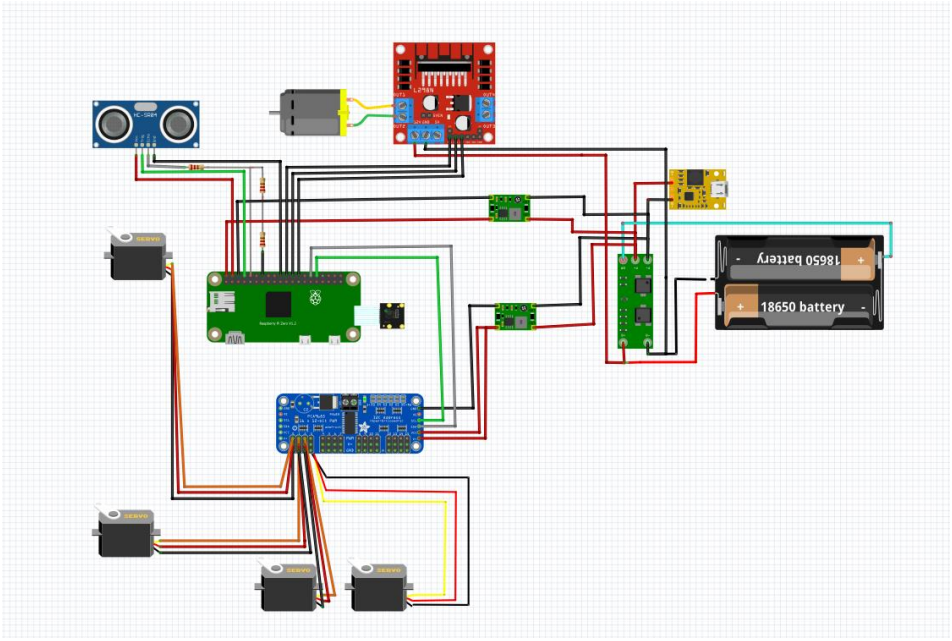

Рисунок 2.11 – Схема з'єднання компонентів для СКТВ

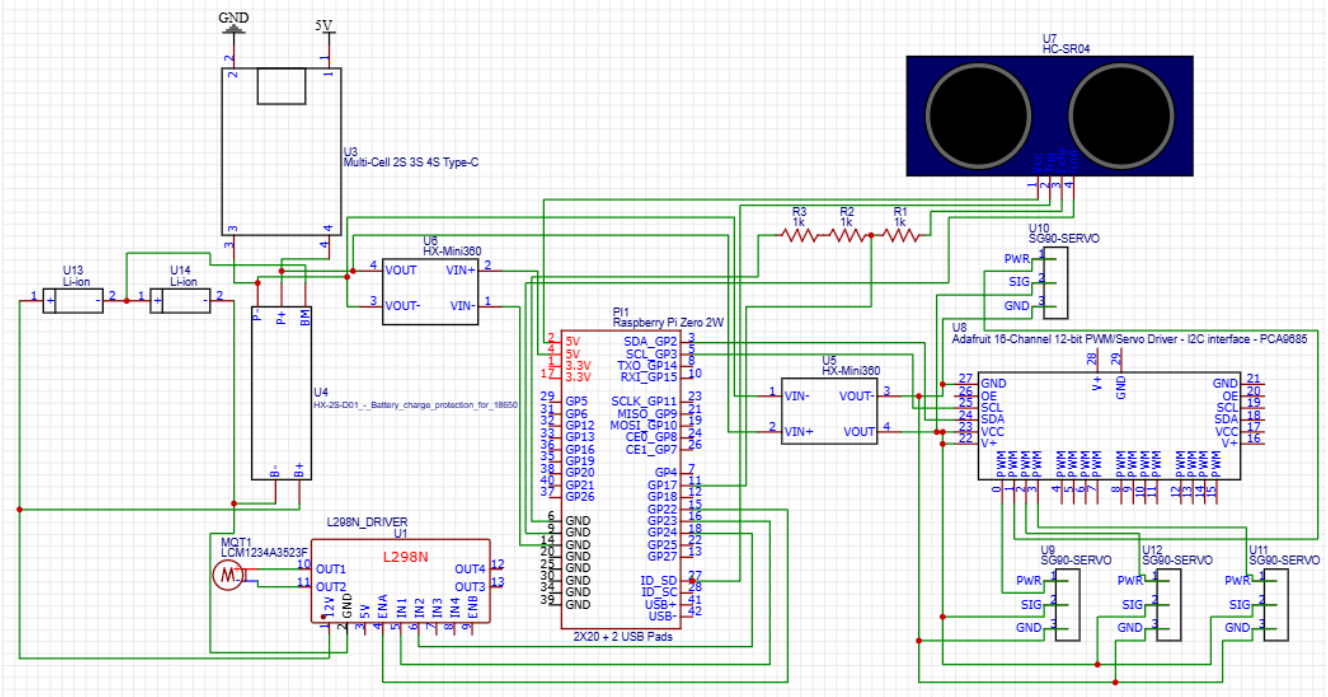

Рисунок 2.12 – Принципова схема з'єднання компонентів

Програмна частина робототехнічної системи побудована на базі асинхронного вебсервера з використанням мови Python та бібліотеки asyncio. Для забезпечення максимальної швидкодії на обмежених ресурсах мікрокомп'ютера, архітектура розділена на чотири паралельні потоки. Головний асинхронний потік миттєво обробляє вхідні пакети та динамічно коригує глобальний стан системи,

окремий потік HTTP відповідає за статику та візуалізацію, фоновий потік сонара з частотою 10 Гц проводить фізичні вимірювання, а апаратний потік камери забезпечує безперервне захоплення відеосигналу.

Передавання даних між клієнтським інтерфейсом та сервером реалізовано через два паралельні канали зв'язку. Перший – це класичний протокол HTTP, який використовується для первинного завантаження елементів керування та безперервної трансляції відеопотоку у форматі MJPEG за технологією multipart/x-mixed-replace. Другий – повнодуплексний протокол WebSockets, який у режимі реального часу забезпечує двосторонній обмін асинхронними JSON-пакетами (див. додаток А). Фронтенд безперервно передає команди руху та кути нахилу сервоприводів, а бекенд надсилає телеметрію та оновлені дані про відстань з ультразвукового датчика.

Висновки до 2 розділу

У результаті аналізу технічних вимог та проведення практичних тестувань для розробки роботизованої системи транспортування вантажів було успішно підібрано оптимальний апаратний комплекс. Його основу складає обчислювальний модуль Raspberry Pi Zero 2W, який забезпечує високу продуктивність в асинхронному режимі за мінімальних габаритів. Для силової частини обрано надійний драйвер мотора L298N у поєднанні з мотор-редуктором із металевими шестернями, що гарантує високий крутний момент та стійкість до механічних навантажень. Безпека руху та навігація реалізовані за допомогою ультразвукового датчика відстані HC-SR04, а візуальний контроль та трансляція потокового відео високої частоти забезпечуються спеціалізованою камерою для Raspberry Pi, підключеною через високошвидкісний CSI-інтерфейс.

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМИ КОНТРОЛЮ ТРАНСПОРТУВАННЯ ВАНТАЖІВ

3.1 Апаратна реалізація для СКТВ

Raspberry Pi Zero 2W має 40-пінний роз'єм GPIO, що видно на рисунку 3.1, який забезпечує взаємодію одноплатного комп'ютера з периферійними пристроями. Піни поділяються на кілька категорій залежно від їхнього призначення. Частина з них є звичайними цифровими входами або виходами. Інші підтримують спеціалізовані інтерфейси такі як I2C, SPI, UART або апаратний ШІМ.

Цифрові піни загального призначення працюють з логічними рівнями 3,3 В. Подача напруги вище цього значення може пошкодити плату. Саме тому у схемі підключення ультразвукового сенсора HC-SR04 використовується дільник напруги на лінії ECHO оскільки сенсор формує сигнал рівнем 5 В.

Апаратний ШІМ доступний на обмеженій кількості пінів. У Raspberry Pi Zero 2W повноцінний апаратний ШІМ підтримується на GPIO 18 та GPIO 19. Програмний ШІМ можна реалізувати на будь-якому піні але він менш стабільний через переривання операційної системи. Саме тому GPIO 18 обрано для керування підсвічуванням де важлива плавність регулювання яскравості.

Інтерфейс I²C реалізований на пінах GPIO 2 та GPIO 3. Через нього підключається ШІМ-контролер PCA9685 що керує чотирма сервоприводами. Перевага I²C полягає у тому що для підключення будь-якої кількості пристроїв достатньо лише двох ліній. Кожен пристрій має унікальну адресу за якою до нього звертається контролер.

Піни керування мотором та ультразвуковим сенсором налаштовані як звичайні цифрові входи або виходи. Швидкість двигуна регулюється через ШІМ-сигнал на піні GPIO 22. Напрямок обертання задається комбінацією рівнів на GPIO 23 та GPIO 24.

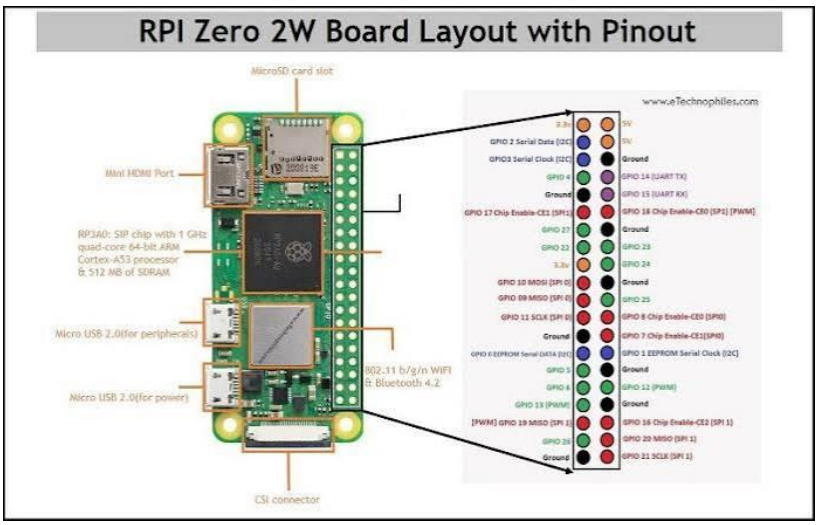

Рисунок 3.1 – Піни для Raspberry Pi Zero 2W [37]

Raspberry Pi Zero 2W має виділений інтерфейс CSI для підключення камерного модуля. На відміну від USB-камер цей інтерфейс забезпечує мінімальну затримку та не навантажує процесор зайвою роботою з протоколом.

Камерний модуль підключається через плаский шлейф до відповідного роз'єму на платі. Після підключення камера керується через бібліотеку picamera2 яка є офіційним інструментом для роботи з камерою на сучасних версіях Raspberry Pi OS. Бібліотека взаємодіє з підсистемою libcamera на рівні операційної системи. Це дозволяє налаштовувати параметри зйомки програмно без потреби у низькорівневому програмуванні.

Для потокової передачі відео використовується кодувальник MJPEGEncoder. Він стискає кожен кадр окремо у формат JPEG і передає його у буфер StreamingOutput. Такий підхід має перевагу. Кожен кадр є самодостатнім і не залежить від попередніх. Це означає що при втраті одного кадру відтворення продовжується без артефактів.

Частота кадрів та роздільна здатність задаються при створенні конфігурації через *create_video_configuration*. Raspberry Pi Zero 2W здатна обробляти відео у роздільній здатності 1280×720 при 15 кадрах на секунду або 320×240 при 60 кадрах на секунду. Вибір між цими режимами залежить від умов використання. При активному керуванні у складних умовах перевага надається низькій затримці а не якості зображення.

Для керування електродвигуном постійного струму використовується драйвер L298N. Він виступає посередником між логічними сигналами Raspberry Pi та силовими колами двигуна, а з’єднання можна побачити в таблиці 3.1.

L298N побудований на основі схеми H-моста. Це означає що він здатен подавати напругу на двигун у двох напрямках. Напрямок обертання визначається комбінацією сигналів на пінах IN1 та IN2. Якщо IN1 має високий рівень а IN2 низький двигун обертається в один бік, як на табл. 3.2. При зворотній комбінації двигун змінює напрямок. Якщо обидва піни мають однаковий рівень двигун гальмує.

Швидкість обертання регулюється через пін ENA за допомогою ШІМ-сигналу. Шпаруватість імпульсів визначає середню напругу що подається на двигун. При рівні ШІМ 100 % двигун обертається з максимальною швидкістю. При 50 % швидкість зменшується приблизно вдвічі. Це дозволяє плавно регулювати швидкість руху транспортного засобу без механічних регуляторів.

Драйвер має окреме живлення для силової частини. Логічна частина живиться від 5 В Raspberry Pi. Силова частина підключається до зовнішнього джерела живлення відповідно до характеристик двигуна. Такий поділ захищає одноплатний комп'ютер від стрибків напруги які виникають при запуску та гальмуванні двигуна.

Таблиця 3.1 – Під’єднання драйверу двигунів L298N до Raspberry Pi Zero 2W

Raspberry Pi Zero 2W	L298N
GPIO 22	ENA
GPIO 23	IN 1
GPIO 24	IN 2

Таблиця 3.2 – Розпіновка роботи драйвера двигунів L298N

Подання напруги	ENA	IN 1	IN 2
Рух вперед	0–1024	1	0
Рух назад	0–1024	0	1

Зупинка руху	0–1024	0	0

Для керування ультразвуковим датчиком необхідно використовувати джерело живлення 5 В. Але вихідний сигнал з самого датчика має напругу 5В, інтерфейс Raspberry Pi Zero 2W може працювати з напругою 5 В. Напруга інтерфейсу 3,3 В, тому треба використовувати дільник напруги, як на рис. 3.2. Через те, що область з вищим опором, необхідна частина напруги буде використовуватися саме інтерфейсом Raspberry Pi Zero 2W. Це працює за співвідношенням опорів.

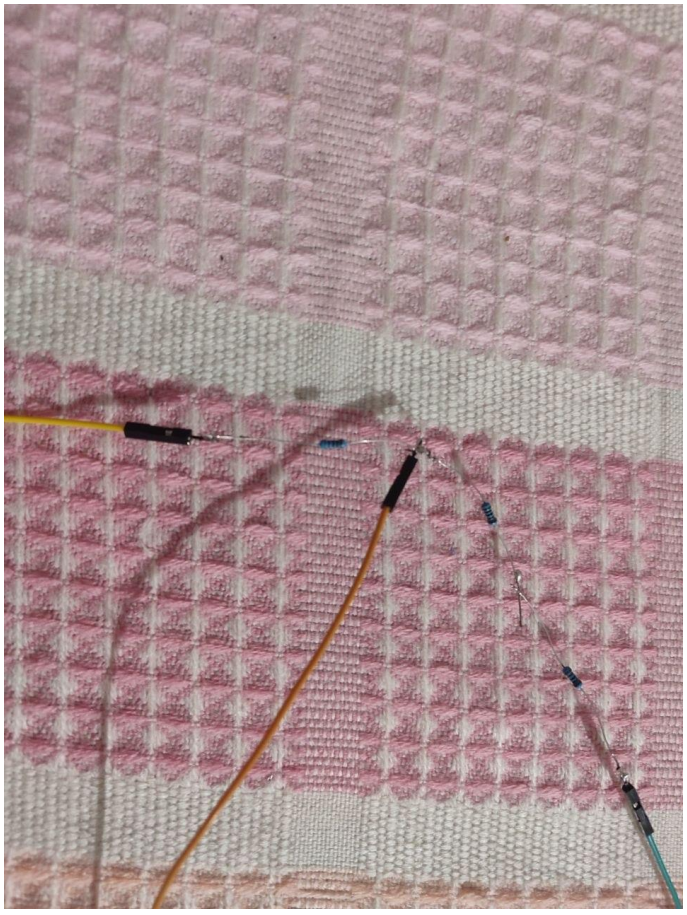

Рисунок 3.2 – Дільник напруги для Raspberry Pi Zero 2W

Для роботи і отримання необхідної напруги необхідно використати резистори у співвідношені до необхідної напруги. 5 В можна розділити на необхідну напругу і її частину, що залишиться, і почне далі йти до землі. Необіхдна напруга 3,3 В, значить залишиться 1,7 В. Для цього можна використати резистори опором 1 кОм і 2 кОм. Дільник напруги можна зробити з використанням 3

резисторів 1 кОм. Перший резистор з номіналом 1 кОм включається між піном ECHO сенсора та піном GPIO плати. З'єднати послідовно два резистора з опором 1 кОм і можна отримати опір 2 кОм. Ці два резистора треба з'єднати з GPIO плати та землею. Таким чином з'єднання між Raspberry Pi Zero 2w та датчиком відстані HC-SR04 стає безпечним.

Ультразвуковий сенсор має 4 контакти. VCC підключається до піна з напругою 5В на роз'ємі Raspberry Pi Zero 2W плати одноплатного комп'ютера. Можливо підключити 3,3 В, але зменшується відстань роботи датчика. Пін GND підключається до заземленого піна на платі, до GND. Пін TRIG підключається до самого GPIO 27. ECHO підключається саме через дільник напруги, а не напряму, до 3,3 В до піна GPIO 17, як вказано в таблиці 3.3.

Принцип роботи сенсора базується на вимірюванні часу повернення звуку до сенсора. Raspberry Pi Zero 2W генерує імпульс тривалістю 10 мікросекунд на пін TRIG. Сенсор у відповідь випромінює серію з восьми ультразвуковими імпульсами частотою 40 кГц. Після цього пін ECHO переходить у високий рівень і залишається в ньому до повернення відлуння.

Таблиця 3.3 – Розпіновка роботи для ультразвукового датчика відстані

Raspberry Pi Zero 2W	HC-SR04
5V	VCC
GND	GND
GPIO 27	TRIG
GPIO 17	ECHO

Програма фіксує момент переходу ECHO у високий рівень і повернення у низький. Різниця між цими моментами часу є тривалістю розповсюдження звуку у просторі і повернення від перешкоди до самого датчика. Ця довжина множиться на 343 м/с і ділиться навпіл. Результатом цих дій є відстань до перешкоди у метрах.

PCA9685 це 16 канальний контролер, розширювач ШІМ-виходів, який використовується в робототехніці для керування сервомоторами, чи світодіодами.

Він зробить генерацію ШІМ-сигналів за Raspberry Pi Zero 2W і знімає завдання з одноплатного комп'ютера.

Сервомотори керуються за допомогою ШІМ-сигналів, де від часу імпульсу залежить сам кут повороту. Генерація сигналів для декількох сервомоторів є складною задачею для Raspberry Pi Zero 2W, що знімає частину її ресурсів. Може не статися виконання важливої обробки сигналу через мікрозатримки для роботи з сервомоторами. PCA9685 має свій вбудований тактовий генератор, тож коли Raspberry Pi Zero 2W дасть сигнал на поворот сервомотора, контролер це зможе виконати. До того ж, цей розширювач сигналів ШІМ може запам'ятовувати позиції і самостійно видає стабільний сигнал на сервомотори. Якщо система буде мати збій, то цей контролер буде тримати заданий кут.

Має підключення за допомогою I2C за допомогою двох провідників. Для силового з'єднання використовується вже 3 провідника. Два з них мають напругу 5 В, хоча для живлення логіки можна використовувати , хоча для живлення логіки можна використовувати 3,3 V, і один провідник із заземленням. До одного PCA9685 можна підключити до 16 сервомоторів. Якщо є необхідність, то можна підключати їх послідовно і зробити схему з декількох таких PCA9685. Таким чином, можна збільшити кількість підключених сервомоторів до 992, маючи обмеження по кількості адрес по шині I2C. Повна схема наведена у таблиці 3.4.

Таблиця 3.4 – Розпіновка з'єднань Raspberry Pi Zero 2W і PCA9685

Raspberry Pi Zero 2W	PCA9685
GPIO 2	SDA
GPIO 3	SCL
5V	VCC
5V	V+
GND	GND

Для системи живлення використовувались акумулятори Li-Ion з ємністю 3000 mAh та номінальною напругою та номінальною напругою 3,7 В. Вони є джерелом живлення для СКТВ. Кількість акумуляторів 2 штуки. Вони з'єднані

послідовно, що дає змогу збільшити напругу до 7,4 В. Ємність залишається 3000 mAh.

Для захисту системи живлення використовується плата BMS. Завдяки цієї плати можна забезпечити захист від короткого замикання, використання високого струму системою, чи короткого замикання. Така плата має серйозний вплив на довговічність акумуляторів. Ця плата може переваги давати живлення всій системі, коли, коли Li-Ion батареї мають достатній рівень заряду та захищає від використання надмірного рівня струму. Коли елементи Li-Ion розряджаються, то BMS плата відключає живлення, і рівень зниження заряду суттєво зменшується, таким чином збільшуючи довговічність системи.

Коли система заряджається, то така плата BMS захищає систему від перезарядки і може допомогти їй довше працювати справно. Мінусом, системи BMS, що використовується для створення самої СКТВ, є відсутність балансування напруги, під час зарядки і під час роботи СКТВ коли вона використовується.

Для забезпечення живлення 5 В від Li-Ion акумуляторів використовуються два перетворювачі. Вони з'єднані паралельно для системи живлення. Одне перетворювач використовується для живлення плати PCA9685 і 4 сервомоторів. Інший перетворювач потрібен для того, щоб напруг, що подавалась на Raspberry Pi Zero 2W була також 5 В. Використання одразу двох перетворювачів напруги потрібно для того, щоб окремо забезпечувати систему для керування сервомоторів і для Raspberry Pi Zero 2W. Сервомотори можуть використовувати одразу до 3 А, а Raspberry Pi Zero 2,5 А. Тому перетворювачі розрахований на таку силу струму і можуть витримувати до 4 А на 3 секунди.

Для зарядки акумуляторів використовується спеціальний модуль зарядки для Li-Ion елементів. Він немає системи для балансування зарядки – BMS, тому необхідна ще одна. Даний модуль може заряджати акумулятори з силою струму до 2 А.

3.2 Програмна реалізація для СКТВ

Блок імпортів підключає всі необхідні модулі для роботи системи. Стандартні бібліотеки Python – asyncio, threading, json, time, io, os, socketserver та http.server, які продемонстровані на рис. 3.4. Ці бібліотеки можуть забезпечують асинхронне виконання задач, паралельні потоки, серіалізацію даних, роботу з файловою системою та HTT сервером і доступні без додаткового встановлення.

```
import asyncio
import json
import threading
import time
import io
import os
import socketserver
import websockets
from http import server as http_server
from http.server import BaseHTTPRequestHandler

try:
    import RPi.GPIO as GPIO
    import board
    import busio
    from adafruit_pca9685 import PCA9685
    from adafruit_motor import servo as adafruit_servo
    HARDWARE = True
except ImportError:
    print("[WARN] GPIO/PCA9685 не знайдено")
    HARDWARE = False

try:
    from picamera2 import Picamera2
    from picamera2.encoders import MJPEGEncoder
    from picamera2.outputs import FileOutput
    CAMERA_AVAILABLE = True
except ImportError:
    print("picamera2 не знайдено")
    CAMERA_AVAILABLE = False
```

Рисунок 3.4 – Підключення бібліотек

Бібліотека websockets підключається окремо і реалізує WebSocket протокол. Імпорт апаратних залежностей – Rpi.GPIO, board, busio, adafruit_pca9685 та adafruit_motor – загорнуто у блок try/except, оскільки ці бібліотеки доступні виключно на Raspberry Pi. Аналогічним чином обробляється імпорт picamera2 з прапором CAMERA_AVAILABLE, що дозволяє запускати та тестувати логіку сервера на будь-якому комп'ютері без фізичного підключення камери чи периферії.

Перша частина фрагменту визначає константи номерів GPIO-пінів у BCM-нумерації, які використовуються для підключення периферійних пристроїв до Raspberry Pi Zero 2W, тобто на рис. 3.5. *PIN_ENA* (GPIO 22) є виходом ШІМ-сигналу для керування швидкістю обертання двигуна через драйвер L298N – змінюючи шпаруватість імпульсів на цьому піні, система плавно регулює оберти від нуля до максимуму. *PIN_IN1* (GPIO 23) та *PIN_IN2* (GPIO 24) визначають

напрямок обертання: подача високого рівня на один із них при низькому рівні на іншому задає напрямок вперед або назад. *PIN_TRIG* (GPIO 27) та *PIN_ECHO* (GPIO 17) обслуговують ультразвуковий далекомір HC-SR04 – перший формує короткий імпульс-запит тривалістю 10 мкс, другий приймає відлуння і за тривалістю відповідного імпульсу обчислюється відстань до перешкоди. *PIN_LED* (GPIO 18) виділений для керування підсвічуванням і обраний невипадково: GPIO 18 є одним із двох апаратних ШІМ-виходів Raspberry Pi, що забезпечує точне та стабільне керування яскравістю без програмної емуляції.

```
PIN_ENA  = 22
PIN_IN1  = 23
PIN_IN2  = 24
PIN_TRIG = 27
PIN_ECHO = 17
PIN_LED  = 18


    "high":   {"size": (1280, 720), "fps": 15, "quality": 90},
    "medium": {"size": (640,  480), "fps": 30, "quality": 75},
    "low":    {"size": (320,  240), "fps": 60, "quality": 60},
}
```

Рисунок 3.5 – Підключення пінів та якості для камери

Друга частина фрагменту описує словник пресетів камери, кожен із яких визначає три параметри: роздільну здатність size, частоту кадрів fps та якість JPEG-стиснення quality. Пресет high налаштований на роздільну здатність 1280 × 720 пікселів при 15 FPS з якістю стиснення 90 % – він забезпечує найдеталізованіше зображення, але потребує найбільшої пропускної здатності мережі та обчислювальних ресурсів процесора. Пресет medium із роздільною здатністю 640 × 480 при 30 кадрах на секунду і якістю 75 % є збалансованим режимом для повсякденного використання, коли важлива плавність відео за помірного навантаження. Пресет low із роздільною здатністю 320 × 240 при 60 кадрах на секунду та якістю 60 % орієнтований на максимальну плавність і мінімальну затримку. Саме цей режим є найбільш придатним для активного керування транспортним засобом у реальному часі, коли швидка реакція оператора важливіша за деталізацію зображення. Це є частиною OSD.

Зниження якості JPEG-стиснення пропорційно зменшенню роздільної здатності є свідомим компромісом. Менший розмір кадру вже сам по собі скорочує обсяг даних, тому додаткове зниження якості дозволяє досягти цільового показника 60 FPS навіть на обмеженому процесорі Raspberry Pi Zero 2W.

Функція camera_init відповідає за первинний запуск камери при старті системи і приймає назву пресету як аргумент, за замовчуванням використовуючи medium. Спочатку перевіряється доступність камери через прапор *CAMERA_AVAILABLE* – якщо камера не підключена, функція завершується негайно (рис. 3.6).

```
def camera_init(preset_name: str = "medium") -> None:
    global _camera, _current_preset
    if not CAMERA_AVAILABLE:
        return
    preset = PRESETS[preset_name]
    _camera = Picamera2()
    config = _camera.create_video_configuration(
        main={"size": preset["size"]},
        controls={"FrameRate": float(preset["fps"])},
    )
    _camera.configure(config)
    encoder = MJPEGEncoder(quality=preset["quality"])
    _camera.start_recording(encoder, FileOutput(stream_output))
    _current_preset = preset_name
    print(f"[CAM] Запущено: {preset['size']} @ {preset['fps']} fps")


def camera_change_preset(preset_name: str) -> None:
    global _current_preset
    if not CAMERA_AVAILABLE or _camera is None:
        return
    if preset_name == _current_preset or preset_name not in PRESETS:
        return

    preset = PRESETS[preset_name]
    print(f"[CAM] Зміна пресету → {preset_name}: {preset['size']} @ {preset['fps']} fps")

    with _camera_lock:
        try:
            _camera.stop_recording()
            config = _camera.create_video_configuration(
                main={"size": preset["size"]},
                controls={"FrameRate": float(preset["fps"])},
            )
            _camera.configure(config)
            encoder = MJPEGEncoder(quality=preset["quality"])
            _camera.start_recording(encoder, FileOutput(stream_output))
            _current_preset = preset_name
```

Рисунок 3.6 – Функції для роботи з камерою

Далі зі словника PRESETS витягується відповідний набір параметрів, створюється об'єкт Picamera2 і формується конфігурація відеозапису з заданою роздільною здатністю та частотою кадрів. Після цього ініціалізується кодувальник MJPEGEncoder із відповідним рівнем якості JPEG, і запис стартує з виведенням потоку у глобальний буфер stream_output. Буфер є спільним між камерою та HTTP-сервером – саме через нього кадри передаються клієнтам у браузер.

Функція *camera_change_preset* вирішує складнішу задачу – перемикання роздільної здатності під час роботи без зупинки сервера. На початку виконуються

три захисні перевірки: чи доступна камера, чи не збігається новий пресет із поточним і чи існує він взагалі у словнику. Якщо всі перевірки пройдено, виконання входить у блок *with _camera_lock* – це м'ютекс, який не дає двом потокам одночасно змінювати стан камери, що важливо для OSD. Всередині блоку поточний запис зупиняється, камера перелаштовується з новими параметрами і запускається знову з оновленим кодувальником. Весь процес загорнуто у try/except, щоб можлива апаратна помилка не покладала весь сервер.

Константа *SCRIPT_DIR* зберігає абсолютний шлях до директорії, де знаходиться сам скрипт, що дозволяє коректно знаходити файли незалежно від того, з якої директорії запускається сервер. Це усуває проблему відносних шляхів.

Клас *CarHTTPHandler* успадковує *BaseHTTPRequestHandler* і обробляє всі вхідні GET-запити, розподіляючи їх за трьома маршрутами: запити до /stream перенаправляються у метод *_handle_mjpeg* для передачі відеопотоку, запити до / або /index.html повертають головну сторінку інтерфейсу, а всі інші шляхи трактуються як запити до статичних файлів (рис. 3.7).

```
class CarHTTPHandler(BaseHTTPRequestHandler):

    def do_GET(self):
        if self.path.startswith("/stream"):
            self._handle_mjpeg()
        elif self.path in ("/", "/index.html"):
            self._serve_file("index.html", "text/html; charset=utf-8")
        else:
            fname = self.path.lstrip("/")
            fpath = os.path.join(SCRIPT_DIR, fname)
            if os.path.isfile(fpath):
                ctype = "application/octet-stream"
                if fname.endswith(".js"):   ctype = "text/javascript"
                if fname.endswith(".css"):  ctype = "text/css"
                if fname.endswith(".ico"):  ctype = "image/x-icon"
                self._serve_file(fname, ctype)
            else:
                self.send_error(404)

    def _handle_mjpeg(self):
        self.send_response(200)
        self.send_header("Age", "0")
        self.send_header("Cache-Control", "no-cache, private")
        self.send_header("Pragma", "no-cache")
        self.send_header("Content-Type", "multipart/x-mixed-replace; boundary=FRAME")
        self.send_header("Access-Control-Allow-Origin", "*")
        self.end_headers()
        try:
            while True:
                with stream_output.condition:
                    got_frame = stream_output.condition.wait(timeout=1.0)
                    if not got_frame or stream_output.frame is None:
                        continue
                    frame = stream_output.frame

                self.wfile.write(b"--FRAME\r\n")
                self.wfile.write(b"Content-Type: image/jpeg\r\n")
```

Рисунок 3.7 – Частина коду для роботи з статичними файлами та запитами

Для статичних файлів тип вмісту визначається за розширенням – .js, .css або .ico – і встановлюється відповідний заголовок Content-Type. Якщо файл за вказаним шляхом не знайдено, клієнту повертається стандартна помилка 404.

Метод *_handle_mjpeg* формує HTTP відповідь спеціального типу multipart/x-mixed-replace, який є основою MJPEG стрімінгу – браузер отримує одне відкрите з'єднання і постійно замінює зображення новими кадрами без перезавантаження. Заголовки Cache-Control: no-cache та Pragma: no-cache забороняють будь-яке кешування, а Access-Control-Allow-Origin: * дозволяє підключатися з будь-якого домену. Після відправки заголовків запускається нескінченний цикл, у якому метод блокується на умовній змінній stream_output.condition з таймаутом одну секунду – це означає, що потік засинає і прокидається лише тоді, коли камера записала новий кадр у буфер. Таймаут захищає від вічного блокування якщо камера раптово зупинилась.

Чотири глобальні змінні *_motor_pwm*, *_led_pwm*, *_pca* та *_servos* оголошуються на рівні модуля зі значеннями None і порожнім списком (рис. 3.8). Це дозволяє іншим функціям звертатися до них до виклику ініціалізації. Такий підхід є стандартним для апаратних драйверів на Python.

Функція gpio_init виконує послідовне налаштування всіх периферійних пристроїв. Спочатку встановлюється BCM-нумерація пінів і вимикаються попередження GPIO. Потім три піни мотора налаштовуються як виходи, створюється об'єкт ШІМ на піні *PIN_ENA* з частотою 1000 Гц і одразу стартує з нульовою шпаруватістю. Пін *PIN_TRIG* ультразвукового сенсора налаштовується як вихід, PIN_ECHO як вхід, після чого TRIG примусово переводиться у низький рівень. Пауза 100 мс після цього потрібна щоб сенсор стабілізувався перед першим вимірюванням. Підсвічування ініціалізується на піні PIN_LED з частотою ШІМ 500 Гц. Далі відкривається I²C-шина через busio.I2C і до неї підключається контролер PCA9685 з частотою 50 Гц. У циклі по чотирьох каналах створюються об'єкти серводвигунів з діапазоном імпульсів від 500 до 2500 мікросекунд. Цей діапазон охоплює повний кут повороту SG90 від 0 до 180 градусів.

```
_motor_pwm = None
_led_pwm   = None
_pca       = None
_servos    = []


def gpio_init() -> None:
    global _motor_pwm, _led_pwm, _pca, _servos
    if not HARDWARE:
        return

    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)

    for pin in (PIN_ENA, PIN_IN1, PIN_IN2):
        GPIO.setup(pin, GPIO.OUT)
    _motor_pwm = GPIO.PWM(PIN_ENA, 1000)
    _motor_pwm.start(0)

    GPIO.setup(PIN_TRIG, GPIO.OUT)
    GPIO.setup(PIN_ECHO, GPIO.IN)
    GPIO.output(PIN_TRIG, False)
    time.sleep(0.1)
    GPIO.setup(PIN_LED, GPIO.OUT)
    _led_pwm = GPIO.PWM(PIN_LED, 500)
    _led_pwm.start(0)

    i2c = busio.I2C(board.SCL, board.SDA)
    _pca = PCA9685(i2c)
    _pca.frequency = 50
    for ch in range(4):
        s = adafruit_servo.Servo(
            _pca.channels[ch],
            min_pulse=500,
            max_pulse=2500,
        )
        _servos.append(s)

    print("[GPIO] Ініціалізовано")
```

Рисунок 3.8 – Ініціалізація пінів для роботи з драйвером двигунів і PCA9685

Функція *gpio_cleanup* виконується при завершенні програми. Спочатку мотор зупиняється примусово. Потім послідовно зупиняються обидва ШІМ-об'єкти. Контролер PCA9685 деініціалізується через deinit. Наприкінці викликається GPIO.cleanup, який переводить усі піни у безпечний стан. Це важливо щоб уникнути пошкодження схеми при наступному запуску.

```
async def _ws_handler(websocket) -> None:
    addr = websocket.remote_address
    print(f"[WS] Підключено: {addr}")
    async with _ws_clients_lock:
        _ws_clients.add(websocket)

    try:
        async for raw in websocket:
            try:
                data: dict = json.loads(raw)
            except json.JSONDecodeError as exc:
                print(f"[WS] JSON помилка: {exc}")
                continue

            if "changeResolution" in data:
                preset = data["changeResolution"]
                threading.Thread(
                    target=camera_change_preset,
                    args=(preset,),
                    daemon=True,
                ).start()
                continue

            _state.update({
                "speed":           int(data.get("speed",           0)),
                "light":           int(data.get("light",           0)),
                "command":         str(data.get("command",         "stop")),
                "lightTurnToggle": int(data.get("lightTurnToggle", 0)),
                "wheel":           int(data.get("wheel",           90)),
                "yaw":             int(data.get("yaw",             90)),
                "pitch":           int(data.get("pitch",           90)),
                "roll":            int(data.get("roll",            90)),
                "toggleSonar":     int(data.get("toggleSonar",     0)),
            })
            _apply_state()

    except Exception as exc:
        print(f"[WS] {addr} від'єднався: {exc}")
    finally:
        async with _ws_clients_lock:
            ws_clients.discard(websocket)
```

Рисунок 3.9 – Код для роботи з вебклієнтом

Функція *_ws_handler* є асинхронним обробником кожного WebSocket-підключення (рис. 3.9). При підключенні нового клієнта його адреса виводиться у консоль. Об'єкт з'єднання додається до множини *_ws_clients* під захистом асинхронного м'ютекса. Це потрібно щоб інші частини системи могли розсилати повідомлення всім підключеним клієнтам одночасно.

Далі запускається асинхронний цикл async for raw in websocket який очікує нові повідомлення. Кожне повідомлення розбирається з JSON-формату. Якщо рядок не є валідним JSON, помилка виводиться у консоль і цикл переходить до наступного повідомлення без зупинки сервера.

Після успішного розбору перевіряється наявність ключа changeResolution. Якщо він присутній, запускається окремий daemon-потік для зміни пресету камери. Потік є daemon щоб не блокувати завершення програми. Після запуску потоку обробник одразу переходить до наступного повідомлення через continue.

Якщо ключ changeResolution відсутній, повідомлення трактується як команда керування транспортним засобом. Усі дев'ять полів зчитуються з словника з безпечними значеннями за замовчуванням через data.get. Це захищає від ситуації коли клієнт надіслав неповний пакет. Оновлений стан одразу передається у функцію _apply_state яка транслює його у команди до GPIO.

Блок finally виконується завжди при закритті з'єднання. З'єднання видаляється з множини клієнтів через discard. Метод discard на відміну від remove не викидає виняток якщо елемент вже відсутній.

Цей фрагмент описує нижню частину вебінтерфейсу оператора (рис. 3.10).

Чотири слайдери керують кутами сервоприводів у діапазоні від 0 до 180 градусів. Початкове значення 90 відповідає центральному положенню кожного серво. Слайдери відповідають за напрямок коліс, горизонтальний і вертикальний поворот камери та кут нахилу ультразвукового сенсора.

Нижче розміщено текстовий рядок для відображення поточних даних далекоміра та дві системні кнопки. Перша вмикає або вимикає опитування ультразвукового сенсора. Друга перемикає підсвічування.

Сітка з дев'яти кнопок реалізує повне керування рухом. Вона організована за принципом цифрової клавіатури. Центральна кнопка STOP виділена червоним кольором. Кутові кнопки задають діагональний рух. Внизу відображається статус WebSocket-з'єднання.

```
    <input type="range" min="0" max="180" value="90" id="wheel">
  </div>

  <div class="row">
    <p id="yawServo">Yaw is 90</p>
    <input type="range" min="0" max="180" value="90" id="yaw">
  </div>

  <div class="row">
    <p id="pitchServo">Pitch is 90</p>
    <input type="range" min="0" max="180" value="90" id="pitch">
  </div>

  <div class="row">
    <p id="rollUltrasonic">Roll ultrasonic is 90</p>
    <input type="range" min="0" max="180" value="90" id="roll">
  </div>

  <hr style="border:0; border-top:1px solid #eee; margin:15px 0;">

  <p id="ultrasonic" style="margin:0 0 8px 0; font-size:14px; text-align:center;">Ultrasonic is off</p>
  <button class="sys-btn" id="sonarBtn"    onclick="sendToggleSonar()">Turn on sonar</button>
  <button class="sys-btn" id="lightToggle" onclick="sendLightToggle()">Turn Light Off</button>

  <div class="grid">
    <button id="leftFrw"  onclick="sendLeftForward()">L.Frw</button>
    <button id="frw"      onclick="sendForward()">Forward</button>
    <button id="rightFrw" onclick="sendRigthForward()">R.Frw</button>

    <button id="lft"      onclick="sendLeft()">Left</button>
    <button id="stop"     onclick="sendStop()">STOP</button>
    <button id="right"    onclick="sendRigth()">Right</button>

    <button id="leftBcw"  onclick="sendLeftBackward()">L.Bcw</button>
    <button id="bcw"      onclick="sendBackward()">Backward</button>
    <button id="rightBcw" onclick="sendRigthBackward()">R.Bcw</button>
  </div>

  <div class="status"><span>Connecting...</span></div>
</div>
```

Рисунок 3.10 – Код HTML-сторінки для керування машинкою

Обробник keyup скидає змінну lastCommand у null коли клавіша відпускається. Це дозволяє повторно надіслати ту саму команду при наступному натисканні (рис. 3.11).

Дев'ять функцій керування рухом влаштовані однаково. Кожна записує рядкову команду у третій елемент масиву message і одразу викликає sendMsg. Масив message є єдиним джерелом стану який відправляється на сервер.

Функція *sendLightToggle* перемикає підсвічування за принципом тумблера. При кожному виклику стан інвертується між 0 і 1. Одночасно оновлюється текст кнопки щоб відображати наступну дію. Після зміни стану повідомлення одразу надсилається на сервер.

```
document.addEventListener("keyup", (e) => {
  if (KEY_CMD[e.key]) lastCommand = null;
});

function sendLeftForward()   { message[2] = "leftForward";   sendMsg(); }
function sendForward()       { message[2] = "forward";       sendMsg(); }
function sendRigthForward()  { message[2] = "rightForward";  sendMsg(); }
function sendLeft()          { message[2] = "left";          sendMsg(); }
function sendStop()          { message[2] = "stop";          sendMsg(); }
function sendRigth()         { message[2] = "right";         sendMsg(); }
function sendLeftBackward()  { message[2] = "leftBackward";  sendMsg(); }
function sendBackward()      { message[2] = "backward";      sendMsg(); }
function sendRigthBackward() { message[2] = "rightBackward"; sendMsg(); }

function sendLightToggle() {
  if (lightStateNow === 0) {
    lightStateNow = 1;
    message[3]    = 1;
    lightToggleBtn.innerHTML = "Turn Light On";
  } else {
    lightStateNow = 0;
    message[3]    = 0;
    lightToggleBtn.innerHTML = "Turn Light Off";
  }
  sendMsg();
}

function sendToggleSonar() {
  if (lastSonarCounter === 0) {
    message[8]        = 1;
    lastSonarCounter  = 1;
    sonarBtnEl.innerHTML     = "Sonar turn Off";
    ultrasonicEl.innerHTML   = "Measuring";
  } else {
    message[8]        = 0;
    lastSonarCounter  = 0;
    sonarBtnEl.innerHTML     = "Turn on sonar";
    ultrasonicEl.innerHTML   = "Ultrasonic is off";
  }
  sendMsg();
```

Рисунок 3.11 – Віпдравка даних у WebSocket

Функція *sendToggleSonar* працює за тим самим принципом тумблера. При вмиканні у масив записується 1 і текст кнопки змінюється на вимкнення. При вимиканні записується 0 і відображається початковий текст. Рядок стану під кнопкою також оновлюється щоб інформувати оператора про поточний режим роботи сенсора.

3.3 Тестування створеного прототипа для СКТВ

Після завершення розробки програмного і апаратного забезпечення треба виконати перевірку роботи самої СКТВ (рис. 3.12). Основною метою тестування було проходження тестування дистанційного керування роботизованою платформою, а саме передачу пакетів. Протестовано роботу вебінтерфейсу на функціональність клавіш та можливість відеопотоку даних від камери до інтерфейсу користувача.

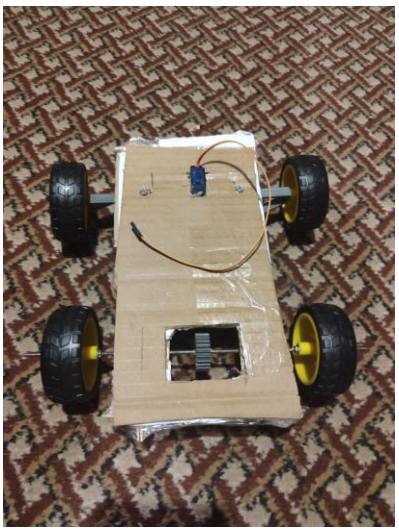

Рисунок 3.12 – Платформа для СКТВ

Raspberry Pi Zero 2W виконує роль центрального обчислювального вузла, забезпечуючи роботу вебсервера, обробку команд та передачу відеоданих. Користувач під'єднується до платформи за допомогою бездротової локальної мережі і керує платформою за допомогою веб браузера.

```
C:\Users\voroj>ping 192.168.100.109

Обмен пакетами с 192.168.100.109 по с 32 байтами данных:
Ответ от 192.168.100.109: число байт=32 время=43мс TTL=64
Ответ от 192.168.100.109: число байт=32 время=3мс TTL=64
Ответ от 192.168.100.109: число байт=32 время=3мс TTL=64
Ответ от 192.168.100.109: число байт=32 время=3мс TTL=64

Статистика Ping для 192.168.100.109:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 3мсек, Максимальное = 43 мсек, Среднее = 13 мсек
```

Рисунок 3.13 – Перевірка з'єднання з СКТВ

Під час тестування перевірялася коректність встановлення мережевого з'єднання з пристроєм, зображено на рис. 3.13, з якого відбувається з'єднання, та Raspberry Pi Zero 2W. Після підключення до локальної мережі користовуч отримував доступ до вебінтерфейсу через IP-адресу серверу. В результаті було підтверджено стабільне завантаження вебсторінки та можливість надсилання команд керування платформою.

Наступним етапом треба перевірити функціонування вебінтерфейсу. На сторінці керування відображається відеопотік з камери та набір кнопок для керування рухом платформи. Інтерфейс розроблений таким чином, щоб забезпечити швидкий доступ до основних функцій керування без встановлення додаткового програмного забезпечення на стороні користувача.

У процесі перевірки були протестовано відсилання команд. Кожна команда передавалася через вебінтерфейс на сервер Raspberry Pi Zero 2W, де обробляється компонентами далі вже за допомогою сигналів. Тести показали обробку повідомлень, що відображено на рисунку 3.14.

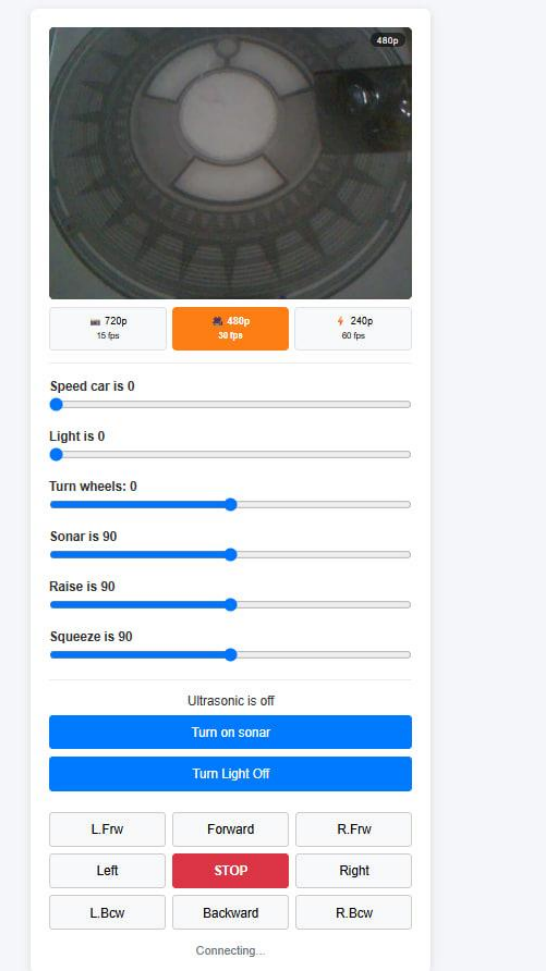

Рисунок 3.14 – Інтерфейс для керування СКТВ

Перевірена робота відеопотоку на вебсторінку в реальному часі. Це може дозволити переміщуватись роботизованій системі в режимі реального часу, що дозволяли би керувати користувачу це вживу. Передача даних здійснюється за допомогою Wi-Fi, з використанням WebSocket з'єднання.

Висновки до 3 розділу

Апаратну частину побудовано на основі Raspberry Pi Zero 2W із драйвером двигуна L298N для ШІМ-керування швидкістю, ШІМ-контролером PCA9685 I²C для незалежного керування чотирма серводвигунами SG90 та ультразвуковим далекоміром HC-SR04 із резистивним дільником напруги для узгодження рівнів

5 В/3,3 В. Реалізовано автоматичний аварійний зупин при виявленні перешкоди на відстані менше 10 см.

Відеострімінг реалізовано у форматі MJPEG через HTTP з трьома профілями якості: 1280 × 720 при 15 FPS, 640 × 480 при 30 кадрах FPS та 320 × 240 при 60 FPS. Перемикання між профілями виконується без перезавантаження системи шляхом динамічного перезапуску кодувальника picamera2.

Програмну частину реалізовано мовою Python на основі asyncio. Протокол WebSocket забезпечує постійне двостороннє з’єднання між браузером і сервером, через яке передаються JSON команди керування без перезавантаження сторінки. Опитування ультразвукового сенсора винесено в окремий потік із частотою 10 Гц, що забезпечує оновлення даних про відстань у реальному часі незалежно від команд керування. HTTP сервер із підтримкою ThreadingMixIn обслуговує одночасно кілька MJPEG клієнтів.

Отримані результати підтверджують практичну придатність обраних технічних рішень для побудови систем дистанційного керування з відеоспостереженням на базі малопотужних одноплатних комп’ютерів.

ВИСНОВКИ

В результаті виконання кваліфікаційної бакалаврської роботи було досліджено технічні характеристики існуючих СКТВ. Проаналізовано нормативно-правові документи для розроблення системи контролю вантажоперевезення. Сформовано технічні вимоги для створення СКТВ. Обрано засоби апаратної реалізації вбудованих систем контролю транспортування вантажів. змодельовано принципову схему та схему підключення апаратних компонентів. створено механізм захоплення вантажів та фотофіксації перешкод. Виконано експерементальні дослідження властивостей розробленої вбудованої системи.

У КБР розглянуто теоретичні засади та здійснено практичну реалізацію вбудованої системи контролю транспортування вантажів на базі одноплатного комп’ютера Raspberry Pi Zero 2W. Головною метою роботи є збереження життя та здоров’я людини у ситуаціях, де її безпосередня присутність пов’язана з ризиком або фізично неможлива. Компактні роботизовані системи здатні працювати там, де велика й громіздка техніка є безсилою: у вузьких складських проходах, у задимлених або хімічно небезпечних приміщеннях, у важкодоступних зонах після техногенних аварій чи стихійних лих. Саме в таких умовах дистанційно керований транспортний засіб із системою відеоспостереження стає інструментом, що дозволяє виконати завдання без ризику для людського життя.

Аналіз предметної області показав, що сучасні вбудовані системи контролю транспортування вантажів висувають жорсткі вимоги до швидкодії передачі команд, надійності відеозв’язку та автономності прийняття рішень у нештатних ситуаціях. Проведений огляд існуючих рішень виявив, що більшість комерційних аналогів або надмірно дорогі для масового застосування, або прив’язані до закритих екосистем і не допускають модифікації. Це підтвердило доцільність розробки власної відкритої платформи на базі доступних компонентів.

За результатами порівняльного аналізу апаратних платформ обрано Raspberry Pi Zero 2W як оптимальний варіант за співвідношенням обчислювальної потужності, компактності та вартості. Для організації каналу керування обрано

протокол WebSocket, що забезпечує двосторонній обмін даними в реальному часі без надлишкового навантаження на мережу. Формат MJPEG обрано для відеострімінгу як найбільш сумісний із браузерними клієнтами без необхідності встановлення додаткового програмного забезпечення.

Реалізована система об’єднує керування приводами, відеоспостереження з трьома профілями якості, автоматичне виявлення перешкод та зручний браузерний інтерфейс оператора. Багатопотокова архітектура програмного забезпечення забезпечує одночасну та незалежну роботу всіх підсистем: обробки команд керування, потокової передачі відео та моніторингу навколишнього середовища. Система повністю відповідає сформульованим вимогам і підтверджує практичну придатність обраних технічних рішень для задач дистанційного контролю транспортування вантажів.

Подальший розвиток системи може передбачати інтеграцію GPS-модуля для відстеження маршруту, датчиків температури та вологості для моніторингу умов зберігання вантажу, а також елементів технічного зору для часткової автономізації руху. Відкрита апаратна і програмна архітектура розробленої платформи створює для цього необхідне підґрунтя.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бурлаченко І. С., Гюльмамедов Н. М. Контролери PWM-Сигналів Для Керування Сервомоторами У Мультиагентних Роботизованих Системах. *Могилянські читання – 2025* : тези доп. XXVIII Всеукр. наук.-практ. конф. Миколаїв, 10–14 листоп. 2025 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2025. С. 58–63. URL: https://dspace.chmnu.edu.ua/jspui/bitstream/123456789/3066/1/%d0%9c%d0%a-2025.%20%d0%a2%d0%b5%d1%85%d0%bd%d1%96%d1%87%d0%bd%d1%96%20%d0%bd%d0%b0%d1%83%d0%ba%d0%b8.pdf (дата звернення 28.05.2026).

2. Kramar M., Kramar V. Research on Impact of IoT on Warehouse Management. *Sensors*. 2023. Vol. 23(4). № 2213. P. 1–30. DOI: 10.3390/s23042213.

3. Samadi Gharajeh M., Jond H. B. Speed Control for Leader-Follower Robot Formation Using Fuzzy System and Supervised Machine Learning. *Robotics Sensors*. 2021. Vol. 21. № 10:3433. P. 1–19. DOI: 10.3390/s21103433.

4. Tagliavini L., Colucci G., Botta A., Cavallone P., Baglieri L., Quaglia G. Wheeled Mobile Robots: State of the Art Overview and Kinematic Comparison Among Three Omnidirectional Locomotion Strategies. *Intelligent & Robotic Systems*. 2022. Vol. 106. № 57. P. 1–18. DOI: 10.1007/s10846-022-01745-7.

5. FOLO-4WD 150KG Automatic Human Following Cart Robot for Heavy Loads. URL: https://store.foxtech.com/folo-4wd-automatic-following-cart-robot-for-heavy-loads/ (Last accessed: 07.05.2026).

6. FOLO-300 4WD Automatic Wheeled Following Robot 300KG Heavy-Duty Chassis. URL: https://store.foxtech.com/folo-300-4wd-following-robot-heavy-duty-chassis/ (Last Accessed: 07.05.2026).

7. Mobile Ranger 4WS 4WD Robot (UGV). URL: https://www.generationrobots.com/en/404081-mobile-ranger-4ws-4wd-robot.html?srsltid=AfmBOoqpaEau4zK73m5lQ3yAsNlO52DeG0bwh7R2GUE_i_E1t5OZgxN_ (Last Accessed: 07.05.2026).

8. FOLO-100 PRO Heavy Duty 100KG Four-wheel Cargo Trolley Automatic Human Following Cart Robot. URL: https://store.foxtech.com/folo-100-100kg-human-following-robot/ (Last Accessed: 07.05.2026).

9. Bruzzone L., Nodehi S. E., Fanghella P. Tracked Locomotion Systems for Ground Mobile Robots: A Review. *Robotics Machines*. 2022. Vol. 10. № 8:648. P. 1–42. DOI: 10.3390/machines10080648.

10. Ugenti A., Galati R., Mantriota G., Reina G. Analysis of an All-Terrain Tracked Robot with Innovative Suspension System. *Mechanism and Machine Theory*. 2023. Vol. 182. № 105237. P. 1–21. DOI: 10.1016/j.mechmachtheory.2023.105237.

11. BUNKER MINI 2.0 User Manual. URL: https://cdn.shopify.com/s/files/1/0551/0630/6141/files/BUNKER_MINI_2.0_User_Manual.pdf?v=1773112703 (Last accessed: 07.05.2026).

12. AgileX Bunker Mini. URL: https://surl.li/rcdmgk (Last accessed: 07.06.2026).

13. Про затвердження Технічного регламенту радіообладнання : постанова КМУ від 15 берез. 2024 р. №335. URL: https://zakon.rada.gov.ua/laws/show/355-2017-%D0%BF#Text (дата звернення: 30.05.2026).

14. Про електронні комунікації : Закон України від 27 лютого 2026 р. №1089. URL: https://zakon.rada.gov.ua/laws/show/1089-20#Text (дата звернення: 29.05.2026).

15. Про затвердження Технічного регламенту низьковольтного електричного обладнання : постанова КМУ від 18 груд. 2025 р. №1067. URL: https://zakon.rada.gov.ua/laws/show/1067-2015-%D0%BF#Text (дата звернення: 30.05.2026).

16. Про захист інформації в інформаційно-комунікаційних системах : Закон України від 20 квіт. 2025 р. №80/94-ВР. URL: https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80#Text (дата звернення: 02.06.2026).

17. Про основні засади забезпечення кібербезпеки України : Закон України від 19 жовт. 2025 р. №2163-VIII. URL: https://zakon.rada.gov.ua/laws/show/2163-19#Text (дата звернення: 02.06.2026).

18. 16-канальний 12-bit PWM/Servo модуль з I2C інтерфейсом на PCA9685. URL: https://arduino.ua/prod1442-16-kanalnii-12-bit-pwmservo-modyl-s-i2c-interfeisom-na-pca9685. (дата звернення: 20.05.2026).

19. Модуль Wi-Fi ESP32-CAM з камерою 2MP. URL: https://arduino.ua/prod3458-modyl-wi-fi-esp32-s-kameroi-2mp (дата звернення: 21.05.2026).

20. Raspberry Pi 3 Model B. URL: https://miniboard.com.ua/mini-kompyuteri/272-raspberry-pi-3-model-b.html?srsltid=AfmBOopSvQIdTSpLAhsieIHhEaj97kaf1hNsE-jrs8I04QkiwIlaNjV1 (дата звернення 19.05.2026).

21. Orange Pi R1 Plus LTS. URL: https://arduino.ua/ru/prod6201-orange-pi-r1-plus-lts (дата звернення: 21.05.2026).

22. Мікрокомп'ютер Orange Pi Zero 2W 1GB. URL: https://www.rcscomponents.kiev.ua/product/mikrokompiuter-orange-pi-zero-2w-1gb_197660.html (дата звернення: 21.05.2026).

23. Raspberry Pi Zero 2 W. URL: https://arduino.ua/ru/prod6668-raspberry-pi-zero-2-w (дата звернення: 21.05.2026).

24. Модуль мініатюрної камери 5Мп для RPi Zero V1.3 від Waveshare. URL: https://arduino.ua/prod2759-modyl-miniaturnoi-5mp-kameri-dlya-raspberry-pi-zero (дата звернення: 21.05. 2026).

25. Камера Raspberry Pi Camera Board V2. URL: https://evo.net.ua/raspberry-pi-camera-board/?srsltid=AfmBOopitQO1cXmTLnO5JEkiPvTHf1d2x_Dq1VQXgQxmOPRZ9DIr2gDC (дата звернення: 20.05.2026).

26. Камера 8Мп RPi Camera V2. URL: https://arduino.ua/prod1555-kamera-8mp-dlya-raspberry-pi-sony-v2-original?srsltid=AfmBOorcXf23qkJ1sTJTaUs0Z-7FatpoPElHG9NYkc3rx3xfr5ecU7p (дата звернення: 20.05.2026).

27. Камера Raspberry Pi Camera Module 3. URL: https://evo.net.ua/kamera-raspberry-pi-camera-module-3/?srsltid=AfmBOoqk746QLV5aGSdijM15KmApc-Gul4mMXcJ3pkR-yidqZvluIEpc (дата звернення: 20.05.2026).

28. Камера Raspberry Pi Camera Module 3 (75°, 12МП, Sony IMX708). URL: https://arduino.ua/ru/prod5865-kamera-raspberry-pi-camera-module-3-75-12mp-sony-imx708?srsltid=AfmBOoqn_aqPnEznofu7dHm2tSM5qSWs-pY5HlRf11gY-RtyGWjEA2fk (дата звернення: 20.05.2026).

29. Мікродвигун з редуктором 6В 1000 RPM. URL: https://arduino.ua/prod3386-mikromotor-s-redyktorom-6v-1000-rpm (дата звернення 21.05.2026).

30. Мікро електродвигун AMS1141M від Elecrow. URL: https://arduino.ua/prod2238-micro-motor-ams1141m (дата звернення: 20.05.2026).

31. Двигун з редуктором 1:48 (одно-осьовий). URL: https://arduino.ua/prod3195-motor-s-redyktorom-148-odno-osevoi (дата звренення 19.05.2026).

32. Драйвер двох двигунів на L298N. URL: https://arduino.ua/prod406-draiver-dvyh-dvigatelei-na-l298n (дата звернення: 19.05.2026).

33. Модуль mini-L298N H-мост 9В. URL: https://arduino.ua/prod204-modyl-mini-l298n-h-most-9v (дата: 21.05.2026).

34. Драйвер двигунів двоканальний на TB6612FNG. URL: https://arduino.ua/prod2377-draiver-dvigatelei-dvyhkanalnii-na-tb6612fng (дата звернення: 19.05.2026).

35. Ультразвуковий датчик відстані HC-SR04. URL: https://arduino.ua/prod182-yltrazvykovoi-datchik-rasstoyaniya-hc-sr04 (дата звренення: 20.05.2026).

36. Модуль лазерного датчика відстані 3-200см TOF10120 VL53L0X. URL: https://arduino.ua/prod3926-modyl-lazernogo-datchika-vidstani-3-200sm-tof10120-vl53l0x (дата звернення: 20.05.2026).

37. RPI Zero 2W Board Layout: GPIO Pinout, Specs, Schematic in detail. URL: https://www.etechnophiles.com/rpi-zero-2w-board-layout-pinout-specs-price/ (дата звернення: 04.06.2026).

ДОДАТОК А

UML-діаграми для роботи програмного забезпечення

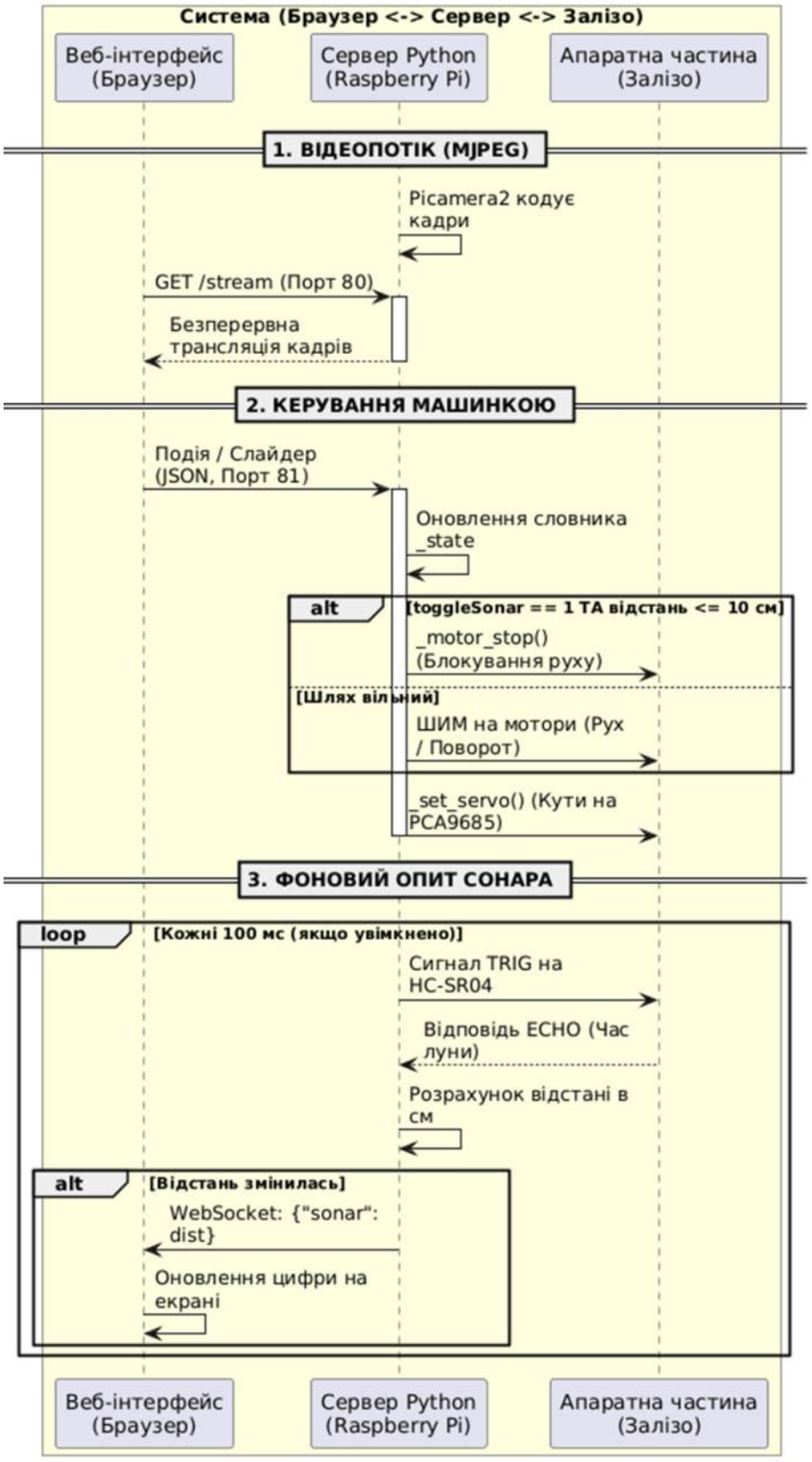

Рисунок А.1 – Діаграма для роботи алгоритму

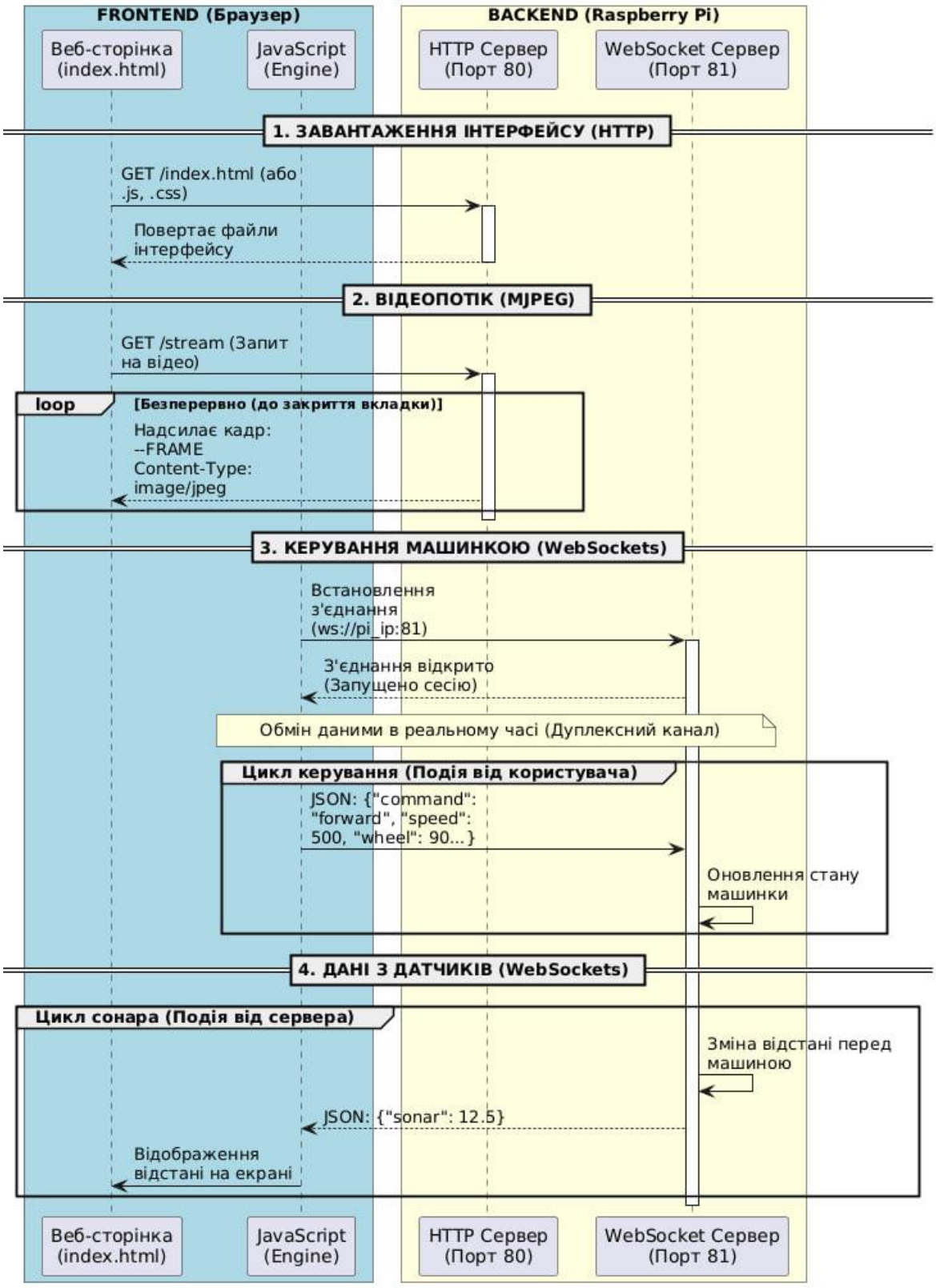

Рисунок А.2 – Діаграма робити програмного забезпечення вебінтерфейсу та вебсокет серверу

ДОДАТОК Б

Код програми

car_server.py

```
import asyncio
import json
import threading
import time
import io
import os
import socketserver
from http import server as http_server
from http.server import BaseHTTPRequestHandler
try:
import RPi.GPIO as GPIO
import board
import busio
from adafruit_pca9685 import PCA9685
from adafruit_motor import servo as adafruit_servo
HARDWARE = True
except ImportError:
print("GPIO/PCA9685 not found")
HARDWARE = False
try:
from picamera2 import Picamera2
from picamera2.encoders import MJPEGEncoder
from picamera2.outputs import FileOutput
import libcamera
CAMERA_AVAILABLE = True
except ImportError:
print("picamera2 not found — camera disabled")
CAMERA_AVAILABLE = False
try:
import websockets
PIN_ENA = 22
PIN_IN1 = 23
PIN_IN2 = 24
PIN_TRIG = 27
PIN_ECHO = 17
PIN_LED = 18
PRESETS: dict[str, dict] = {
"high": {"size": (1280, 720), "fps": 15, "quality": 90},
"medium": {"size": (640, 480), "fps": 30, "quality": 75},
"low": {"size": (320, 240), "fps": 60, "quality": 60},
}
class StreamingOutput(io.BufferedIOBase):
def __init__(self):
self.frame: bytes | None = None
self.condition = threading.Condition()
def write(self, buf: bytes) -> int:
with self.condition:
self.frame = buf
self.condition.notify_all()
return len(buf)
stream_output = StreamingOutput()
_camera: "Picamera2 | None" = None
_camera_lock = threading.Lock()
_current_preset = "medium"
def camera_init(preset_name: str = "medium") -> None:
global _camera, _current_preset
if not CAMERA_AVAILABLE:
return
preset = PRESETS[preset_name]
_camera = Picamera2()
config = _camera.create_video_configuration(
main={"size": preset["size"]},
controls={"FrameRate": float(preset["fps"])},
transform=libcamera.Transform(hflip=True, vflip=True),
)
_camera.configure(config)
encoder = MJPEGEncoder()
encoder.quality = preset["quality"]
```

```
_camera.start_recording(encoder, FileOutput(stream_output))
_current_preset = preset_name
print(f"[CAM] Started: {preset['size']} @ {preset['fps']} fps")
def camera_change_preset(preset_name: str) -> None:
global _current_preset
if not CAMERA_AVAILABLE or _camera is None:
return
if preset_name == _current_preset or preset_name not in PRESETS:
return
preset = PRESETS[preset_name]
print(f"[CAM] Changing preset → {preset_name}: {preset['size']} @ {preset['fps']} fps")
with _camera_lock:
try:
_camera.stop_recording()
config = _camera.create_video_configuration(
main={"size": preset["size"]},
controls={"FrameRate": float(preset["fps"])},
transform=libcamera.Transform(hflip=True, vflip=True),
)
_camera.configure(config)
encoder = MJPEGEncoder()
encoder.quality = preset["quality"]
_camera.start_recording(encoder, FileOutput(stream_output))
_current_preset = preset_name
print(f"[CAM] Preset changed successfully")
except Exception as exc:
print(f"[CAM] Error changing preset: {exc}")
def camera_shutdown() -> None:
if _camera is None:
return
try:
_camera.stop_recording()
_camera.close()
except Exception:
pass
SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))
class CarHTTPHandler(BaseHTTPRequestHandler):
def do_GET(self):
if self.path.startswith("/stream"):
self._handle_mjpeg()
elif self.path in ("/", "/index.html"):
self._serve_file("index.html", "text/html; charset=utf-8")
else:
fname = self.path.lstrip("/")
fpath = os.path.join(SCRIPT_DIR, fname)
if os.path.isfile(fpath):
ctype = "application/octet-stream"
if fname.endswith(".js"): ctype = "text/javascript"
if fname.endswith(".css"): ctype = "text/css"
if fname.endswith(".ico"): ctype = "image/x-icon"
self._serve_file(fname, ctype)
else:
self.send_error(404)
def _handle_mjpeg(self):
self.send_response(200)
self.send_header("Age", "0")
self.send_header("Cache-Control", "no-cache, private")
self.send_header("Pragma", "no-cache")
self.send_header("Content-Type", "multipart/x-mixed-replace; boundary=FRAME")
self.send_header("Access-Control-Allow-Origin", "*")
self.end_headers()
try:
while True:
with stream_output.condition:
got_frame = stream_output.condition.wait(timeout=1.0)
if not got_frame or stream_output.frame is None:
continue
frame = stream_output.frame
self.wfile.write(b"--FRAME\r\n")
self.wfile.write(b"Content-Type: image/jpeg\r\n")
self.wfile.write(f"Content-Length: {len(frame)}\r\n\r\n".encode())
self.wfile.write(frame)
self.wfile.write(b"\r\n")
except Exception:
pass
def _serve_file(self, filename: str, content_type: str):
fpath = os.path.join(SCRIPT_DIR, filename)
```

```
try:
with open(fpath, "rb") as f:
data = f.read()
self.send_response(200)
self.send_header("Content-Type", content_type)
self.send_header("Content-Length", str(len(data)))
self.end_headers()
self.wfile.write(data)
except FileNotFoundError:
self.send_error(404, f"File not found: {filename}")
def log_message(self, fmt, *args):
pass
class _ThreadedHTTPServer(socketserver.ThreadingMixIn, http_server.HTTPServer):
allow_reuse_address = True
daemon_threads = True
def _http_thread():
srv = _ThreadedHTTPServer(("", 80), CarHTTPHandler)
print("[HTTP] Server started on port 80")
srv.serve_forever()
_state: dict = {
"speed": 0, "light": 0, "command": "stop",
"lightTurnToggle": 0,
"wheel": 90, "yaw": 90, "pitch": 90, "roll": 90,
"toggleSonar": 0,
}
_sonar_distance: float = 0.0
_sonar_lock = threading.Lock()
_ws_clients: set = set()
_ws_clients_lock = asyncio.Lock()
_motor_pwm = None
_led_pwm = None
_pca = None
_servos = []
def gpio_init() -> None:
global _motor_pwm, _led_pwm, _pca, _servos
if not HARDWARE:
return
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
for pin in (PIN_ENA, PIN_IN1, PIN_IN2):
GPIO.setup(pin, GPIO.OUT)
_motor_pwm = GPIO.PWM(PIN_ENA, 1000)
_motor_pwm.start(0)
GPIO.setup(PIN_TRIG, GPIO.OUT)
GPIO.setup(PIN_ECHO, GPIO.IN)
GPIO.output(PIN_TRIG, False)
time.sleep(0.1)
GPIO.setup(PIN_LED, GPIO.OUT)
_led_pwm = GPIO.PWM(PIN_LED, 500)
_led_pwm.start(0)
i2c = busio.I2C(board.SCL, board.SDA)
_pca = PCA9685(i2c)
_pca.frequency = 50
for ch in range(4):
s = adafruit_servo.Servo(
_pca.channels[ch],
min_pulse=500,
max_pulse=2500,
)
_servos.append(s)
print("[GPIO] Initialized")
def gpio_cleanup() -> None:
if not HARDWARE:
return
_motor_stop()
if _motor_pwm: _motor_pwm.stop()
if _led_pwm: _led_pwm.stop()
if _pca: _pca.deinit()
GPIO.cleanup()
print("[GPIO] Cleaned up")
def _set_motor(speed_0_1024: int, in1: bool, in2: bool) -> None:
if not HARDWARE:
return
duty = min(100.0, speed_0_1024 / 1024 * 100)
_motor_pwm.ChangeDutyCycle(duty)
GPIO.output(PIN_IN1, GPIO.HIGH if in1 else GPIO.LOW)
GPIO.output(PIN_IN2, GPIO.HIGH if in2 else GPIO.LOW)
```

```
def _motor_stop(): _set_motor(0, False, False)
def _motor_forward(s): _set_motor(s, True, False)
def _motor_backward(s): _set_motor(s, False, True)
def _motor_left(s): _set_motor(s, True, False)
def _motor_right(s): _set_motor(s, True, False)
def _motor_lf(s): _set_motor(s, True, False)
def _motor_rf(s): _set_motor(s, True, False)
def _motor_lb(s): _set_motor(s//3, False, True)
def _motor_rb(s): _set_motor(s, False, True)
_CMD_MAP = {
"forward": _motor_forward,
"backward": _motor_backward,
"left": _motor_left,
"right": _motor_right,
"leftForward": _motor_lf,
"rightForward": _motor_rf,
"leftBackward": _motor_lb,
"rightBackward": _motor_rb,
"stop": lambda _: _motor_stop(),
}
def _set_servo(channel: int, degree: int) -> None:
degree = max(0, min(180, degree))
if not HARDWARE or channel >= len(_servos):
return
_servos[channel].angle = degree
def _set_led(light_0_256: int, toggle: int) -> None:
if not HARDWARE:
return
if toggle == 0:
_led_pwm.ChangeDutyCycle(0)
return
duty = max(0.0, min(100.0, (256 - light_0_256) / 256 * 100))
_led_pwm.ChangeDutyCycle(duty)
def _measure_distance() -> float:
if not HARDWARE:
return 0.0
GPIO.output(PIN_TRIG, True)
time.sleep(0.00001)
GPIO.output(PIN_TRIG, False)
start = time.time()
stop_t = start
deadline = time.time() + 0.04
while GPIO.input(PIN_ECHO) == 0:
start = time.time()
if start > deadline:
return -1.0
deadline = time.time() + 0.04
while GPIO.input(PIN_ECHO) == 1:
stop_t = time.time()
if stop_t > deadline:
return -1.0
return round((stop_t - start) * 34300 / 2, 1)
def _sonar_thread(loop: asyncio.AbstractEventLoop) -> None:
global _sonar_distance
last_sent: float = -999.0
while True:
if _state["toggleSonar"] == 1:
dist = _measure_distance()
with _sonar_lock:
_sonar_distance = dist
if dist >= 0 and dist != last_sent:
last_sent = dist
payload = json.dumps({"sonar": dist})
asyncio.run_coroutine_threadsafe(_broadcast(payload), loop)
time.sleep(0.1)
else:
last_sent = -999.0
time.sleep(0.05)
async def _broadcast(payload: str) -> None:
async with _ws_clients_lock:
clients = set(_ws_clients)
dead = set()
for ws in clients:
try:
await ws.send(payload)
except Exception:
dead.add(ws)
```

```
if dead:
async with _ws_clients_lock:
_ws_clients.difference_update(dead)
def _apply_state() -> None:
speed = _state["speed"]
cmd = _state["command"]
if _state["toggleSonar"] == 1:
with _sonar_lock:
dist = _sonar_distance
if 0 < dist <= 10:
_motor_stop()
_set_servo(0, _state["wheel"])
return
fn = _CMD_MAP.get(cmd, lambda _: _motor_stop())
fn(speed)
_set_servo(0, _state["wheel"])
_set_servo(1, _state["yaw"])
_set_servo(2, _state["pitch"])
_set_servo(3, _state["roll"])
_set_led(_state["light"], _state["lightTurnToggle"])
async def _ws_handler(websocket) -> None:
addr = websocket.remote_address
print(f"[WS] Connected: {addr}")
async with _ws_clients_lock:
_ws_clients.add(websocket)
try:
async for raw in websocket:
try:
data: dict = json.loads(raw)
except json.JSONDecodeError as exc:
print(f"[WS] JSON error: {exc}")
continue
if "changeResolution" in data:
preset = data["changeResolution"]
threading.Thread(
target=camera_change_preset,
args=(preset,),
daemon=True,
).start()
continue
_state.update({
"speed": int(data.get("speed", 0)),
"light": int(data.get("light", 0)),
"command": str(data.get("command", "stop")),
"lightTurnToggle": int(data.get("lightTurnToggle", 0)),
"wheel": int(data.get("wheel", 90)),
"yaw": int(data.get("yaw", 90)),
"pitch": int(data.get("pitch", 90)),
"roll": int(data.get("roll", 90)),
"toggleSonar": int(data.get("toggleSonar", 0)),
})
_apply_state()
except Exception as exc:
print(f"[WS] {addr} disconnected: {exc}")
finally:
async with _ws_clients_lock:
_ws_clients.discard(websocket)
print(f" Removed: {addr}")
async def main() -> None:
gpio_init()
camera_init("medium")
loop = asyncio.get_event_loop()
threading.Thread(target=_sonar_thread, args=(loop,), daemon=True).start()
threading.Thread(target=_http_thread, daemon=True).start()
print("WebSocket started on port 81")
async with websockets.serve(_ws_handler, "0.0.0.0", 81):
ip = os.popen("hostname -I").read().strip().split()[0]
print(f"\ Ready! Open browser: http://{ip}\n")
await asyncio.Future()
if __name__ == "__main__":
try:
asyncio.run(main())
except KeyboardInterrupt:
print("\nStopping")
finally:
camera_shutdown()
gpio_cleanup()
```

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=0.8">
<title>Pi Car Control</title>
<style>
body {
font-family: Arial, sans-serif;
background: #f4f6f9;
color: #333;
padding: 20px;
display: flex;
justify-content: center;
}
.container {
background: white;
padding: 20px;
border-radius: 8px;
box-shadow: 0 2px 10px rgba(0,0,0,0.1);
width: 100%;
max-width: 400px;
}
.row {
margin-bottom: 15px;
}
.row p {
margin: 0 0 5px 0;
font-size: 14px;
font-weight: bold;
}
input[type="range"] {
width: 100%;
margin: 0;
}
.grid {
display: grid;
grid-template-columns: repeat(3, 1fr);
gap: 8px;
margin: 15px 0;
}
button {
padding: 10px;
font-size: 14px;
border: 1px solid #ccc;
background: #f8f9fa;
border-radius: 4px;
cursor: pointer;
transition: background 0.2s;
}
button:hover { background: #e2e6ea; }
button:active { background: #dae0e5; }
#stop { background: #dc3545; color: white; border-color: #dc3545; font-weight: bold; }
#stop:hover { background: #c82333; }
.sys-btn { background: #007bff; color: white; border-color: #007bff; width: 100%; margin-bottom: 8px; }
.sys-btn:hover { background: #0069d9; }

.status { text-align: center; color: #6c757d; font-size: 13px; margin-top: 15px; }
.camera-section {
margin-bottom: 14px;
}

.camera-wrap {
position: relative;
background: #111;
border-radius: 6px;
overflow: hidden;
min-height: 160px;
display: flex;
align-items: center;
justify-content: center;
}
#cameraStream {
width: 100%;
height: auto;
```

```
display: block;
object-fit: contain;
border-radius: 6px;
}
.cam-placeholder {
position: absolute;
color: #555;
font-size: 12px;
text-align: center;
pointer-events: none;
user-select: none;
}
.cam-placeholder span {
display: block;
font-size: 28px;
margin-bottom: 4px;
}
.cam-badge {
position: absolute;
top: 6px;
right: 8px;
background: rgba(0,0,0,0.55);
color: #fff;
font-size: 10px;
padding: 2px 7px;
border-radius: 10px;
pointer-events: none;
letter-spacing: 0.4px;
}
.res-row {
display: flex;
gap: 6px;
margin-top: 8px;
}
.res-btn {
flex: 1;
padding: 8px 4px;
font-size: 11px;
line-height: 1.35;
border: 2px solid #dee2e6;
background: #f8f9fa;
border-radius: 5px;
cursor: pointer;
transition: all 0.18s;
text-align: center;
}
.res-btn:hover:not(.active) {
background: #e9ecef;
border-color: #adb5bd;
}
.res-btn.active {
background: #28a745;
color: #fff;
border-color: #28a745;
font-weight: bold;
}
.res-btn.active.quality-high { background: #dc3545; border-color: #dc3545; }
.res-btn.active.quality-medium { background: #fd7e14; border-color: #fd7e14; }
.res-btn.active.quality-low { background: #28a745; border-color: #28a745; }
</style>
</head>
<body>
<div class="container">
<div class="camera-section">
<div class="camera-wrap">
<img id="cameraStream" src="/stream" alt="">
<div class="cam-placeholder" id="camPlaceholder">
<span>📷</span>Waiting for video...
</div>
<div class="cam-badge" id="camBadge">480p · 30fps</div>
</div>
<div class="res-row">
<button class="res-btn quality-high" id="res-high" onclick="setRes('high')">
📸 720p<br>
</button>
<button class="res-btn quality-medium active" id="res-medium" onclick="setRes('medium')">
📠 480p<br>
```

```
</button>
<button class="res-btn quality-low" id="res-low" onclick="setRes('low')">
🗲 240p<br>
</button>
</div>
</div>
<hr style="border:0; border-top:1px solid #eee; margin:14px 0 16px;">
<div class="row">
<p id="speedLabel">Speed car is 0</p>
<input type="range" min="0" max="1024" value="0" id="slider">
</div>
<div class="row">
<p id="lightSuper">Light is 0</p>
<input type="range" min="0" max="256" value="0" id="light">
</div>
<div class="row">
<p id="wheelsServo">Steering: 90°</p>
<input type="range" min="0" max="180" value="90" id="wheel">
</div>
<div class="row">
<p id="yawServo">Sonar: 90°</p>
<input type="range" min="0" max="180" value="90" id="yaw">
</div>
<div class="row">
<p id="pitchServo">Claw lift: 90°</p>
<input type="range" min="0" max="180" value="90" id="pitch">
</div>
<div class="row">
<p id="rollUltrasonic">Claw grip: 90°</p>
<input type="range" min="0" max="180" value="90" id="roll">
</div>
<hr style="border:0; border-top:1px solid #eee; margin:15px 0;">
<p id="ultrasonic" style="margin:0 0 8px 0; font-size:14px; text-align:center;">Ultrasonic is off</p>
<button class="sys-btn" id="sonarBtn" onclick="sendToggleSonar()">Turn on sonar</button>
<button class="sys-btn" id="lightToggle" onclick="sendLightToggle()">Turn Light Off</button>
<div class="grid">
<button id="leftFrw" onclick="sendLeftForward()">L.Frw</button>
<button id="frw" onclick="sendForward()">Forward</button>
<button id="rightFrw" onclick="sendRigthForward()">R.Frw</button>
<button id="lft" onclick="sendLeft()">Left</button>
<button id="stop" onclick="sendStop()">STOP</button>
<button id="right" onclick="sendRigth()">Right</button>
<button id="leftBcw" onclick="sendLeftBackward()">L.Bcw</button>
<button id="bcw" onclick="sendBackward()">Backward</button>
<button id="rightBcw" onclick="sendRigthBackward()">R.Bcw</button>
</div>
<div class="status"><span>Connecting...</span></div>
</div>
<script>
const sliderEl = document.getElementById("slider");
const lightEl = document.getElementById("light");
const lightSuperEl = document.getElementById("lightSuper");
const speedLabelEl = document.getElementById("speedLabel");
const statusEl = document.querySelector("span");
const lightToggleBtn = document.getElementById("lightToggle");
const servoWheelEl = document.getElementById("wheel");
const servoWheelName = document.getElementById("wheelsServo");
const yawEl = document.getElementById("yaw");
const yawServoEl = document.getElementById("yawServo");
const pitchEl = document.getElementById("pitch");
const pitchServoEl = document.getElementById("pitchServo");
const rollEl = document.getElementById("roll");
const rollUltrasonicEl = document.getElementById("rollUltrasonic");
const ultrasonicEl = document.getElementById("ultrasonic");
const sonarBtnEl = document.getElementById("sonarBtn");
const camPlaceholder = document.getElementById("camPlaceholder");
const camBadge = document.getElementById("camBadge");
const camImg = document.getElementById("cameraStream");
let message = [0, 0, "stop", 0, 90, 90, 90, 90, 0];
let lastCommand = null;
let lastSpeedVal = 0;
let lastLightVal = 0;
let lastWheelVal = 90;
let lastYawVal = 90;
let lastPitchVal = 90;
let lastRollVal = 90;
let lightStateNow = 0;
```

```
let lastSonarCounter = 0;
let distanceUltrasonic = 0;
let currentRes = "medium";
const PRESET_LABELS = {
high: "720p · 15fps",
medium: "480p · 30fps",
low: "240p · 60fps",
};
camImg.addEventListener("load", () => {
if (camImg.naturalWidth > 0) camPlaceholder.style.display = "none";
});
camImg.addEventListener("error", () => {
camPlaceholder.style.display = "flex";
});
function setRes(res) {
if (res === currentRes) return;
currentRes = res;
["high","medium","low"].forEach(r => {
document.getElementById("res-" + r).classList.toggle("active", r === res);
});
camBadge.textContent = PRESET_LABELS[res];
if (ws.readyState === WebSocket.OPEN) {
ws.send(JSON.stringify({ changeResolution: res }));
}
camPlaceholder.style.display = "flex";
camImg.src = "";
setTimeout(() => {
camImg.src = "/stream?t=" + Date.now();
}, 450);
}
const ws = new WebSocket(`ws://${location.hostname}:81`);
ws.onopen = () => { statusEl.innerHTML = "Connected"; };
ws.onclose = () => { statusEl.innerHTML = "Disconnected"; };
ws.onerror = () => { statusEl.innerHTML = "Error"; };

ws.onmessage = function(event) {
try {
const resp = JSON.parse(event.data);
if (resp.sonar !== undefined) {
distanceUltrasonic = resp.sonar;
if (lastSonarCounter === 1) {
ultrasonicEl.innerHTML = "📡 Distance: " + distanceUltrasonic + " cm";
}
}
} catch(e) {}
};
function sendMsg() {
if (ws.readyState !== WebSocket.OPEN) return;
ws.send(JSON.stringify({
speed: message[0],
light: message[1],
command: message[2],
lightTurnToggle: message[3],
wheel: message[4],
yaw: message[5],
pitch: message[6],
roll: message[7],
toggleSonar: message[8],
}));
}
sliderEl.addEventListener("input", () => {
if (lastSpeedVal == sliderEl.value) return;
message[0] = sliderEl.value;
speedLabelEl.textContent = "Speed car is " + sliderEl.value;
lastSpeedVal = sliderEl.value;
sendMsg();
});
document.addEventListener("wheel", (e) => {
e.preventDefault();
let step = 0;
if (e.deltaMode === 0) step = e.deltaY * 0.5;
if (e.deltaMode === 1) step = e.deltaY * 20;
if (e.deltaMode === 2) step = e.deltaY * 200;
sliderEl.value = Math.min(1024, Math.max(0, parseInt(sliderEl.value) - step));
message[0] = sliderEl.value;
speedLabelEl.textContent = "Speed car is " + sliderEl.value;
lastSpeedVal = sliderEl.value;
```

```
sendMsg();
}, { passive: false });
lightEl.addEventListener("input", () => {
if (lastLightVal == lightEl.value) return;
lightSuperEl.innerHTML = "Light is " + lightEl.value;
message[1] = lightEl.value;
lastLightVal = lightEl.value;
sendMsg();
});
servoWheelEl.addEventListener("input", () => {
if (lastWheelVal == servoWheelEl.value) return;
servoWheelName.innerHTML = "Steering: " + servoWheelEl.value + "°";
message[4] = servoWheelEl.value;
lastWheelVal = servoWheelEl.value;
sendMsg();
});
yawEl.addEventListener("input", () => {
if (lastYawVal == yawEl.value) return;
yawServoEl.innerHTML = "Sonar: " + yawEl.value + "°";
message[5] = yawEl.value;
lastYawVal = yawEl.value;
sendMsg();
});
pitchEl.addEventListener("input", () => {
if (lastPitchVal == pitchEl.value) return;
pitchServoEl.innerHTML = "Claw lift: " + pitchEl.value + "°";
message[6] = pitchEl.value;
lastPitchVal = pitchEl.value;
sendMsg();
});
rollEl.addEventListener("input", () => {
if (lastRollVal == rollEl.value) return;
rollUltrasonicEl.innerHTML = "Claw grip: " + rollEl.value + "°";
message[7] = rollEl.value;
lastRollVal = rollEl.value;
sendMsg();
});
const KEY_CMD = { w: "forward", s: "backward", a: "left", d: "right" };
document.addEventListener("keydown", (e) => {
const cmd = KEY_CMD[e.key];
if (cmd && lastCommand !== cmd) {
message[2] = cmd;
lastCommand = cmd;
sendMsg();
return;
}
if (e.key === " " && lastCommand !== "stop") {
sendStop();
return;
}
if (e.key === "z") {
lightEl.value = Math.min(256, parseInt(lightEl.value) + 10);
lightSuperEl.innerHTML = "Light is " + lightEl.value;
message[1] = lightEl.value;
sendMsg();
}
if (e.key === "x") {
lightEl.value = Math.max(0, parseInt(lightEl.value) - 10);
lightSuperEl.innerHTML = "Light is " + lightEl.value;
message[1] = lightEl.value;
sendMsg();
}
if (e.key === "l") sendLightToggle();
// Change resolution with keys 1/2/3
if (e.key === "1") setRes("high");
if (e.key === "2") setRes("medium");
if (e.key === "3") setRes("low");
});
document.addEventListener("keyup", (e) => {
if (KEY_CMD[e.key]) lastCommand = null;
});
function sendLeftForward() { message[2] = "leftForward"; sendMsg(); }
function sendForward() { message[2] = "forward"; sendMsg(); }
function sendRigthForward() { message[2] = "rightForward"; sendMsg(); }
function sendLeft() { message[2] = "left"; sendMsg(); }
function sendStop() { message[2] = "stop"; sendMsg(); }
function sendRigth() { message[2] = "right"; sendMsg(); }
```

```
function sendLeftBackward() { message[2] = "leftBackward"; sendMsg(); }
function sendBackward() { message[2] = "backward"; sendMsg(); }
function sendRigthBackward() { message[2] = "rightBackward"; sendMsg(); }
function sendLightToggle() {
if (lightStateNow === 0) {
lightStateNow = 1;
message[3] = 1;
lightToggleBtn.innerHTML = "Turn Light On";
} else {
lightStateNow = 0;
message[3] = 0;
lightToggleBtn.innerHTML = "Turn Light Off";
}
sendMsg();
}
function sendToggleSonar() {
if (lastSonarCounter === 0) {
message[8] = 1;
lastSonarCounter = 1;
sonarBtnEl.innerHTML = "Sonar turn Off";
ultrasonicEl.innerHTML = "📡 Measuring...";
} else {
message[8] = 0;
lastSonarCounter = 0;
sonarBtnEl.innerHTML = "Turn on sonar";
ultrasonicEl.innerHTML = "Ultrasonic is off";
}
sendMsg();
}
</script>
</body>
</html>
```

ДОДАТОК В

Матеріали апробації роботи

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2025:
досвід та тенденції розвитку суспільства в Україні: глобальний,
національний та регіональний аспекти»**

XXVIII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 10–14 листопада 2025 року

Миколаїв – 2025

Підсекції:

- **КОМП’ЮТЕРНА ІНЖЕНЕРІЯ**