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

Комплекс дистанційного спостереження на базі колісної робоплатформи та ESP32-CAM

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія»

Здобувач _____Ілля ГУЛЯЄВ_____
підпис

«__» _____ 202__ р.

Керівник PhD, ст. викл. каф. КІ _____Євген ДАРНАПУК_____
підпис
«__» _____ 202__ р.

Миколаїв – 2025

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
«_____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Гуляєв Ілля Сергійович

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи

Комплекс дистанційного спостереження на базі колісної робоплатформи та ESP32-CAM

Затверджена наказом по ЧНУ ім. Петра Могили від 30.10.2024 № 297.

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

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Розробка функціонального прототипу системи дистанційного відеоспостереження на базі мобільної колісної платформи з використанням мікроконтролера ESP32-CAM.

4. Перелік питань, що підлягають розробці:

1) аналіз сучасних рішень та технологій у галузі мобільних систем дистанційного спостереження;

2) вибір апаратного забезпечення для побудови мобільної платформи;

3) розробка архітектура АПК;

4) розробка програмного забезпечення для управління платформою та передавання відеоданих;

5) провести експериментальні дослідження функціональності комплексу та оцінити його ефективність у різних умовах експлуатації.

5. Перелік графічних матеріалів
Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Євген ДАРНАПУК
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Ілля ГУЛЯЄВ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Комплекс дистанційного спостереження на базі колісної робоплатформи та ESP32-CAM

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	30.10.2024	30.10.2024	Виконано
2	Огляд літератури за темою роботи	01.11.2024	15.12.2024	Виконано
3	Складання календарного плану КБР	10.12.2024	20.12.2024	Виконано
4	Аналіз предметної області	02.01.2025	25.01.2025	Виконано
5	Розробка проєктних рішень	26.01.2025	28.02.2025	Виконано
6	Моделювання та конструювання АПК	01.03.2025	31.03.2025	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПК, аналіз результатів тестування	01.04.2025	30.04.2025	Виконано
8	Оформлення КБР та презентації	11.05.2025	04.06.2025	Виконано
9	Попередній захист	21.05.2025	04.06.2025	Виконано
10	Відгук керівника КБР	01.05.2025	10.06.2025	Виконано
11	Рецензування	04.06.2025	15.06.2025	Виконано
12	Завершення оформлення КБР та презентації	16.06.2025	17.06.2025	Виконано
13	Захист кваліфікаційної бакалаврської роботи	25.06.2025	25.06.2025	Виконано

Керівник роботи _____ Євген ДАРНАПУК

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Ілля ГУЛЯЄВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Комплекс дистанційного спостереження
на базі колісної робоплатформи та ESP32-CAM»
Студент 405 гр.: Гуляєв Ілля Сергійович
Керівник: PhD, ст.викл Дарнапук Євген Сергійович

Дана кваліфікаційна бакалаврська присвячена розробці комплексу дистанційного спостереження на базі колісної роботизованої платформи з використанням мікроконтролера ESP32-CAM. Метою є розробка мобільної, автономної системи відеомоніторингу в режимі реального часу, яка може передавати зображення з небезпечних або важкодоступних зон без безпосереднього втручання людини. Такий підхід зменшує ризики для життя людей у критичних умовах, таких як зони бойових дій або зони стихійних лих. Актуальність даного дослідження полягає у зростаючому попиті на компактні, недорогі та автономні системи спостереження, здатні працювати автономно в небезпечних або важкодоступних місцях. Об'єктом дослідження є комплекс віддаленого відеоспостереження, а предметом - апаратні та програмні компоненти системи на базі ESP32-CAM. Метою роботи є розробка та тестування функціонального прототипу, здатного дистанційно захоплювати, передавати та обробляти візуальну інформацію. Практичне значення системи полягає в її потенційному застосуванні у сфері реагування на надзвичайні ситуації, моніторингу безпеки та військової розвідки, що дозволяє отримувати інформацію про ситуацію в реальному часі без загрози для життя людей.

Розроблена система складається з мікроконтролера ESP32-CAM, що відповідає за зйомку відео та бездротову передачу даних, колісної роботизованої платформи для мобільності, драйвера двигуна L298N для керування рухом та джерела живлення на основі Li-Ion акумуляторів. Всі компоненти інтегровані в компактну, енергоефективну конструкцію з можливістю автономної роботи. Програмне забезпечення для керування реалізовано в середовищі Arduino IDE і включає базовий алгоритм навігації платформою та трансляцію відео через Wi-Fi.

Пояснювальна записка бакалаврської роботи складається зі вступу, трьох розділів, висновків та двох додатків. У першому розділі представлено аналітичний огляд існуючих рішень та визначено системні вимоги. Другий розділ описує вибір елементів та вимоги до компонентів. Третій розділ присвячений проектуванню та реалізації апаратного забезпечення, підключенню, програмному управлінню, а також інтеграції функцій передачі відеосигналу.

Апробація результатів кваліфікаційної роботи відбулася під час Всеукраїнської науково-практичної конференції «Могилянські читання 2024» (Миколаїв, 6–8 листопада 2024 р.).

Кількість сторінок 69 (без додатків), таблиць 10, рисунків 46, використаних джерел посилання 26 та додатків 2.

Ключові слова: ESP32-CAM, OV2640, дистанційне спостереження, роботизована платформа, мобільний моніторинг, передача зображення, L298N, автономна система.

ABSTRACT

of the Bachelor's Thesis

"Remote monitoring system based on a wheeled robot platform and ESP32-CAM"

Student: Illia Huliaiev

Supervisor: PhD, senior lecturer Yevhen Darnapuk

This bachelor's thesis focuses on the development of a remote surveillance complex based on a wheeled robotic platform using the ESP32-CAM microcontroller. The aim of the project is to design a mobile, autonomous system for real-time video monitoring that can transmit images from hazardous or hard-to-reach areas without direct human intervention. This approach reduces risks to human life in critical environments such as combat zones or disaster areas. The relevance of this research lies in the growing demand for compact, low-cost, and autonomous surveillance systems capable of operating independently in dangerous or inaccessible locations. The object of the study is a remote video surveillance complex, while the subject includes the hardware and software components of the system based on ESP32-CAM. The purpose of the work is to develop and test a functional prototype capable of capturing, transmitting, and processing visual information remotely. The practical significance of the system is its potential application in emergency response, security monitoring, and military reconnaissance, allowing real-time situational awareness without endangering human lives.

The developed system consists of an ESP32-CAM microcontroller responsible for capturing video and wireless data transmission, a wheeled robotic platform for mobility, an L298N motor driver for motion control, and a power source based on Li-Ion batteries. All components are integrated into a compact, energy-efficient structure with the ability to operate autonomously. The control software was implemented in the Arduino IDE and includes a basic algorithm for platform navigation and video streaming over Wi-Fi.

The explanatory note of the bachelor's thesis includes an introduction, three chapters, conclusions, and two appendices. The first chapter presents an analytical overview of existing solutions and defines system requirements. The second section describes the selection of elements and the requirements for the components. The third chapter focuses on the design and implementation of the hardware, wiring, and software control, as well as the integration of video transmission functionality.

The research results were presented and discussed at the All-Ukrainian Scientific and Practical Conference «Petro Mohyla Readings 2024» (Mykolaiv, November 6–8, 2024).

The paper consists of 69 pages (excluding appendices), 10 tables, 46 figures, 26 references and 2 appendices.

Keywords:ESP32-CAM, OV2640, remote surveillance, robotic platform, wireless video transmission, Arduino, autonomous system, L298N.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПК – апаратно-програмний комплекс

КБР – кваліфікаційна бакалаврська робота

ОС –операційна система

GUI – Graphical User Interface

IoT – Internet of Things

HTTP – Hypertext Transfer Protocol

WPA – Wi-Fi Protected Access

ВСТУП

В умовах сучасних загроз, пов’язаних з безпекою об'єктів інфраструктури, особливо в контексті військових дій та зростаючої небезпеки у контрольованих зонах, особливої актуальності набуває розвиток автономних мобільних систем дистанційного спостереження. Відсутність потреби у фізичній присутності оператора на небезпечній території дозволяє мінімізувати ризики для життя та забезпечити безперервний моніторинг складних або віддалених ділянок. Технічний прогрес у галузі мікроелектроніки, зокрема поява доступних і потужних рішень на базі ESP32- CAM [10], створює умови для широкого застосування дистанційних роботизованих платформ.

Практичне значення роботи полягає в розробці економічно доступного, мобільного комплексу для дистанційного візуального моніторингу, який може застосовуватися в різних сферах: від охорони об'єктів до проведення розвідки у важкодоступних або потенційно небезпечних місцях.

Мета роботи полягає у забезпечення безпеки на об'єктах інфраструктури та контрольованих зонах для дистанційного моніторингу небезпечних територій за рахунок розробки АПК на базі колісної робоплатформи та мікроконтролера ESP32-CAM.

Для досягнення визначеної мети необхідно:

– провести аналіз сучасних рішень та технологій у галузі мобільних систем дистанційного спостереження;

– обґрунтувати вибір апаратного забезпечення для побудови мобільної платформи;

– розробити архітектуру АПК;

– реалізувати програмне забезпечення для управління платформою та передавання відеоданих;

– провести експериментальні дослідження функціональності комплексу.

Об'єкт дослідження – процес дистанційного моніторингу об'єктів і територій за допомогою мобільних роботизованих платформ.

Предмет дослідження – АПК на базі колісної робоплатформи з мікроконтролером ESP32-CAM для збору, обробки і передавання візуальної інформації.

Необхідність розробки подібного комплексу обумовлюється обмеженими можливостями традиційних систем відеоспостереження, які часто вимагають стаціонарного розміщення і не забезпечують гнучкого реагування на зміну ситуації у реальному часі. Аналіз сучасного стану показує, що на ринку існують дорогі комерційні рішення на базі спеціалізованих роботизованих платформ, однак їх вартість та складність експлуатації обмежують масове використання. Тому створення доступної за ціною мобільної системи із використанням відкритих апаратних і програмних рішень є актуальним завданням.

Світові тенденції у цій галузі вказують на зростання популярності застосування IoT-пристроїв та систем штучного інтелекту для покращення автономності та інтелектуальних можливостей роботизованих платформ. Провідні фірми, як Boston Dynamics та Unitree Robotics, активно досліджують можливості створення автономних мобільних платформ для задач моніторингу та розвідки, однак такі рішення є надзвичайно складними та дорогими для масового застосування.

Сфера застосування результатів включає:

– дистанційний моніторинг небезпечних територій;
– патрулювання охоронюваних об'єктів;
– екологічний моніторинг територій;
– розвідку у зонах обмеженого доступу без ризику для людини;
– освітні проекти у сфері робототехніки та розробки автономних систем.

Обґрунтування основних проєктних рішень полягає у використанні мікроконтролера ESP32-CAM як базового обчислювального модуля завдяки

його компактності, енергоефективності, наявності вбудованої камери та підтримки бездротового передавання даних через Wi-Fi. Колісна робоплатформа на базі комплекту Smart Robot Chassis Kit [11] та драйвера L298N [12] забезпечує мобільність та маневреність комплексу за умов мінімальних фінансових витрат.

Апробація результатів кваліфікаційної роботи відбулася під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання – 2024» (Миколаїв, 06–10 листопада 2024 р.).

Публікації. Основні положення роботи опубліковані у збірнику матеріалів XXVII Всеукраїнської науково-практичної конференції «Могилянські читання – 2024» [22].

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

В умовах сучасних загроз і викликів, з якими стикається Україна, виникає гостра потреба у створенні автономних і надійних засобів дистанційного спостереження за потенційно небезпечними територіями. Одним із таких засобів є мобільний комплекс на базі колісної роботизованої платформи, оснащеної мікроконтролером ESP32-CAM, який здатний забезпечити отримання візуальної інформації з віддалених або важкодоступних місць без ризику для життя людей.

1.1 Об'єкт і предмет дослідження

Об'єктом дослідження у даній КБР є мобільна система дистанційного відеоспостереження, що базується на інтеграції апаратних та програмних засобів для забезпечення оперативного збору, передавання та обробки візуальної інформації.

Предметом дослідження є апаратно-програмне рішення, що включає в себе мікроконтролер ESP32-CAM, колісну роботизовану платформу на базі Smart Robot Chassis Kit, драйвер моторів L298N та джерело автономного живлення, об'єднані в єдину систему для реалізації завдань мобільного відеоспостереження в режимі реального часу.

Метою дослідження є розробка і дослідження ефективної апаратно-програмної системи, що забезпечує дистанційний моніторинг територій із використанням бездротової передачі даних та можливістю автоматизованого аналізу отриманого відеопотоку.

Таким чином, дослідження зосереджено на створенні мобільної платформи, яка може автономно переміщатися та передавати якісне відеозображення до серверної частини системи для подальшої обробки за допомогою алгоритмів комп'ютерного зору, що істотно підвищує ефективність моніторингу та безпеку персоналу.

1.2 Огляд існуючих рішень системи, що розробляється

На сьогодні існує низка рішень у сфері дистанційного відеоспостереження із використанням мобільних роботизованих платформ. Більшість із них спрямовані на військове застосування, цивільне спостереження, промислову автоматизацію або наукові дослідження. У даному підрозділі розглянемо найбільш поширені приклади таких систем, їхні характеристики, переваги та обмеження, а також визначимо місце запропонованого рішення на базі ESP32-CAM серед сучасних технологій.

1.2.1 Роботизовані системи на базі Raspberry Pi

Одним із популярних варіантів побудови мобільних спостережних систем є використання одноплатного комп'ютера Raspberry Pi у поєднанні з камерою Raspberry Pi Camera Module [1] (рис. 1.1). Такі системи дозволяють реалізувати складні алгоритми обробки відео, комп’ютерного зору та штучного інтелекту безпосередньо на платформі завдяки достатньо високій обчислювальній потужності (4-ядерний процесор ARM Cortex-A72 у версії Raspberry Pi 4).

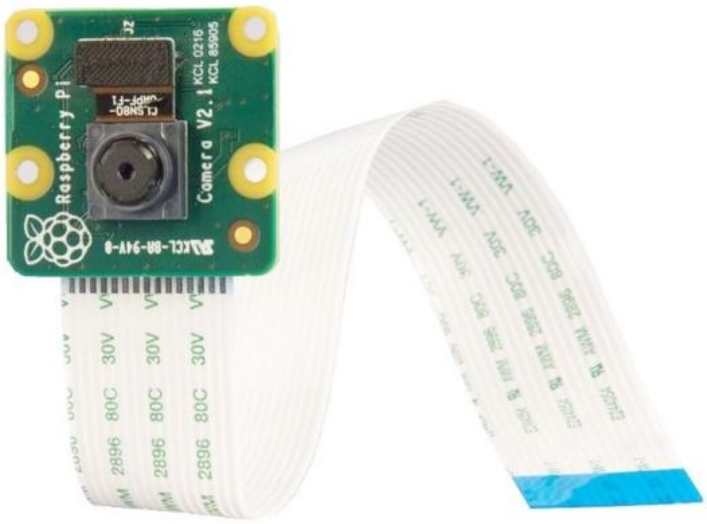


Рисунок 1.1 – Raspberry Pi Camera Module [1]

Переваги рішень на Raspberry Pi:

– потужні обчислювальні ресурси для локальної обробки відео та зображень;

– широка підтримка бібліотек (OpenCV, TensorFlow, PyTorch тощо);

– велика спільнота користувачів і документація.

Недоліки:

– високе енергоспоживання (порівняно з мікроконтролерами) – потребує використання більших акумуляторів або зовнішніх джерел живлення;

– вища вартість комплектуючих;

– складність налаштування систем безпеки у бездротових мережах.

Таким чином, хоча Raspberry Pi є чудовою платформою для складних проєктів, для завдань економного та довготривалого автономного спостереження його використання часто є недоцільним через надлишкові ресурси та обмежену енергоефективність.

1.2.2 Комерційні рішення

Серед комерційних рішень, орієнтованих на навчання, дослідження та частково на задачі моніторингу, окреме місце займає платформа DJI RoboMaster S1 (рис. 1.2) [4].

Це багатофункціональний робот, розроблений для інтерактивного навчання програмуванню та робототехніці, проте він також демонструє ряд можливостей, які можуть бути використані у системах відеоспостереження. S1 оснащений камерою зі стабілізацією, модулями для розпізнавання об’єктів, функцією автослідування, вбудованою ОС, а також можливістю програмування за допомогою Python або Scratch.

Рисунок 1.2 – DJI RoboMaster S1 [4]

Переваги RoboMaster S1:

– висока якість виготовлення та надійність;
– вбудовані алгоритми розпізнавання та автономної навігації;
– широкий функціонал і можливості розширення.

Недоліки:

– дуже висока вартість (від 500 USD і більше);
– закритість деяких частин ПЗ і обмежена можливість модифікації для нестандартних завдань;
– орієнтованість більше на навчальні процеси, ніж на реальні польові умови експлуатації.

Аналогами RoboMaster S1 є інші комерційні освітні чи дослідницькі платформи, зокрема Makeblock mBot Ranger, Parallax ActivityBot 360, Robotis TurtleBot3 та інші.

Makeblock mBot Ranger [5] – це роботизований комплект, створений для STEM-освіти, однак має модулі для підключення камер, датчиків і контролерів руху. Оснащується Bluetooth або WiFi-модулями, що дозволяє передавати дані

на мобільні пристрої. Відносно недорогий, простий у налаштуванні, але потребує додаткових модулів для реалізації повноцінного відеоспостереження.

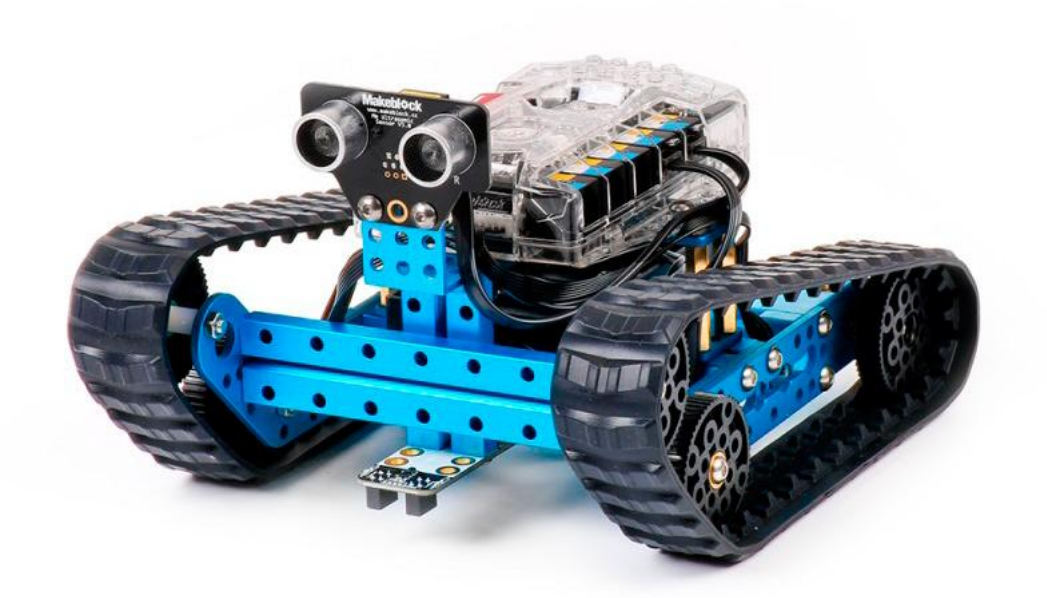


Рисунок 1.3 – Makeblock mBot Ranger [5]

Переваги Makeblock mBot Ranger:

– широка підтримка у сфері STEM;
– модульна архітектура;
– простота інтеграції з камерами.

Недоліки:

– обмежені можливості для реального часу;
– не призначений для польових умов.

Parallax ActivityBot 360 [6] – це платформа на базі мікроконтролера Propeller, яка використовується у вивченні автономних роботів. Підтримує додавання камер і базових систем комп’ютерного зору, але обмежена в обчислювальних потужностях та швидкості передачі даних. Може працювати в обмежених інтер’єрних умовах.

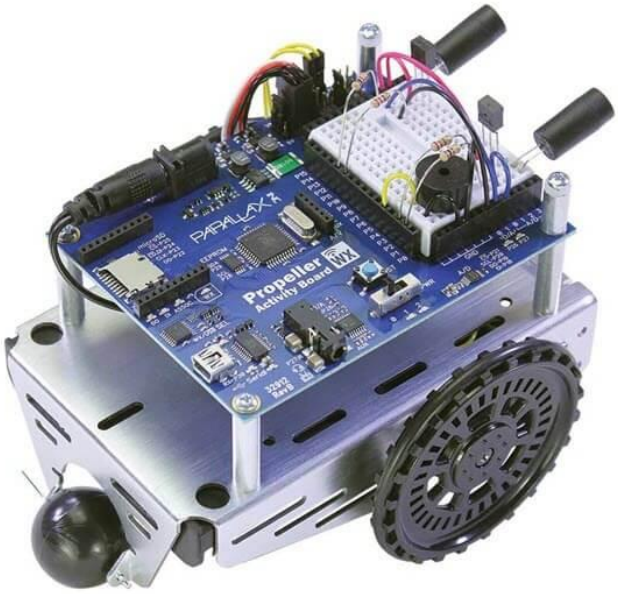


Рисунок 1.4 – Parallax ActivityBot 360 [6]

Переваги:

– надійна апаратна реалізація;
– підтримка автономного програмування руху;
– гнучкість у налаштуваннях.

Недоліки:

– відсутність вбудованої обробки відео;
– вузька спеціалізація та висока ціна для свого функціоналу.

Robotis TurtleBot3 [2] – це більш потужне рішення для дослідницьких цілей, сумісне з ROS (Robot Operating System). TurtleBot3 дозволяє реалізовувати складні навігаційні алгоритми, SLAM, а також розпізнавання об'єктів через підключення камер (наприклад, RealSense). Проте ціна такого рішення значно перевищує бюджет спостережної системи, побудованої на ESP32-CAM.

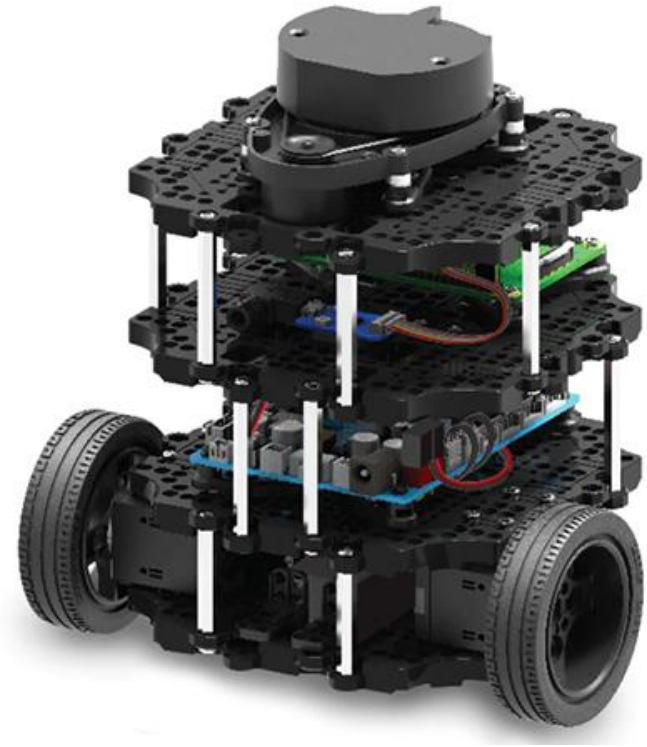


Рисунок 1.5 – Robotis TurtleBot3 [2]

Переваги:

– ROS-сумісність;
– потужне програмне середовище;
– широкі можливості сенсорної інтеграції.

Недоліки:

– дуже висока ціна (від 500 до 1500 USD);
– надлишкова складність для простих задач відеоспостереження.

Таким чином, хоча існуючі комерційні платформи – такі як DJI RoboMaster S1, TurtleBot3, mBot Ranger та інші – пропонують широкі функціональні можливості, включно з автономною навігацією, відеозахопленням та розпізнаванням об’єктів, їхнє застосування в умовах обмеженого бюджету та підвищеного ризику є малоефективним. Основними обмеженнями є висока вартість, надмірна складність реалізації, вимоги до спеціалізованого програмного забезпечення та енергоспоживання. Більшість з них орієнтовані на освітню чи дослідницьку сферу, а не на польові завдання оперативного моніторингу.

Відповідно, для реалізації економічної, мобільної та надійної системи дистанційного спостереження, доцільніше використовувати адаптовані рішення на основі ESP32-CAM, які, хоч і мають нижчий рівень інтеграції та автономності, забезпечують необхідний баланс між функціональністю, гнучкістю та вартістю для роботи в реальних умовах з ризиками.

1.2.3 Прості мобільні рішення на основі Arduino

Іншим поширеним варіантом є створення простих роботизованих платформ на основі мікроконтролерів Arduino Uno, Nano або Mega у поєднанні із зовнішніми модулями камер наприклад, OV7670 (рис. 1.3) [9]. Такі системи використовуються переважно для базових проєктів дистанційного керування або автономної навігації.

Рисунок 1.6 – Камера для Arduino OV7670 [9]

Переваги Arduino-систем:

– низька вартість комплектуючих;

– простота реалізації базових алгоритмів управління рухом;

– велика кількість прикладів та документації.

Недоліки:

– відсутність обчислювальної потужності для обробки відео у реальному часі;

– залежність від додаткових модулів (Wi-Fi, камери, передавачів даних);

– низька якість відео та обмежена роздільна здатність зображення.

Таким чином, Arduino підходить для дуже простих систем, проте не забезпечує потрібного рівня обробки та передачі візуальної інформації для завдань моніторингу.

1.2.4 Готові рішення на базі ESP32

Окрім розробки власних платформ, на ринку представлені готові рішення на базі ESP32, наприклад TTGO T-Camera (рис. 1.4) [8] або M5Stack Camera (рис. 1.5) [3]. Ці пристрої оснащені інтегрованими камерами та модулями бездротового зв'язку.

Рисунок 1.7 – TTGO T-Camera [8]

Рисунок 1.8 – M5Stack Camera [3]

Переваги готових рішень:

– висока інтеграція компонентів;

– простота розгортання і налаштування;

– підтримка розширених функцій, таких як розпізнавання облич за допомогою ESP-WHO.

Недоліки:

– обмежені можливості модифікації апаратної частини;

– вища вартість у порівнянні з окремими модулями ESP32-CAM.

Таким чином, хоча такі рішення значно спрощують розробку, вони не завжди дозволяють оптимізувати платформу під специфічні завдання, що обмежує їхнє використання у спеціалізованих проєктах.

1.2.5 Професійні розвідувальні платформи

Recon Scout Throwbot 2 (рис. 1.6) [7] – це професійний робот для військових і рятувальних служб, який дозволяє дистанційно оглядати небезпечні території. Робот має невеликі розміри, працює на базі спеціалізованих безпечних радіоканалів і передає відео в реальному часі.

Рисунок 1.9 – Recon Scout Throwbot 2 [7]

Переваги Recon Scout:

– надійність і витривалість у екстремальних умовах;
– професійна якість передаваного відео;
– оптимізація під спеціальні задачі розвідки.

Недоліки:

– висока вартість (понад 15 000 USD за комплект);
– закритість розробки, неможливість самостійного модифікування.

Таким чином, незважаючи на високий рівень професіоналізму, такі рішення недоступні для масового або бюджетного застосування, особливо в умовах обмежених ресурсів.

1.3 Обґрунтування та вибір підходів щодо виконання завдань КБР

На відміну від Arduino, ESP32-CAM(рис. 1.7) має достатню обчислювальну потужність для базової обробки відео; у порівнянні з Raspberry Pi – значно нижче споживає енергію і має менші габарити; а у співвідношенні ціни та функціоналу – значно випереджає комерційні готові рішення.

Рисунок 1.10 – ESP32-CAM [10]

Мікроконтролер ESP32-CAM об'єднує у собі переваги різних підходів:

– низька вартість (<10 USD за модуль);

– інтегрований WiFi-модуль для бездротової передачі даних;

– вбудована 2-мегапіксельна камера OV2640 з роздільною здатністю до 1600 × 1200;

– підтримка локальної обробки зображення (наприклад, на базі бібліотеки ESP-WHO);

– підтримка збереження даних на microSD-карту;

– можливість енергоефективної роботи завдяки режимам сну.

Вибір ESP32-CAM як базового елемента мобільного комплексу дистанційного спостереження є обґрунтованим як з точки зору технічних можливостей, так і з точки зору економічної доцільності, що особливо важливо в умовах обмеженого фінансування або застосування в польових умовах.

1.4 Формування вимог до апаратно-програмного забезпечення

Для створення прототипу системи дистанційного відеоспостереження на базі мікроконтролера ESP32-CAM необхідно визначити базові вимоги, які забезпечать коректну роботу апаратного та програмного забезпечення, а також відповідність очікуваним функціональним і нефункціональним характеристикам. Врахування цих вимог дозволяє забезпечити стабільність роботи системи, її сумісність з іншими пристроями, ефективність у використанні ресурсів та зручність обслуговування. У процесі формування вимог беруться до уваги технічні обмеження обраного апаратного забезпечення, характеристики середовища експлуатації, а також специфіка задач, які має вирішувати система. Надалі вимоги поділено на апаратні, програмні та нефункціональні.

1.4.1 Апаратні вимоги

Основні апаратні компоненти, які необхідні для реалізації функціоналу мобільної системи відеоспостереження (табл. 1.1).

Таблиця 1.1 – Апаратні вимоги

№	Компонент	Характеристика / Вимога
1	Мікроконтролер	ESP32-CAM, вбудований Wi-Fi, камера OV2640, підтримка microSD, до 1600×1200 px
2	Камера	2 МП, сумісна з ESP32-CAM (OV2640), можливість зйомки відео у режимі реального часу
3	Роботизована платформа	Smart Robot Chassis, 2-4 колеса, підтримка керування через драйвер моторів
4	Драйвер моторів	L298N або аналог, підтримка живлення 5–12 В, керування 2 моторами
5	Джерело живлення	Акумулятор Li-Ion 7.4 В або PowerBank з виходом 5 В через стабілізатор
6	Додаткові компоненти	Перетворювач живлення (наприклад, AMS1117), модуль живлення, дроти, Breadboard тощо
7	Корпус або кріплення	Легка конструкція, можливість знімання для техобслуговування

Запропоновані апаратні компоненти є доступними, компактними та енергоефективними. Це дозволяє створити мобільну систему з автономним живленням і базовою маневреністю при збереженні прийнятної вартості.

1.4.2 Програмні вимоги

Основні програмні вимоги які необхідні для реалізації функціоналу мобільної системи відеоспостереження (табл. 1.2).

Таблиця 1.2 – Програмні вимоги

№	Програмний компонент	Характеристика / Вимога
1	Мова програмування	C/C++ (Arduino IDE) для ESP32-CAM, можливе використання MicroPython
2	Розробницьке середовище	Arduino IDE / PlatformIO / ESP-IDF
3	Бібліотеки	ESP32-CAM, WiFi, WebServer, FS, SD, ESPAsyncWebServer, ESP-WHO (для комп’ютерного зору)
4	Вебінтерфейс	Простий вебсервер з трансляцією MJPEG-потоку, кнопками керування рухом
5	Алгоритм керування рухом	Програмне PWM-керування моторами через L298N (вперед, назад, повороти)
6	Система передачі відео	MJPEG-потік або захоплення зображень з передачею по HTTP
7	Логування / збереження даних	Можливість запису зображень/відео на microSD (за потреби)

Використання Arduino IDE та підтримуваних бібліотек дозволяє спростити процес розробки та налагодження. Програмна архітектура підтримує базові функції вебтрансляції та дистанційного керування, з можливістю подальшого розширення.

1.4.3 Нефункціональні вимоги

Нефункціональні вимоги які впливають на зручність, надійність, масштабованість та експлуатаційні можливості(табл. 1.3).

Таблиця 1.3 – Нефункціональні вимоги

№	Вимога	Опис
1	Надійність	Система повинна стабільно працювати в автономному режимі щонайменше 1–2 години
2	Енергоефективність	Підтримка режиму глибокого сну, оптимізація енергоспоживання
3	Захищеність	Парольний доступ до потоку, обмеження доступу до вебінтерфейсу
4	Простота розгортання	Швидке налаштування через WiFi-точку доступу або вебінтерфейс
5	Вартість	Загальна вартість системи не повинна перевищувати умовний бюджет у 40–60 USD
6	Універсальність	Можливість роботи в приміщенні та на відкритому повітрі (при наявності захисту корпусу)
7	Масштабованість	Можливість додавання нових функцій (датчиків, алгоритмів розпізнавання тощо)

Нефункціональні вимоги забезпечують стабільність, безпечність і зручність використання системи. Урахування цих вимог дозволяє створити гнучку, масштабовану платформу з можливістю подальшого розвитку функціоналу.

Сформульовані вимоги створюють чітке технічне підґрунтя для реалізації прототипу мобільної системи відеоспостереження. Завдяки поєднанню доступних апаратних рішень, перевірених програмних засобів та

належного врахування нефункціональних аспектів проєкт здатен задовольнити базові потреби користувача у віддаленому моніторингу з можливістю подальшого масштабування.

Висновок до розділу 1

Було здійснено всебічний аналіз предметної області, пов’язаної з дистанційним відеоспостереженням на базі мобільних платформ. Основні результати та висновки полягають у наступному:

– визначення об'єкта і предмета дослідження: об'єктом є мобільна система відеоспостереження, а предметом – апаратно-програмне рішення на базі ESP32-CAM для автономного збору та передавання візуальної інформації;

– огляд існуючих рішень: розглянуто широкий спектр комерційних, навчальних та самостійно зібраних платформ, кожна з яких має свої переваги і обмеження в контексті відеоспостереження;

– порівняльний аналіз: встановлено, що більшість існуючих рішень мають або завищену вартість, або надмірну складність реалізації для умов оперативного моніторингу, тоді як ESP32-CAM демонструє оптимальний баланс між ціною, функціональністю та споживанням енергії;

– вибір базової технології: обґрунтовано вибір ESP32-CAM як основного елемента мобільного комплексу завдяки його низькій вартості, підтримці бездротової передачі, вбудованій камері та здатності до обробки зображень;

– формування вимог: визначено ключові апаратні та програмні вимоги до системи, що створює основу для подальшої реалізації прототипу мобільної платформи відеоспостереження.

Таким чином, результати аналізу предметної області дозволяють перейти до етапу проектування, спираючись на обрані технології, що відповідають цілям, поставленим у межах дослідження, а також забезпечують належну ефективність, автономність та економічну доцільність майбутньої системи.

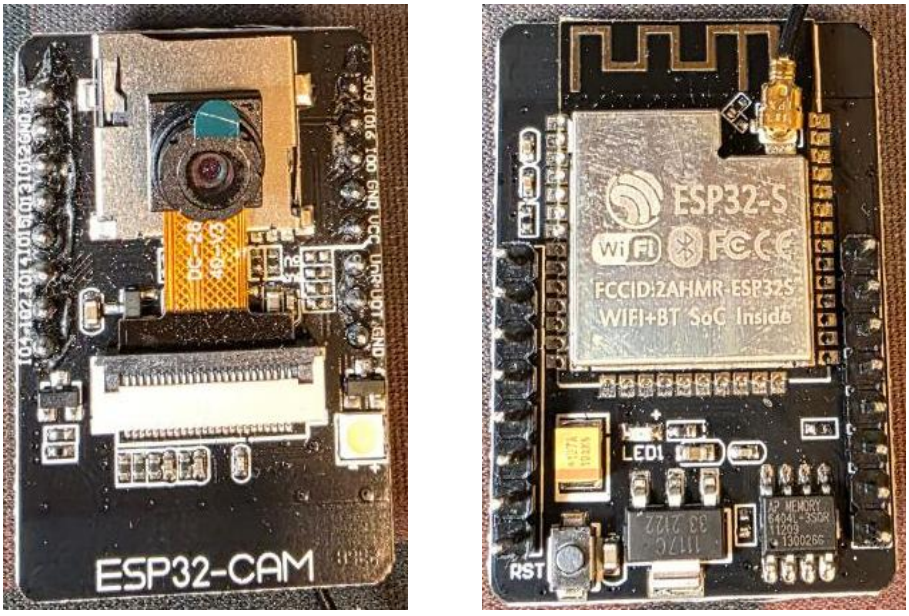
2 ВИМОГИ ТА ВИБІР ОБЛАДНАННЯ ДЛЯ КОМПЛЕКСУ ДИСТАНЦІЙНОГО СПОСТЕРЕЖЕННЯ

2.1 Вибір технологій та компонентів для реалізації АПК

Для побудови АПК дистанційного спостереження було обрано низку компонентів, які забезпечують необхідну функціональність, енергоефективність та доступність у реалізації.

2.1.1 Вибір мікроконтролера ESP32-CAM

Для реалізації системи візуального спостереження з функцією бездротової передачі відео обрано мікроконтролер ESP32-CAM. Цей модуль недорогий і потужний мікроконтролер з підтримкою Wi-Fi та Bluetooth, який поєднує в собі обчислювальні можливості двоядерного процесора та інтегровану камеру OV2640.



а) б)

Рисунок 2.1 – ESP32-CAM (а, б)

Завдяки компактним розмірам, широкій функціональності та низькому енергоспоживанню, цей модуль ідеально підходить для розробки

інтелектуальних IoT-рішень, мобільних роботизованих систем, систем відеоспостереження, моніторингу об’єктів і віддаленого керування.

ESP32-CAM має 16 доступних контактів, з яких частина використовується камерою та внутрішніми підсистемами. Деякі GPIO мають обмеження, зокрема ті, що пов’язані з внутрішнім SPI-з’єднанням PSRAM та камери. Для роботи з периферією (наприклад, драйверами моторів) необхідно уважно підбирати вільні порти. Піни розташовані з двох сторін по 8 штук. Нижче наведено опис пінів, розділений на ліву та праву сторону згідно з фізичним розташуванням на платі (рис. 2.2).

Ліва сторона ESP32-CAM (зліва від антени):

– GPIO2 (HS2_DATA1): цифровий I/O, PWM, управління LED або двигунами;

– GPIO4 (HS2_DATA0): широко використовується, наприклад, для SD-карти або LED;

– GPIO14 (HS2_CLK): PWM, управління драйверами, також використовується в SPI;

– GPIO15 (HS2_CMD): I/O, PWM. Також підходить для підключення моторів;

– GPIO13 (HS2_DATA3): I/O, для підключення сенсорів або моторів;

– GPIO12 (HS2_DATA2): особливий пін: має бути на низькому рівні під час завантаження прошивки;

– GND: земля (мінус живлення);

– 5V: основне живлення модуля (рекомендується подавати саме сюди, а не на 3.3V).

Права сторона ESP32-CAM (справа від антени):

– GND: земля (мінус живлення);

– GPIO1 (U0TXD): передача даних UART, використовується при завантаженні прошивки;

– GPIO3 (U0RXD): прийом даних UART, використовується разом із TX;

– 3.3V / 5V (вивід): вихід живлення, не слід подавати сюди живлення;

– GPIO0: важливий пін, який під’єднується до GND для входу в режим прошивки (BOOT MODE);

– GPIO16: універсальний цифровий пін для керування пристроями;

– 3.3V: вихід живлення (до 500 мА, для живлення дрібних модулів, сенсорів).

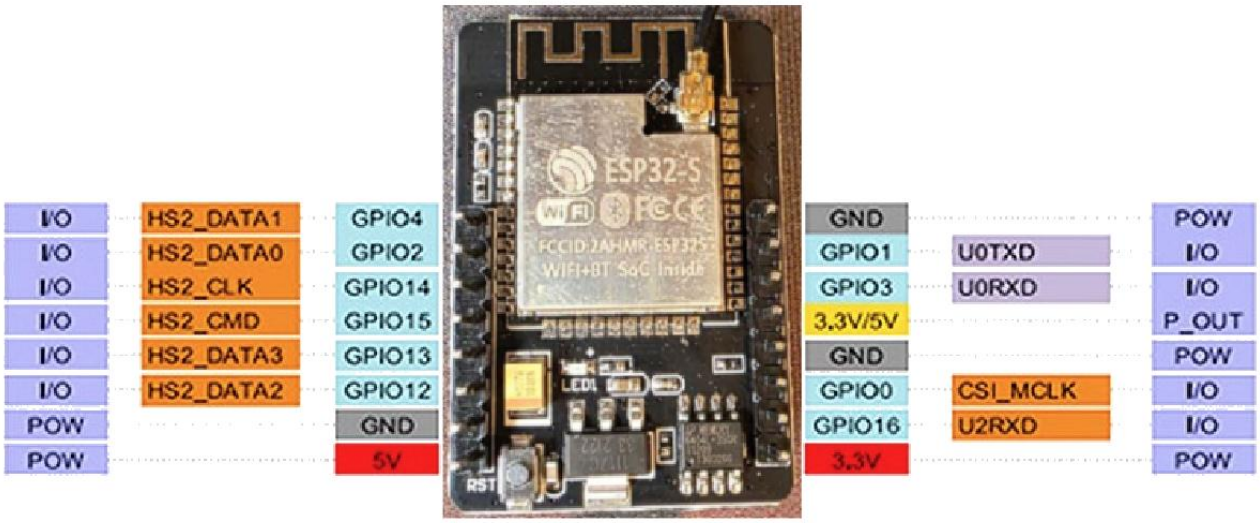


Рисунок 2.2 – Розпіновка ESP32-CAM

ESP32-CAM оснащений двоядерним процесором, що дозволяє одночасно обслуговувати відеозйомку, зв’язок та обробку даних. Це особливо важливо в контексті мобільних роботизованих платформ, де необхідна низька затримка при передаванні зображень у реальному часі.

Таблиця 2.1 – Основні характеристики модуля ESP32-CAM

Характеристика	**Значення**
Процесор	Xtensa® Dual-Core 32-bit LX6 (до 240 МГц)
Пам’ять,КБ	SRAM 520, PSRAM 4096
Камера, Мп	OV2640 2
Максимальна роздільна здатність, пікселів	1600×1200
Wi-Fi, ГГц	IEEE 802.11 b/g/n , 2.4
Слот для карти пам’яті, ГБ	до 4 (FAT32)
Живлення	5 В (через зовнішній стабілізатор або UART)
Контакти	16 з інтерфейсами UART, SPI, I2C, PWM
Підтримка Bluetooth	BLE 4.2 (обмежена у цій моделі)
Розміри, мм	27×40.5
Ціна, грн	~269,00

На етапі вибору розглядалися й інші рішення, зокрема Raspberry Pi Zero W в комплекті з камерою та Arduino Uno у парі з модулем камери OV7670. Кожне з цих рішень має свої переваги та обмеження, пов’язані зі швидкістю обробки, потребами в енергоспоживанні, складністю реалізації та ціною.

Таблиця 2.2 – Порівняння контролерів з підтримкою відео

Характеристика	**ESP32-CAM**	**Raspberry Pi Zero W + камера**	**Arduino Uno + OV7670**
Підтримка Wi-Fi	Так	Так	Ні
Камера в комплекті	Так (OV2640)	Ні (окремо)	Ні (окремо)
Кількість ядер	2	1	1
Потужність процесора, МГц	Висока 240	Середня 1000	Низька 16

Характеристика	**ESP32-CAM**	**Raspberry Pi Zero W + камера**	**Arduino Uno + OV7670**
Об’єм оперативної пам’яті, КБ	520	512000	2
Можливість обробки відео	Обмежена (JPEG, стрімінг)	Висока (можлива ML/AI)	Мінімальна
Інтерфейси вводу/виводу	GPIO, UART, SPI, I2C	GPIO, SPI, I2C, USB	GPIO, UART, SPI, I2C
Розміри	Дуже компактні	Компактні	Великі
Споживання енергії, мА	Низьке (~180–250)	Середнє (~350–500)	Низьке (~50, без камери)
Вартість, грн	~269	~650 (без камери)	~450 (з камерою)
Простота програмування	Висока (Arduino IDE, ESP-IDF)	Висока (Python, Linux)	Середня (Arduino IDE)

ESP32-CAM було обрано через його універсальність, компактність та можливість обробки відеопотоку без потреби у зовнішньому комп’ютері або платі обробки. Завдяки вбудованому WiFi-модуль забезпечує автономну роботу з передачею відео у браузер користувача.

2.1.2 Вибір драйвера моторів

Для реалізації системи керування рухом колісної платформи було обрано драйвер моторів L298N, який базується на H-мості.

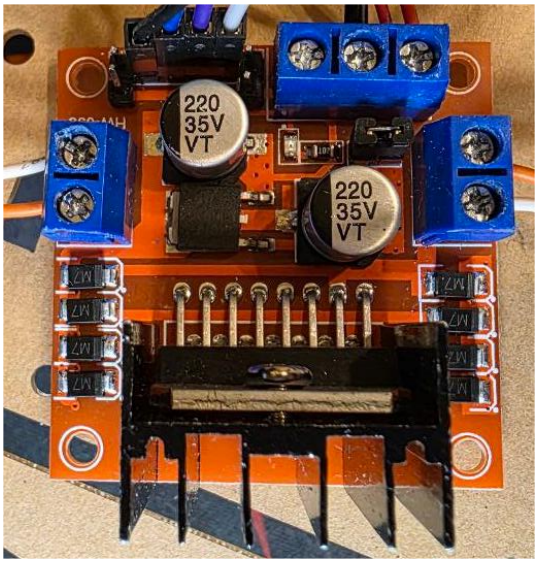


Рисунок 2.3 – Драйвер моторів L298N

Цей модуль широко використовується в проектах із робототехніки завдяки простоті використання та здатності забезпечувати достатню потужність для більшості малогабаритних двигунів постійного струму.

Плата має вбудований стабілізатор напруги на 5 В, що дозволяє живити мікроконтролер або інші компоненти. Її легко інтегрувати з будь-яким мікроконтролером завдяки простому інтерфейсу, і вона є досить надійною для типових застосувань з низьковольтними двигунами 3–6 В.

Плата має зручні гвинтові клеми та pin-інтерфейс для з'єднання з двигунами, джерелами живлення та контролером. Нижче наведено опис основних пінів:

Живлення та керування:

– +12V power: основне живлення для двигунів. Може бути від 5 до 35 В;

– power GND: загальна земля для живлення та логіки;

– +5V power: вихід живлення 5 В (або вхід, якщо використовується без стабілізатора);

– 5V enable (джампер): дозволяє або відключає вбудований стабілізатор 5 В. Якщо джампер встановлений, +5В формується на виході і може живити інші модулі.

Керуючі входи:

– A enable: дозволяє/блокує роботу двигуна A. Для PWM-керування швидкістю слід подавати ШІМ-сигнал;

– input (1, 2): входи для вибору напрямку обертання двигуна A;

– B enable: аналогічно, дозволяє роботу двигуна B;

– input (3, 4): входи для керування напрямком обертання двигуна B.

Виходи до двигунів:

– output A (OUT1, OUT2): з’єднання з мотором A;

– output B (OUT3, OUT4): з’єднання з мотором B.

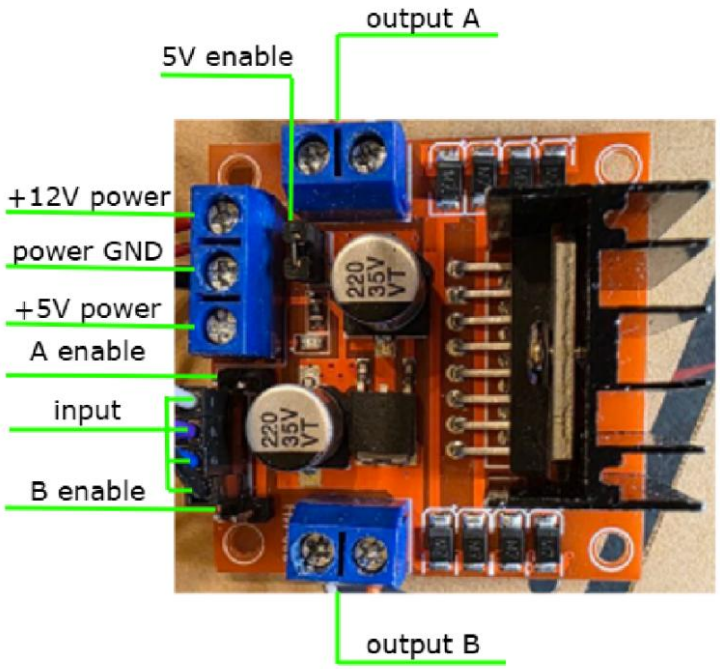


Рисунок 2.4 – Призначення контактів моторів L298N

Завдяки двоканальній схемі H-моста та можливості PWM-контролю швидкості, L298N забезпечує стабільне та ефективне управління навіть у динамічних умовах зміни навантаження.

Таблиця 2.3 – Основні характеристики драйвера L298N

Характеристика	**Значення**
Тип драйвера	Двоканальний H-міст
Напруга живлення мотора, В	5–35
Напруга логіки, В	5
Максимальний струм на канал, А	2
Кількість керованих моторів	2 мотори постійного струму або 1 крокoвий
Захист	Захист від перегріву
Елементи інтерфейсу, В	Вбудований стабілізатор 5
Розміри, мм	43×43×26
Ціна, грн	~89,00

Альтернативами обраного драйверу є L293D – компактний аналог з нижчим струмом, та TB6612FNG – більш сучасний драйвер з покращеною ефективністю. Для обґрунтування вибору було виконано порівняльний аналіз цих рішень.

Таблиця 2.4 – Порівняння драйверів моторів

Характеристика	**L298N**	**L293D**	**TB6612FNG**
Тип драйвера	Двоканальний H-міст	Двоканальний H-міст	Двоканальний H-міст
Максимальний струм на канал, А	2	0.6	1.2
Пікова потужність, Вт	До 25	До 8	До 15
Робоча напруга, В	5–35	4.5–36	2.5–13.5
Втрати енергії	Високі	Середні	Низькі
Ефективність	Низька (~40–50%)	Середня	Висока (~90%)
Розміри	Великі	Середні	Компактні
Вбудований стабілізатор 5В	Так	Ні	Ні
Захист від перегріву	Ні	Частково	Так
Ціна, грн	~89	~45	~110
Простота підключення	Висока	Висока	Середня

Вибір на користь L298N обумовлений його універсальністю, наявністю вбудованого стабілізатора напруги, що дозволяє живити інші компоненти, та широким діапазоном робочих напруг.

L298N забезпечує просте і ефективне управління двигунами, що дозволяє зменшити складність керуючої логіки. Його використання забезпечує стабільну роботу з електродвигунами навіть за умов зміни навантаження чи напруги.

2.1.3 Вибір типу шасі

Механічною основою мобільної системи обрана робо-платформа на три колеса, яка відзначається компактністю, простотою конструкції та достатнім простором для розміщення електронних компонентів. Компактна та легка платформа, яка ідеально підходить для проєктів початкового та середнього рівня складності.

Рисунок 2.5 – Комплект робо-платформи на три колеса

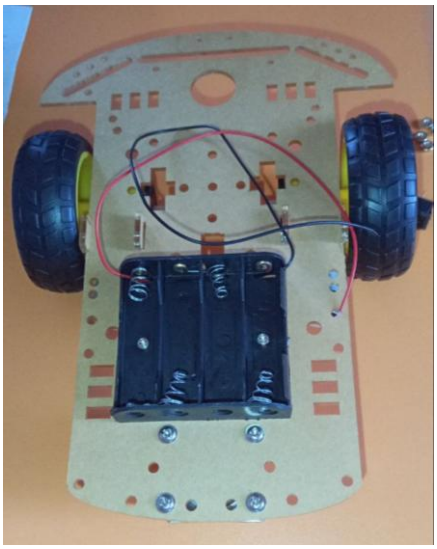


Рисунок 2.6 – Зібраний комплект

Конструкція передбачає два ведучих двигуни та одне обертове колесо, що забезпечує високу маневровість на пласкій поверхні. Мотори мають редуктори зі співвідношенням 48:1, що забезпечує достатній крутний момент для перевезення плати ESP32, драйвера, акумуляторів і додаткових сенсорів.

Таблиця 2.5 – Характеристики робо-платформи

Характеристика	**Значення**
Тип шасі	Однопалубне, триколісне (2 провідних колеса)
Матеріал	Акрил прозорий
Кількість двигунів	2 (з редуктором 48:1)
Напруга живлення, В	3–6
Струм двигунів, мА	100–120
Додатково	Місце для батарей, камера, контролерів
Сумісність	Arduino, ESP32-CAM
Ціна, грн	~177,00

Альтернативами є чотириколісна платформа, яка забезпечує кращу стійкість, та гусенична база, що добре працює на пересіченій місцевості. Проте обидві альтернативи є дорожчими та складнішими в реалізації. З метою прийняття обґрунтованого рішення було проведено порівняння можливих варіантів.

Таблиця 2.6 – Порівняння типів шасі

Характеристика	**3-колісне**	**4-колісне**	**Гусенична платформа**
Маневровість	Висока	Середня	Низька
Складність збірки	Низька	Середня	Висока
Стійкість	Помірна	Висока	Висока
Вартість, грн	~177	~230	~350
Несуча здатність	Середня	Вища	Висока
Поверхня для монтажу	Достатня	Велика	Велика

У результаті, триколісна платформа виявилася вибором для цілей прототипування та реалізації освітнього або дослідницького проєкту, де пріоритетом є простота конструкції та низька вартість.

Ця платформа ідеально підходить для прототипування мобільних роботів. Її невисока вартість, універсальність кріплень та невеликі розміри роблять її доцільним вибором для освітніх, дослідницьких та аматорських проєктів.

2.1.4 Вибір середовища розробки

Розробка програмного забезпечення для апаратного комплексу дистанційного спостереження здійснюється із використанням одного з трьох найпопулярніших середовищ для програмування мікроконтролерів ESP32. У таблиці нижче подано порівняння трьох основних IDE: Arduino IDE, PlatformIO (на базі VS Code) та ESP-IDF (офіційного інструментарію від розробника ESP32 – компанії Espressif).

Arduino IDE [21] є найпростішим і найпопулярнішим середовищем для початку розробки під ESP32-CAM. Його перевагами є низький поріг входу, інтуїтивно зрозумілий інтерфейс, велика база готових бібліотек, зокрема для роботи з камерою, Wi-Fi, серверами, PWM, I2C, UART, та іншими протоколами. Завдяки своїй популярності, IDE має величезну спільноту, де можна знайти готові приклади, документацію та підтримку на форумах. Arduino IDE підтримує підключення ESP32 через розширення плати через менеджер плат, а прошивка виконується через звичайний USB або UART-перехідник.

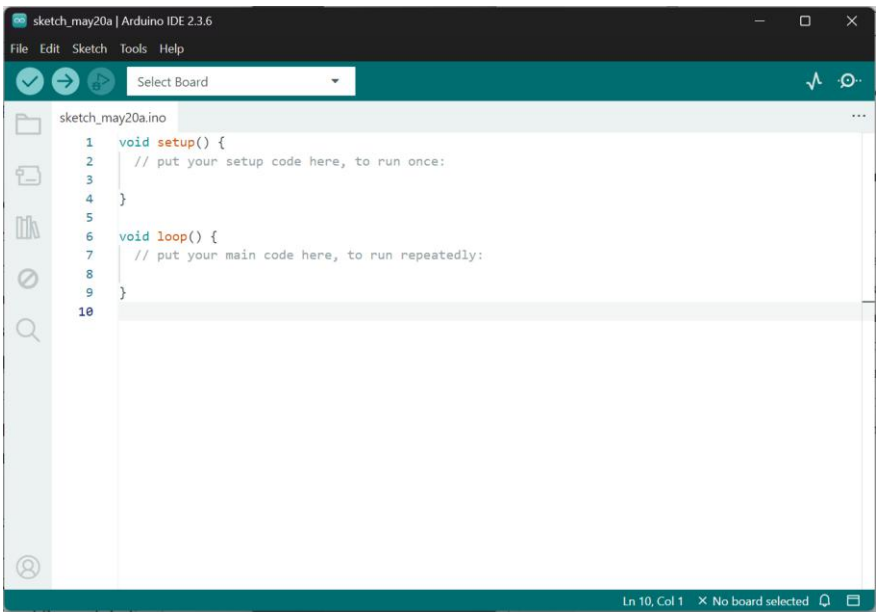


Рисунок 2.7 – GUI Arduino IDE

Це середовище обране для реалізації поточного проєкту через ефективність при базовій роботі, простоту та сумісність із усіма компонентами комплексу.

PlatformIO [20] – це потужне середовище розробки, інтегроване у редактор коду Visual Studio Code. Його перевагами є підтримка сучасного підходу до проєктування – багатофайлова структура, автоматичне управління бібліотеками, вбудована система контролю версій Git, тестування, налагодження тощо. PlatformIO підтримує роботу з ESP32-CAM на рівні з Arduino, але вимагає глибшого розуміння структури проєктів та конфігурацій.

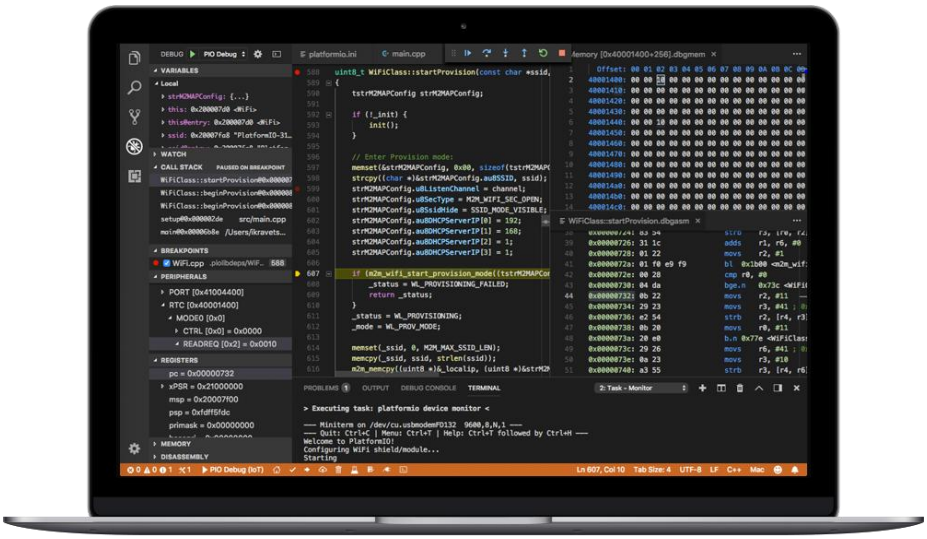


Рисунок 2.8 – GUI PlatformIO [20]

Це середовище є рекомендованим для складніших, комерційних чи довгострокових проєктів. У випадку розширення функціональності комплексу у майбутньому PlatformIO може стати наступним кроком розвитку проєкту.

ESP-IDF [19] – офіційне середовище від розробника ESP32, призначене для роботи на низькому рівні. Воно забезпечує повний контроль над апаратною частиною та є найбільш ефективним за ресурсами. Однак поріг входу до нього досить високий: налаштування відбувається переважно через командний рядок або складну систему конфігурації з великою кількістю параметрів. ESP-IDF дозволяє писати високопродуктивні, енергоефективні програми, але не є доцільним на першому етапі розробки.

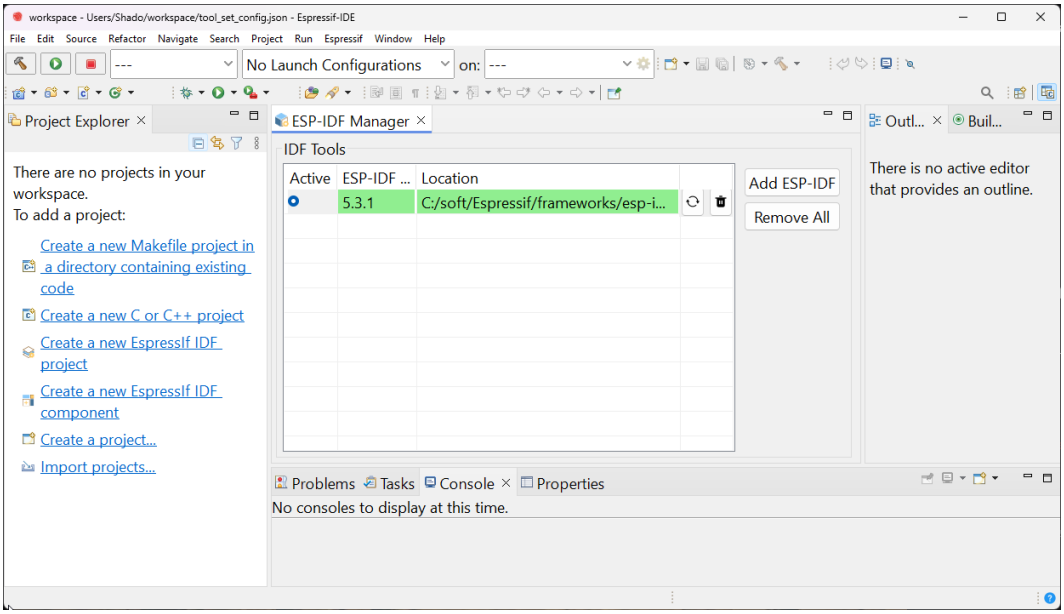


Рисунок 2.9 – GUI ESP-IDF

Це середовище рекомендується лише у випадку створення промислових систем із чіткими вимогами до оптимізації та масштабування.

Таблиця 2.7 – Порівняння середовищ розробки

Параметр	**Arduino IDE**	**PlatformIO (VS Code)**	**ESP-IDF (офіційне SDK)**
Підтримка ESP32-CAM	Повна (через бібліотеки ESP32)	Повна (через esp32dev платформу)	Повна
Мова програмування	C/C++	C/C++, Python, інші	C/C++

Параметр	**Arduino IDE**	**PlatformIO (VS Code)**	**ESP-IDF (офіційне SDK)**
Інтерфейс	Простий, мінімалістичний	Розширений, інтеграція з VS Code	Консоль + розширення VS Code
Вимоги до ресурсів системи	Низькі	Високі (VS Code)	Високі
Наявність бібліотек	Величезна кількість	Величезна кількість	Обмежена, здебільшого низькорівневі
Порог входу	Дуже низький	Середній	Високий
OTA (оновлення по Wi-Fi)	Підтримується	Підтримується	Підтримується
Документація / спільнота	Дуже велика	Велика	Середня (переважно технічна)
Налагодження (debug)	Базове	Повноцінне	Повноцінне

Arduino IDE було обрано як оптимальний варіант для цього проєкту завдяки його простоті, доступності, великій кількості документації і прямій підтримці ESP32. Воно дозволяє швидко розгорнути прошивку, протестувати функціонал та реалізувати складну логіку керування з мінімальним порогом входу. У разі розширення функціональності системи у майбутньому (наприклад, для відлагодження, написання модульних тестів або роботи з RTOS) можливий перехід до PlatformIO або ESP-IDF.

2.2 Концепція та архітектура АПК

Розроблюваний АПК являє собою мобільну платформу з можливістю віддаленого відеоспостереження, що базується на недорогих та доступних апаратних компонентах. Основна мета – створення компактного, автономного пристрою, здатного передавати відео у реальному часі, пересуватися дистанційно за допомогою бездротового керування, а також бути платформою для розширення функціоналу

2.2.1 Компонентна структура АПК

Архітектура побудована на принципі модульності – кожен функціональний блок виконує чітко визначену задачу і може бути замінений або вдосконалений без потреби переробляти всю систему.

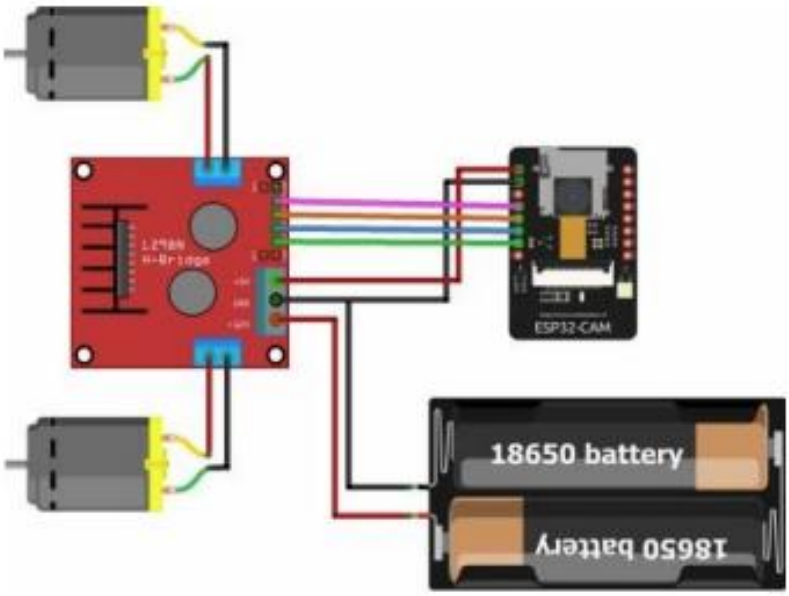


Рисунок 2.10 – Схема підключення компонентів пристрою [23]

Модуль відеоспостереження – мікроконтролер ESP32-CAM з камерою OV2640 виконує функцію відеозахоплення та передачі даних у реальному часі через Wi-Fi. Створює власну точку доступу або підключається до локальної мережі для передачі потокового відео у форматі MJPEG. Реалізує вебсервер для приймання HTTP-запитів: отримання зображення, керування GPIO (рух), зміна налаштувань зйомки.

Модуль керування рухом – використовується двоканальний драйвер L298N, який приймає сигнали з ESP32-CAM через цифрові виходи GPIO. Реалізовано керування напрямком та швидкістю обертання двох моторів. Логіка керування (наприклад: вперед, назад, ліворуч, праворуч, стоп) реалізована через виклики вебкоманд до ESP32 (GET-запити).

Мобільна база – основа трьохколісна платформа (2 провідні колеса та 1 опорне). Два двигуни з редукторами забезпечують достатню тягу для

переміщення по рівній поверхні. Оптимальний розмір платформи дозволяє монтувати камеру, мікроконтролер, батареї, датчики.

Модуль живлення – для живлення використовується батарейний блок 4×AA або дві Li-Ion акумулятори 18650, що забезпечують стабільне живлення для ESP32-CAM та драйвера моторів.При потребі можна додати DC-DC понижуючий стабілізатор для зменшення напруги до 5 В для ESP32.

2.2.2 Апаратна реалізація

Модуль ESP32-CAM виконує як функцію відеозйомки, так і функцію керування рухом, надсилаючи сигнали на драйвер моторів L298N. Уся система живиться від акумуляторного блоку, з можливістю стабілізації живлення до 5 В для чутливих елементів. На рисунку 2.6 зображено блок-схему апаратної частини розроблюваного АПК.

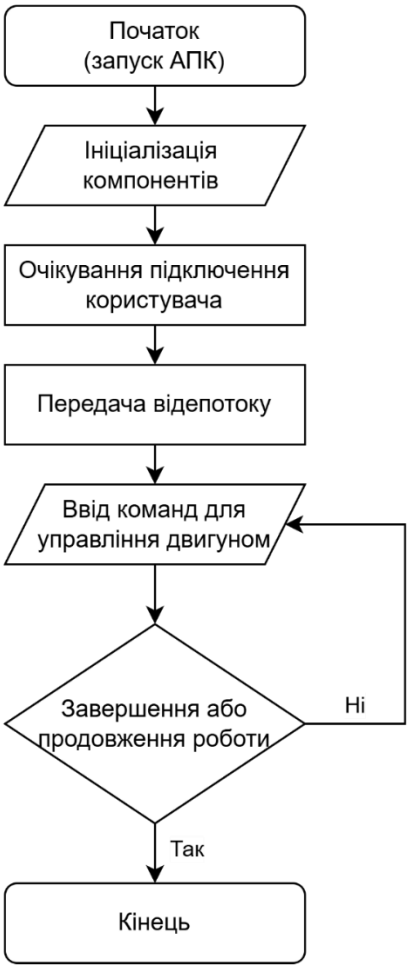


Рисунок 2.11 – Блок-схема АПК

Схема ілюструє взаємодію основних апаратних модулів: мікроконтролера ESP32-CAM, драйвера двигунів L298N, джерела живлення та виконавчих механізмів у вигляді двох електродвигунів. Камера, вбудована в ESP32-CAM, передає відеопотік по Wi-Fi. Вебінтерфейс дозволяє користувачу надсилати керувальні команди, які через HTTP-запити обробляються мікроконтролером. У відповідь ESP32-CAM змінює сигнали на своїх цифрових виходах, що подаються на драйвер L298N. Драйвер, у свою чергу, керує подачею живлення на електродвигуни, змінюючи напрям і швидкість руху мобільної платформи.

2.2.3 Логіка обробки HTTP-запитів на ESP32-CAM

Мікроконтролер ESP32-CAM має вбудований вебсервер, що дозволяє керувати всією системою через браузер. Принцип роботи такий:

1) ініціалізація сервера – після увімкнення ESP32-CAM створює точку доступу або підключається до локального Wi-Fi, та запускає HTTP-сервер;

2) обробка запитів – при переході на головну сторінку IP-адреси ESP32, користувач бачить інтерфейс відеотрансляції. Інтерфейс містить кнопки керування рухом (вперед, назад, ліворуч тощо);

3) управління GPIO – функція обробки запиту змінює рівень сигналу на GPIO-виводах, які з'єднані з драйвером L298N;

4) відповідь клієнту – у відповідь на запит сервер надсилає HTML-код сторінки або коротке повідомлення.

Таким чином, реалізований веб-сервер на ESP32-CAM забезпечує повноцінний інтерфейс для керування платформою в режимі реального часу. Завдяки простій логіці обробки HTTP-запитів, система дозволяє не лише передавати відео у браузер, але й виконувати керуючі дії через зміну стану GPIO-виводів.

2.2.4 Система відеотрансляції

Для організації відеотрансляції у реальному часі в системі використовується формат MJPEG (Motion JPEG). Це один з найпростіших способів реалізації потокового відео на мікроконтролерах з обмеженими ресурсами, таких як ESP32-CAM.

MJPEG – це відеопотік, у якому кожен кадр є окремим зображенням у форматі JPEG. На відміну від більш складних відеокодеків (H.264, MPEG-4), MJPEG не використовує міжкадрову компресію. Кожен кадр обробляється незалежно, що значно спрощує кодування на мікроконтролері.

ESP32-CAM формує MJPEG-потік, який транслюється через HTTP-протокол. Користувач підключається до вебсервера ESP32 через браузер, після чого на вебсторінці відображається потік відео у вигляді оновлюваних JPEG-зображень. Такий підхід не потребує використання зовнішніх бібліотек на клієнтській стороні – лише доступ до Wi-Fi і стандартний браузер.

Затримка залежить від декількох факторів:

– роздільна здатність камери (наприклад, при 320×240 пікселів затримка менша, ніж при 1024×768);

– пропускна здатність WiFi-мережі;

– частота кадрів (зазвичай від 5 до 15 fps);

– навантаження на ESP32, якщо одночасно обробляються вебзапити.

Цей підхід забезпечує сумісність із будь-яким сучасним браузером без потреби в додатковому програмному забезпеченні. Незважаючи на певні обмеження у якості та затримці, реалізація через MJPEG є оптимальним рішенням для задач дистанційного відеоспостереження.

Висновок до розділу 2

У другому розділі було здійснено обґрунтований вибір апаратного та програмного забезпечення для побудови автономної роботизованої

платформи з функцією відеоспостереження. На основі аналізу технічних характеристик, вартості, доступності та відповідності поставленим вимогам було обрано ключові компоненти системи.

Як центральний обчислювальний блок і модуль відеозахоплення обрано ESP32-CAM – компактний і функціональний мікроконтролер з вбудованою камерою та підтримкою Wi-Fi. Його можливості з обробки зображень і передачі потокового відео у форматі MJPEG дозволяють реалізувати віддалене відеоспостереження без використання додаткових плат або комп’ютерів. Розглянута розпіновка модуля дозволила коректно інтегрувати його з іншими компонентами, зокрема драйвером моторів та джерелом живлення.

Для реалізації управління рухом платформи було обрано драйвер L298N, що є перевіреним рішенням для двоканального керування двигунами постійного струму. Його H-мостова архітектура, підтримка струму до 2 А на канал і вбудований стабілізатор напруги забезпечують стабільну та ефективну роботу навіть у змінних умовах навантаження.

Як шасі використано трьохколісну мобільну платформу, яка поєднує простоту конструкції, стабільність руху, а також зручність у встановленні додаткових елементів. Її конфігурація дозволяє підтримувати баланс без складних систем управління або сенсорного зворотного зв’язку.

Архітектура системи побудована за модульним принципом. Кожен компонент виконує чітко визначену функцію, а логіка керування побудована на обробці HTTP GET-запитів через вебсервер ESP32. Це спрощує як розробку, так і можливість розширення функціоналу у майбутньому.

3 РОЗРОБКА АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис апаратної частини системи

Апаратна реалізація АПК виконувалась у декілька логічно послідовних етапів, що дозволило забезпечити надійність, керованість і гнучкість при збиранні та тестуванні системи. Кожен крок складання передбачав встановлення окремого модуля, його механічне закріплення, підключення до інших елементів схеми та обов’язкову перевірку функціональності на даному етапі. Такий покроковий підхід мінімізує ризик помилок, полегшує пошук несправностей, а також дозволяє оперативно виявляти конфлікти живлення, помилки у з’єднаннях або нестабільну роботу компонентів.

3.1.1 Підготовка мобільної бази

На початковому етапі виконується підготовка мобільної платформи – це механічна основа пристрою, яка забезпечує його пересування. Базується вона на трьохколісній схемі: два редукторні двигуни розміщуються симетрично по обидва боки основи. Матеріалом основи може бути акрил, пластик або легкий метал, залежно від вимог до ваги, міцності та гнучкості конструкції. Для кріплення використовуються болти, пластикові затискачі або монтажні пластини.

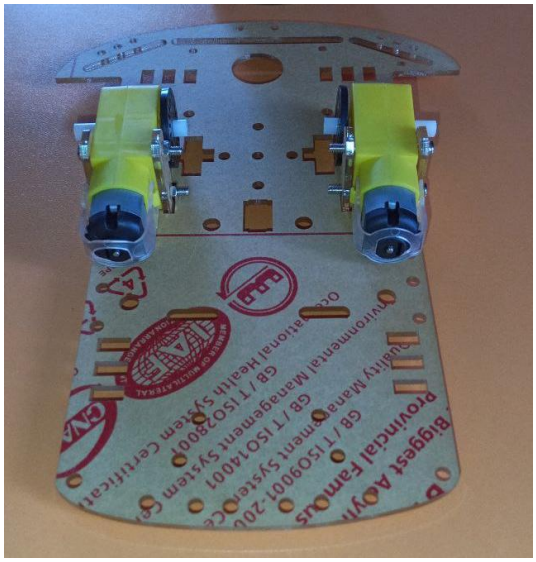


Рисунок 3.1 – Встановлення редукторних двигунів

Після монтажу двигунів переходять до кріплення коліс. Провідні колеса встановлюються безпосередньо на вали редукторів із використанням відповідних кріплень або гвинтів. Важливо забезпечити надійне з'єднання, що унеможливлює прокручування або зміщення під час експлуатації. Заднє вільне колесо, зазвичай кульове або роликове, розміщується на нижній частині шасі та виконує функцію опори для балансування всієї конструкції.

З метою забезпечення стабільності та рівноваги до задньої частини платформи приєднується опорне колесо. Воно сприяє плавному обертанню, маневруванню та пересуванню в усіх напрямках без втрати стійкості. Залежно від конструктивних особливостей, воно може кріпитися як на окрему монтажну пластину, так і безпосередньо до основи.

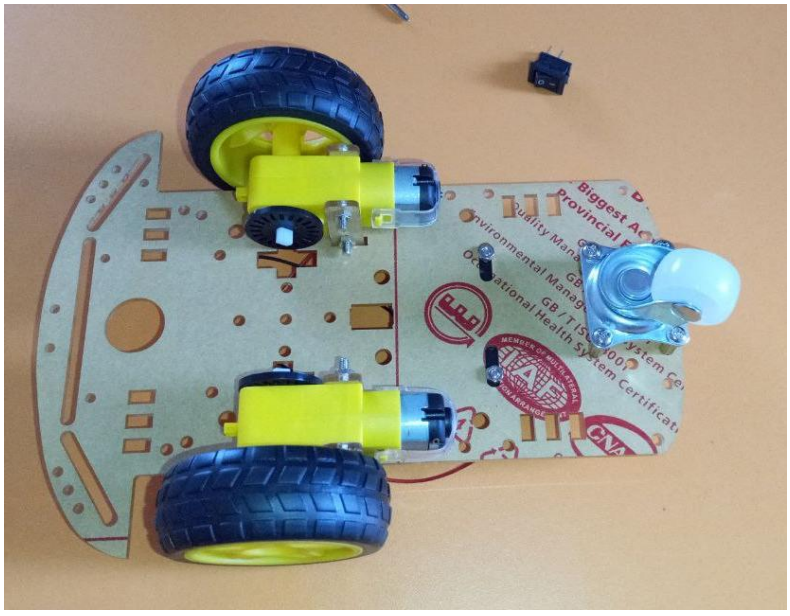


Рисунок 3.2– Встановлення колес

На завершення всі з’єднання перевіряються на надійність: оцінюється щільність кріплення коліс, фіксація двигунів та рівномірність розподілу маси. Готова механічна основа виступає базовою платформою для подальшого встановлення електронних компонентів та джерела живлення.

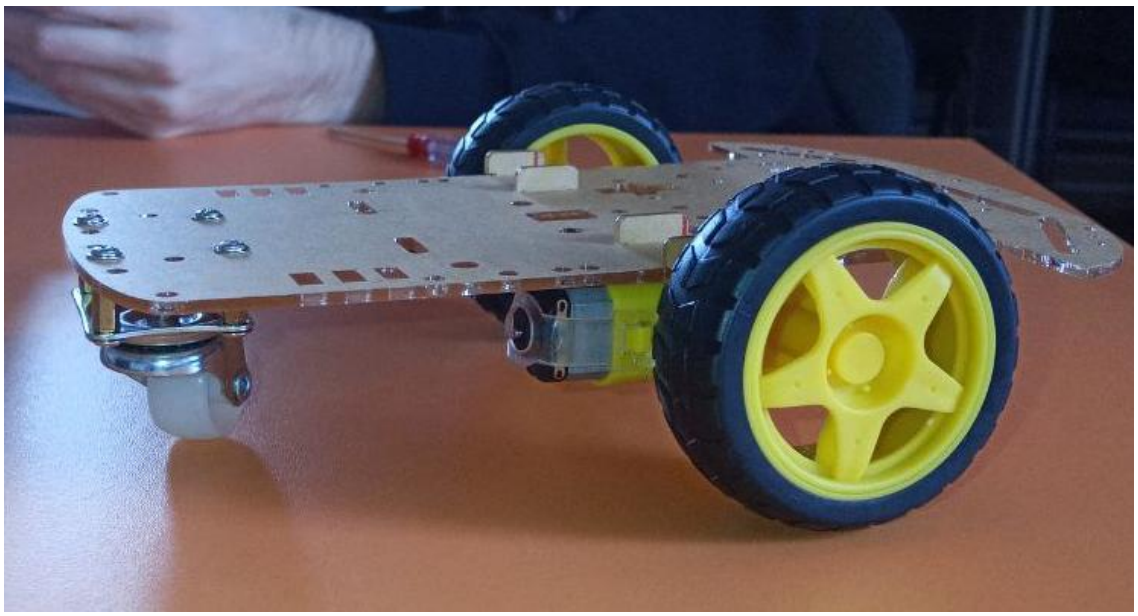


Рисунок 3.3 – Зібрана платформа

Завершена механічна база дозволяє визначити оптимальні точки кріплення для плати ESP32-CAM, драйвера моторів, джерела живлення та допоміжних датчиків. Таким чином, підготовка мобільної платформи створює фундамент для наступних етапів збірки – монтажу електроніки, підключення живлення та налаштування керування.

3.1.2 Монтаж драйвера L298N

Після завершення складання шасі, наступним кроком є інтеграція драйвера L298N – ключового модуля, який керує обертанням двигунів, контролюючи як напрямок, так і швидкість. Через нього плата ESP32-CAM подає логічні сигнали на лінії живлення моторів, забезпечуючи точне керування рухом.

Спершу обирається місце розташування модуля. Враховується зручність доступу до клем для підключення двигунів, джерела живлення та мікроконтролера. Зазвичай драйвер розміщується ближче до задньої або центральної частини основи, щоб мінімізувати довжину з’єднувальних проводів та зменшити втрати напруги.

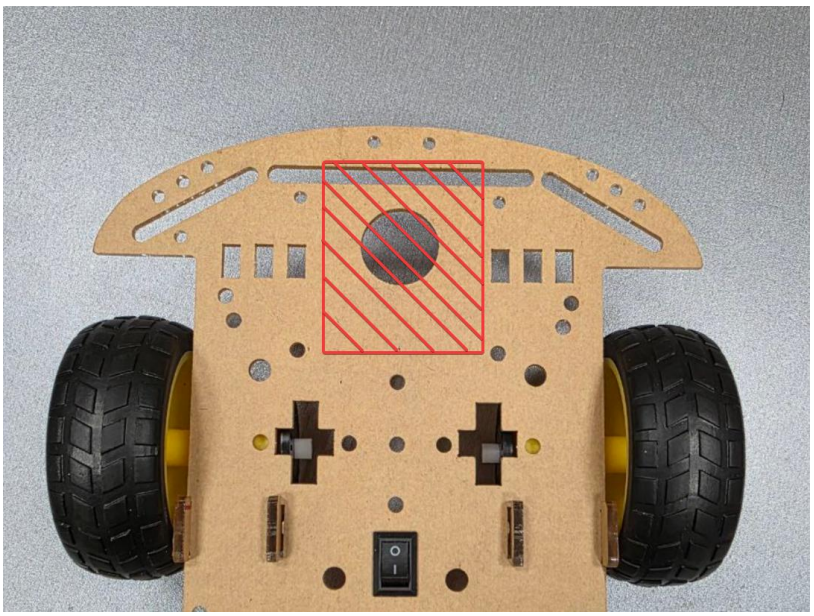


Рисунок 3.4 – Вибір місця для встановлення драйвера L298N

Після вибору розташування модуль фіксується. Якщо конструкція дозволяє, використовуються монтажні отвори на платі L298N із кріпленням через гвинти та гайки. Якщо така можливість відсутня, застосовують пластикові стяжки або термоізоляційний двосторонній скотч.

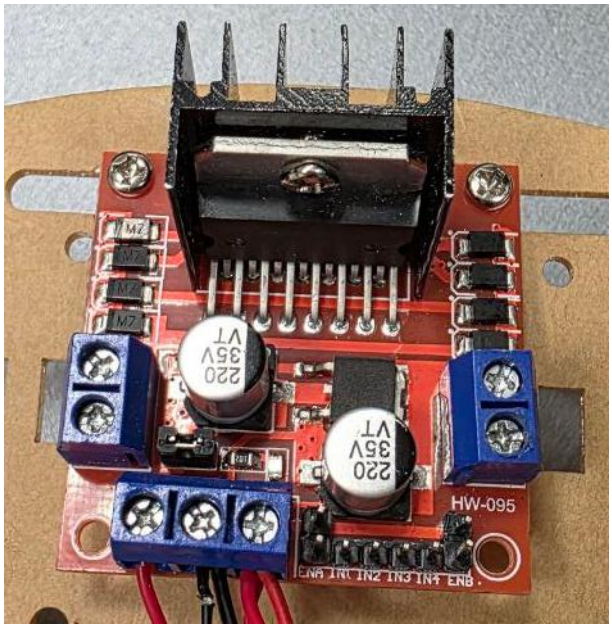


Рисунок 3.5 – Механічне кріплення модуля L298N на основі платформи

Після встановлення драйвера виконується підключення живлення: до контактів 5V та GND під’єднується джерело живлення. Важливо зазначити, що між контактами +12V та 5V необхідно встановити перемичку (джампер),

оскільки при живленні драйвера тільки через +12V без цієї перемички логічна частина L298N не активується і модуль не буде реагувати на керуючі сигнали. Така конфігурація дозволяє одночасно живити як силову, так і логічну частини модуля з одного джерела.

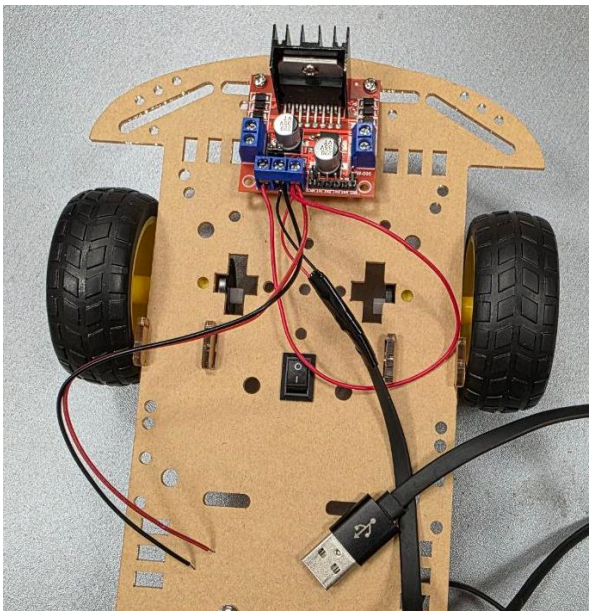


Рисунок 3.6 – Підключення живлення до модуля L298N

На виходи OUT1–OUT4 під’єднуються дроти, що ведуть до обох редукторних двигунів. Обов’язково перевіряється правильність полярності та надійність з’єднань, щоб уникнути некоректного напрямку обертання або перегріву.

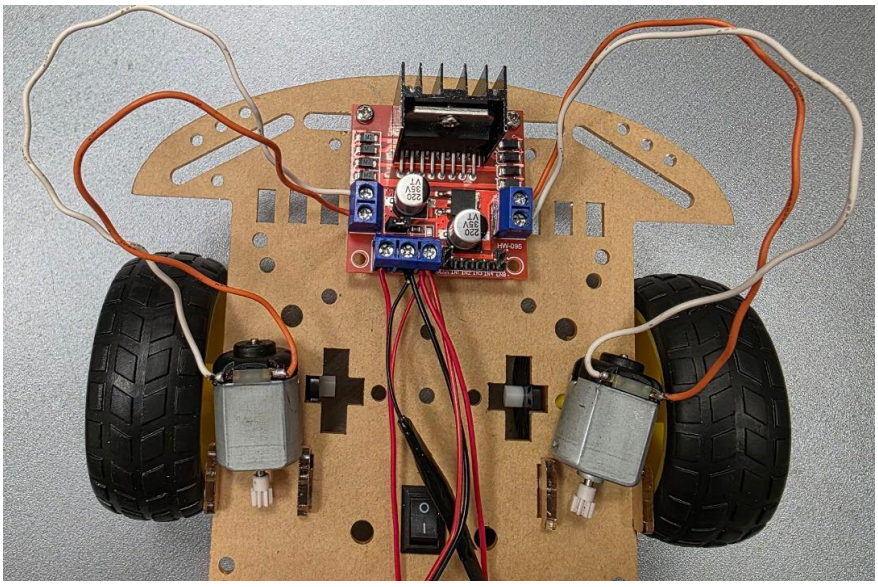


Рисунок 3.7 – Підключення дротів для керування двигунів

Останнім кроком є підключення керуючих сигналів від ESP32-CAM. До входів IN1–IN4 та ENA/ENB (через джампери або ШІМ-виводи ESP32) під’єднуються відповідні GPIO, які будуть відповідальні за команди руху. На цьому етапі також перевіряється наявність на драйвері перемички 5V–EN, яка дозволяє використовувати вбудований стабілізатор живлення для подачі 5 В на логічну частину.

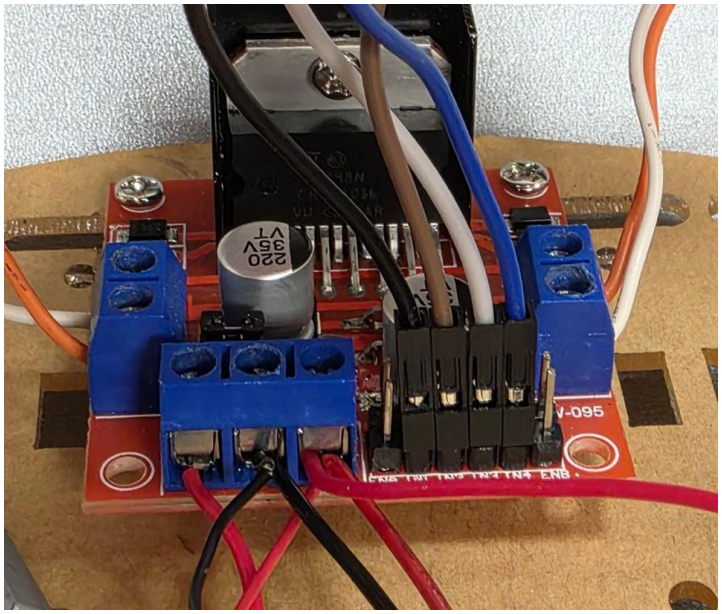


Рисунок 3.8 – Підключення керуючих сигналів до драйвера L298N

На цьому етапі лише виконується механічне закріплення драйвера та електричне підключення його виходів до моторів і входів до ESP32-CAM. Живлення до L298N ще не подається. Важливо переконатися, що всі з’єднання зроблені правильно, відповідно до схеми, але перевірка працездатності системи виконується лише після етапу підключення джерела живлення.

3.1.3 Установка модуля ESP32-CAM

Після монтажу драйвера L298N переходять до встановлення модуля ESP32-CAM, який виконує функції відеозахоплення, передавання потоку MJPEG і дистанційного керування рухом. Він створює WiFi-точку доступу або під’єднується до мережі, приймаючи HTTP-запити для керування платформою.

Спочатку визначається оптимальне місце встановлення модуля. Основні вимоги – відкритий огляд для камери, захист від механічного впливу та зручність підключення живлення і GPIO-виводів. Найчастіше модуль розташовується у передній верхній частині конструкції.

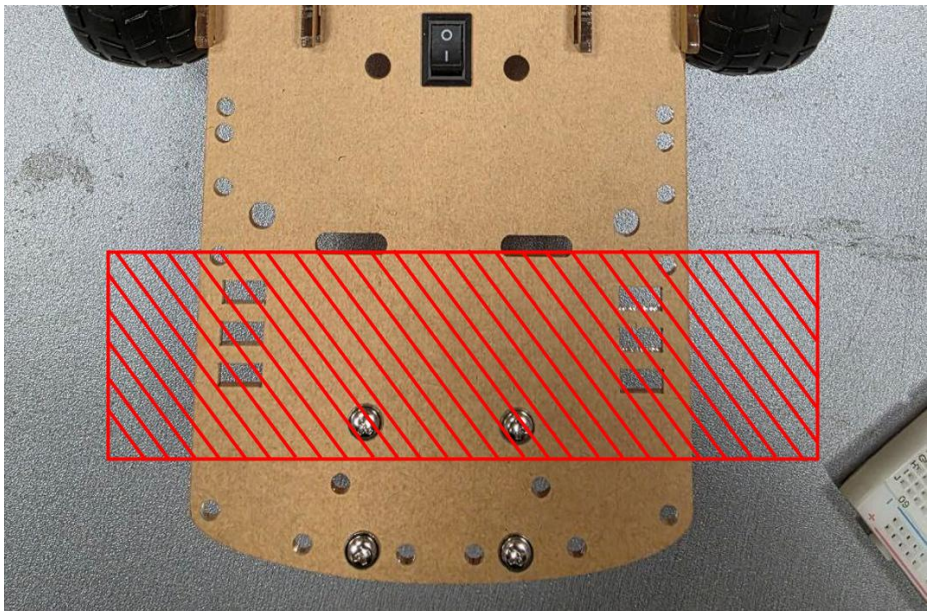


Рисунок 3.9 – Вибір місця для встановлення ESP32-CAM

Після здійснюється підключення живлення до модуля. Живлення модуля подається від вбудованого стабілізатора драйвера L298N, який має вихід 5V.

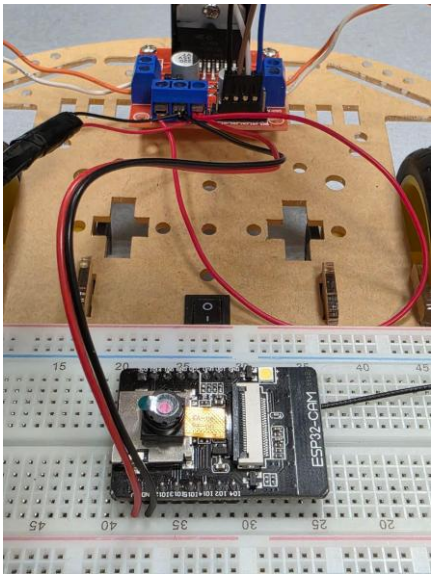


Рисунок 3.10 – Підключення ESP32-CAM до виходу 5V драйвера L298N

Це дозволяє уникнути використання окремого DC-DC перетворювача та спрощує загальну схему підключень. Вихід 5V на платі драйвера забезпечує стабільну напругу, необхідну для роботи мікроконтролера.

Живлення підключається до відповідних контактів 5V та GND на ESP32-CAM. Важливо дотримуватися полярності та переконатися у наявності стабільної напруги на виході драйвера перед підключенням, оскільки перевищення допустимого рівня напруги (понад 6 В на вході драйвера) може призвести до перенавантаження стабілізатора або пошкодження ESP32-CAM.

Наступним кроком є з’єднання GPIO-виводів ESP32-CAM із входами IN1–IN4 на драйвері L298N. Залежно від логіки керування, можна використовувати 4 цифрові виходи ESP32. Всі з'єднання виконуються за допомогою м’яких проводів або спеціальних шлейфів з фіксаторами, щоб забезпечити надійність під час руху платформи.

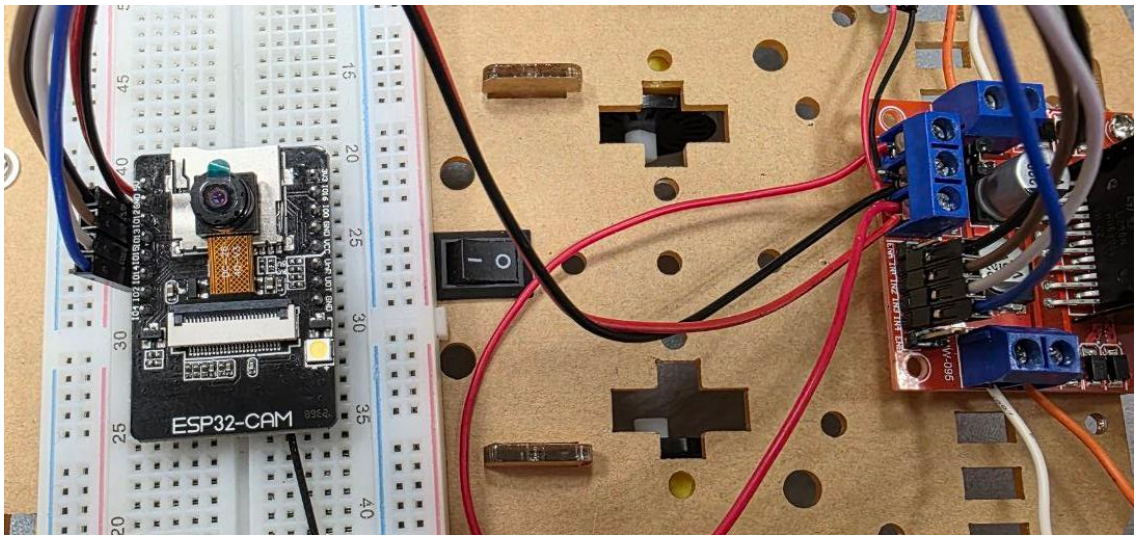


Рисунок 3.11 – Підключення ESP32-CAM до драйвера моторів L298N

Перевірка працездатності схеми виконується після підключення джерела живлення, що буде описано в наступному підрозділі. Це дозволяє уникнути помилкової оцінки роботи у разі нестабільної подачі струму або неповного підключення.

3.1.4 Підключення джерела живлення

Для автономної роботи системи використовується портативний повербанк ємністю 10000 mAh, який забезпечує стабільне живлення через USB-вихід з параметрами 5V/2A. Це дозволяє підтримувати тривалу роботу апаратного комплексу без необхідності частого підзаряджання. Обраний тип живлення є зручним і доступним рішенням для мобільних платформ, оскільки не потребує складної елементної бази чи окремого зарядного блоку.

Повербанк під’єднується через стандартний USB-кабель до вхідного роз’єму драйвера L298N, який підтримує подачу живлення з напругою 5 В. Таким чином, одне джерело живлення забезпечує електроживлення як самого драйвера, так і підключеного до нього модуля ESP32-CAM, який отримує напругу через вихід 5V драйвера.

Джерело живлення розміщується поблизу центру мобільної платформи. Таке розташування обране не лише для збереження балансу конструкції під час руху, але й виконує додаткову функцію – забезпечує легкий нахил корпусу платформи, що орієнтує ESP32-CAM під невеликим кутом до поверхні.

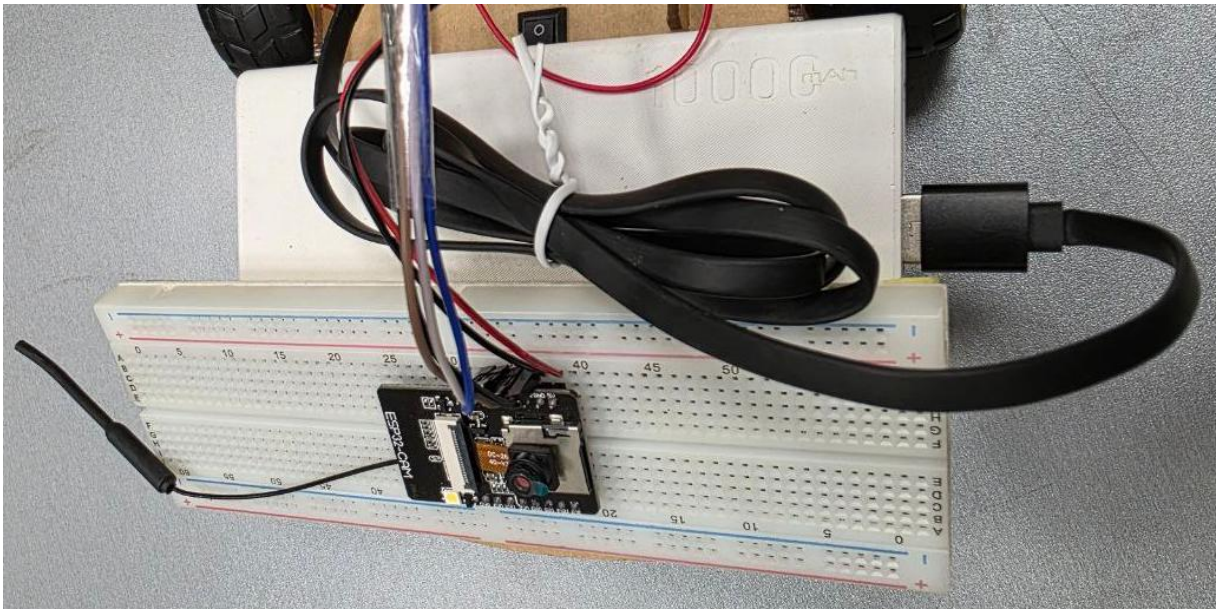


Рисунок 3.12 – Розміщення джерела живлення на платформі

Це створює оптимальні умови для тестового положення камери, зручного для спостереження за об’єктами, розташованими на рівні підлоги або

попереду рухомого пристрою. Для фіксації повербанка використовуються двосторонній скотч або монтажні стяжки, які забезпечують надійність кріплення й водночас дозволяють легко демонтувати джерело живлення при необхідності.

Після остаточного підключення живлення виконується перевірка напруги на ключових вузлах системи. За допомогою мультиметра контролюється подача 5V на вхід драйвера L298N, а також наявність стабільного живлення на виводі 5V, до якого підключено ESP32-CAM. У разі правильної подачі живлення індикатор ESP32-CAM починає світитися або мигати, що сигналізує про запуск модуля. Аналогічно перевіряється живлення обмоток двигунів через драйвер. Якщо всі параметри в нормі – система готова до початкового запуску та подальшої програмної конфігурації.

У процесі підключення важливо дотримуватись послідовності: спочатку з’єднати всі елементи, а потім під’єднати повербанк. Після підключення, система автоматично запускається – індикатори на ESP32-CAM та L298N сигналізують про подачу живлення, після чого модуль починає виконання прошитої програми.

3.2 Опис програмної частини системи

3.2.1 Встановлення та налаштування середовища

Для програмної реалізації модуля ESP32-CAM використовується середовище розробки Arduino IDE. Першим кроком є завантаження останньої стабільної версії Arduino IDE з офіційного сайту [24]. Після завантаження інсталяційного пакету його потрібно встановити, дотримуючись стандартних вказівок майстра встановлення.

Рисунок 3.13 – Завантаження Arduino IDE з офіційного сайту

Оскільки плати на базі ESP32 не входять до стандартного набору плат, попередньо вбудованих у Arduino IDE, необхідно вручну додати репозиторій плат від компанії Espressif Systems – розробника мікроконтролера ESP32. Це дозволяє завантажити та встановити всі необхідні налаштування, бібліотеки та інструменти компіляції для повноцінної роботи з платою ESP32-CAM.

Для цього у вікні Arduino IDE необхідно відкрити меню Boards Manager яке знаходиться у лівій колонці. Після цього з’явиться бокова панель пошуку, де можна переглянути або встановити доступні пакети платформ для різних типів мікроконтролерів. У полі пошуку вводимо запит «esp32» – серед знайдених результатів з’явиться «esp32 by Espressif Systems».

Після натискання на Install, Arduino IDE автоматично завантажує всі необхідні компоненти, включаючи конфігураційні файли, компілятор, бібліотеки та приклади для всіх підтримуваних плат на базі ESP32 (у тому числі й ESP32-CAM). Процес може тривати кілька хвилин, залежно від швидкості інтернет-з’єднання. Після завершення встановлення кнопка зміниться на Remove, що означає успішне завершення процесу.

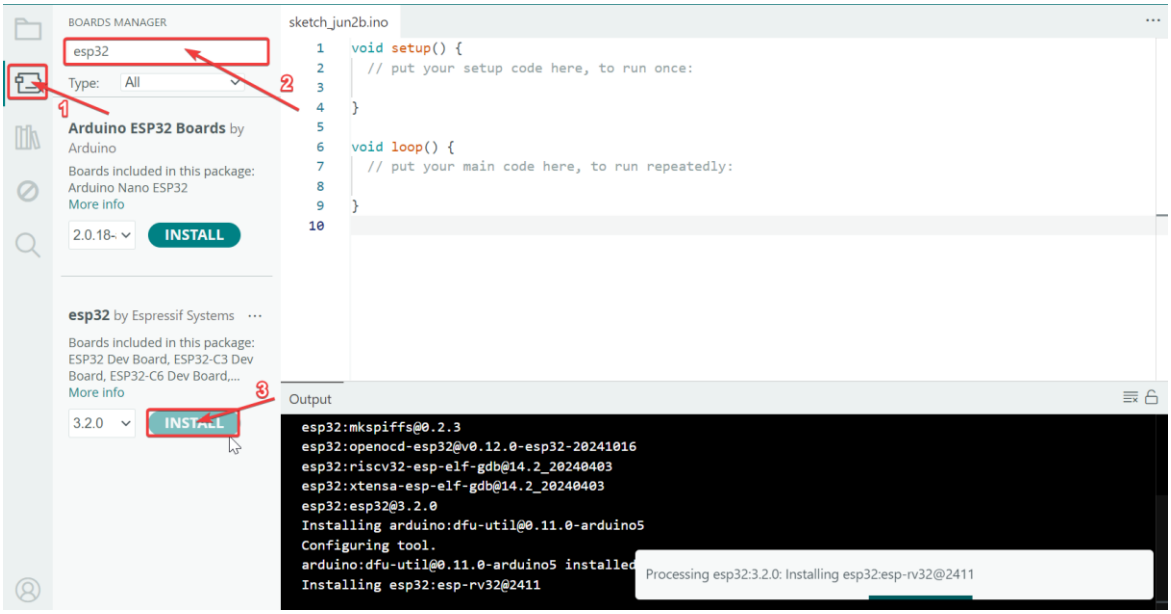


Рисунок 3.14 – Завантаження ESP32 by Espressif Systems

Після завершення встановлення пакету esp32 by Espressif Systems, у меню вибору плат з’являється повний перелік доступних моделей ESP32. Для роботи з ESP32-CAM необхідно перейти до меню Tools далі обрати Board та зі списку обрати плату AI Thinker ESP32-CAM – саме ця конфігурація відповідає апаратній реалізації модуля, що використовується в даному проєкті.

Після вибору плати потрібно також вказати правильний порт, через який здійснюється з’єднання з мікроконтролером. Для цього в тому ж меню слід обрати відповідний COM-порт, який з’являється після підключення плати до комп’ютера через USB-UART перехідник.

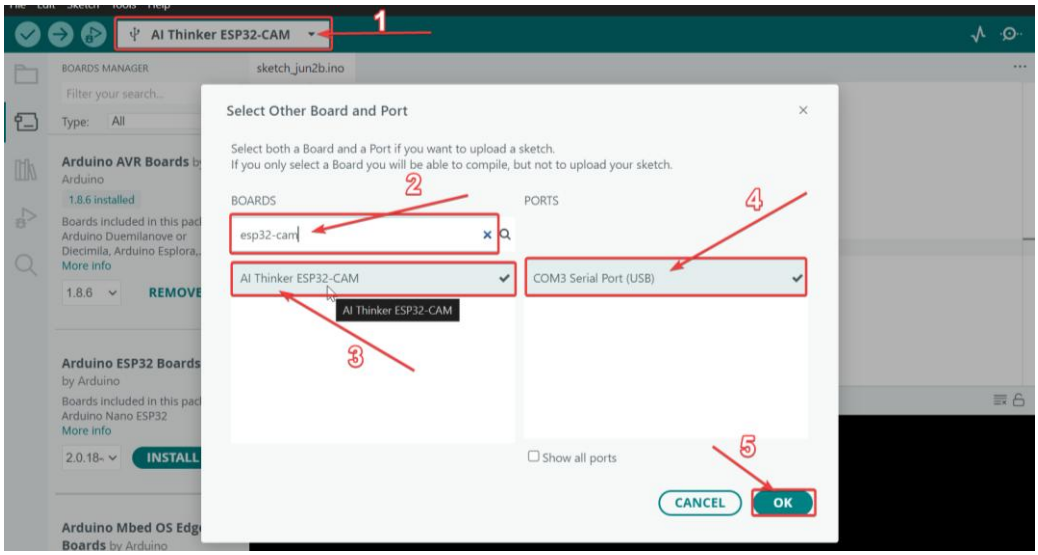


Рисунок 3.15 – Вибір плати AI Thinker ESP32-CAM та порту у Arduino IDE

У більшості випадків Arduino IDE автоматично розпізнає активний порт, проте якщо в системі підключено декілька пристроїв, слід переконатися, що вибраний саме той, що відповідає підключеній платі. Коректне встановлення плати та порту є критично важливим для подальшого завантаження прошивки, компіляції коду та обміну даними між комп’ютером і мікроконтролером.

3.2.2 Конфігурація мікросхеми ESP32-CAM

Після налаштування середовища Arduino IDE можна переходити до конфігурації мікросхеми ESP32-CAM. У цьому проєкті використовується модуль AI-Thinker ESP32-CAM з вбудованою камерою та можливістю бездротового зв’язку через Wi-Fi. Також підключено два двигуни, керування якими реалізовано через вебінтерфейс.

Першим етапом після завантаження мікроконтролера є підключення до WiFi-мережі. У коді задаються SSID та пароль, які використовуються для підключення до існуючої мережі. Якщо підключення успішне, пристрій отримує локальну IP-адресу, яку виводить у монітор порту. За цією адресою користувач може звернутися до пристрою через веббраузер.

```
const char* ssid = "Sw";
const char* password = "00000000";
```

(а)

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

Serial.print("Camera Stream Ready! Go to: http://");
Serial.println(WiFi.localIP());
```

(б)

Рисунок 3.16 – Налаштування підключення: а – мережеві параметри; б – підключення до WiFi-мережі

Альтернативно, ESP32-CAM може бути налаштована у режимі точки доступу, що дозволяє створити власну бездротову підмережу без необхідності підключення до зовнішнього маршрутизатора. Це зручно у польових умовах або для автономної роботи пристрою.

Після встановлення з’єднання з мережею відбувається ініціалізація камери. Цей процес включає в себе створення об’єкта конфігурації типу *camera_config_t*, у якому зазначається використання відповідних пінів, формат кадру, роздільна здатність та інші параметри. Потім ця конфігурація передається до функції *esp_camera_init()*, яка запускає модуль камери. Якщо ініціалізація пройшла успішно, камера готова передавати зображення.

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if(psramFound()){
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}
```

а)

```
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
  Serial.printf("Camera init failed with error 0x%x", err);
  return;
}
```

б)

Рисунок 3.17 – Підключення камери: а – конфігурація камери;
б – ініціалізація камери

Одночасно із запуском камери відбувається запуск вебсервера, що забезпечує взаємодію з користувачем. Основна частина сервера працює на HTTP-порті 80, і забезпечує доступ до HTML-інтерфейсу з кнопками керування. Крім того, відкривається окремий обробник MJPEG-потоку, який працює у нескінченному циклі, захоплюючи кадри з камери, перетворюючи їх на JPEG-формат і надсилаючи клієнту по протоколу HTTP у вигляді безперервного потоку. Це дозволяє переглядати відео у браузері без затримок.

```
void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
  };

  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 1;
  config.ctrl_port += 1;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}
```

Рисунок 3.18 – Запуск вебсервера

HTML-сторінка, що передається клієнту при зверненні до сервера, містить простий інтерфейс з кнопками керування: «Вперед», «Назад», «Вліво», «Вправо» та «Стоп». Натискання на будь-яку з них спричиняє надсилання HTTP-запиту з відповідним параметром (наприклад, /action?go=forward). Цей запит потрапляє до окремого обробника на сервері, який зчитує параметр і відповідно вмикає або вимикає GPIO-виходи, що підключені до H-моста або драйвера моторів. Відповідно, змінюється полярність на двигунах, і платформа рухається у заданому напрямку. Після завершення написання коду, починається процес прошивки.

Процес прошивки ESP32-CAM дещо відрізняється від класичних плат Arduino, оскільки модуль не має вбудованого USB-інтерфейсу для прямого підключення до комп’ютера. Для цього було використано USB-UART перетворювач. Це спеціальний пристрій, який дозволяє комп’ютеру передавати дані через інтерфейс USB до UART-портів мікроконтролера. Оскільки ESP32-CAM не має вбудованого USB-інтерфейсу для програмування, використання такого перетворювача є обов’язковим етапом.

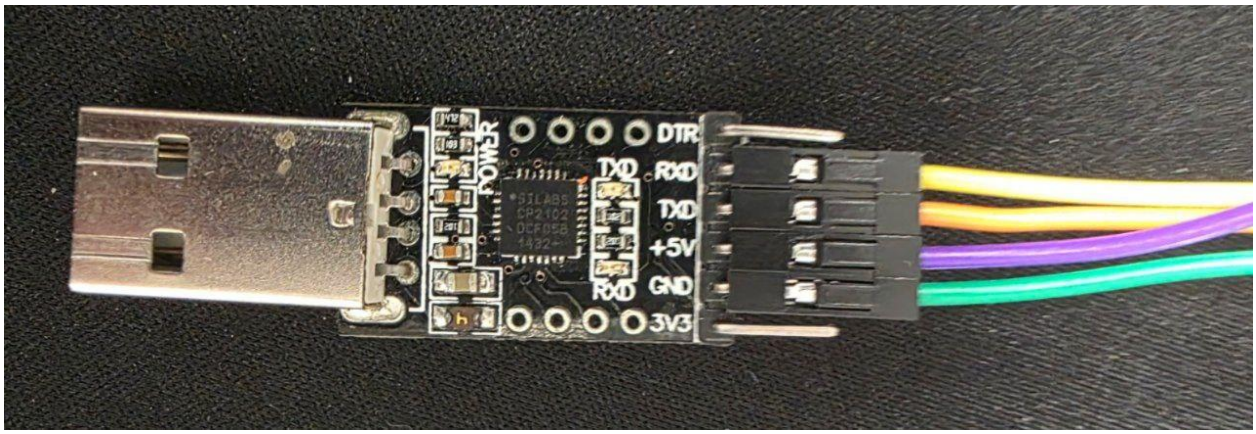


Рисунок 3.19 – USB-UART

Перед початком прошивки потрібно правильно з’єднати ESP32-CAM з USB-UART адаптером:

– GND (UART) до GND (ESP32-CAM);
– 5V (UART) до 5V (ESP32-CAM);
– TX (UART) до U0R (RX) ESP32-CAM;

– RX (UART) до U0T (TX) ESP32-CAM;

– додатково: IO0 слід з’єднати з GND, щоб перевести плату в режим завантаження (boot mode).

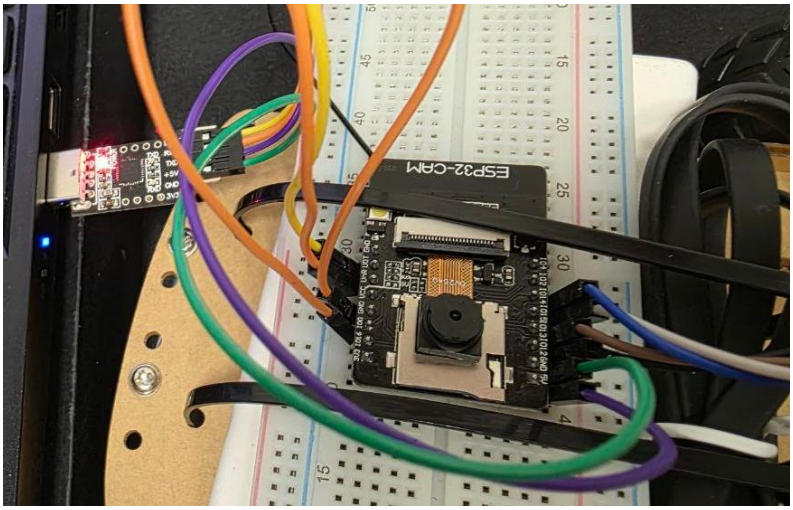


Рисунок 3.20 – Підключення USB-UART до ESP32-CAM

Після фізичного підключення ESP32-CAM до USB-UART в Arduino IDE, у якому вже має бути встановлена підтримка плат ESP32 зверху зліва натискаємо на кнопку Upload, воно робить перевірку написаного коду. Якщо код виконується без помилок, програма записується на ESP32-CAM, в іншому випадку буде виведена помилка.

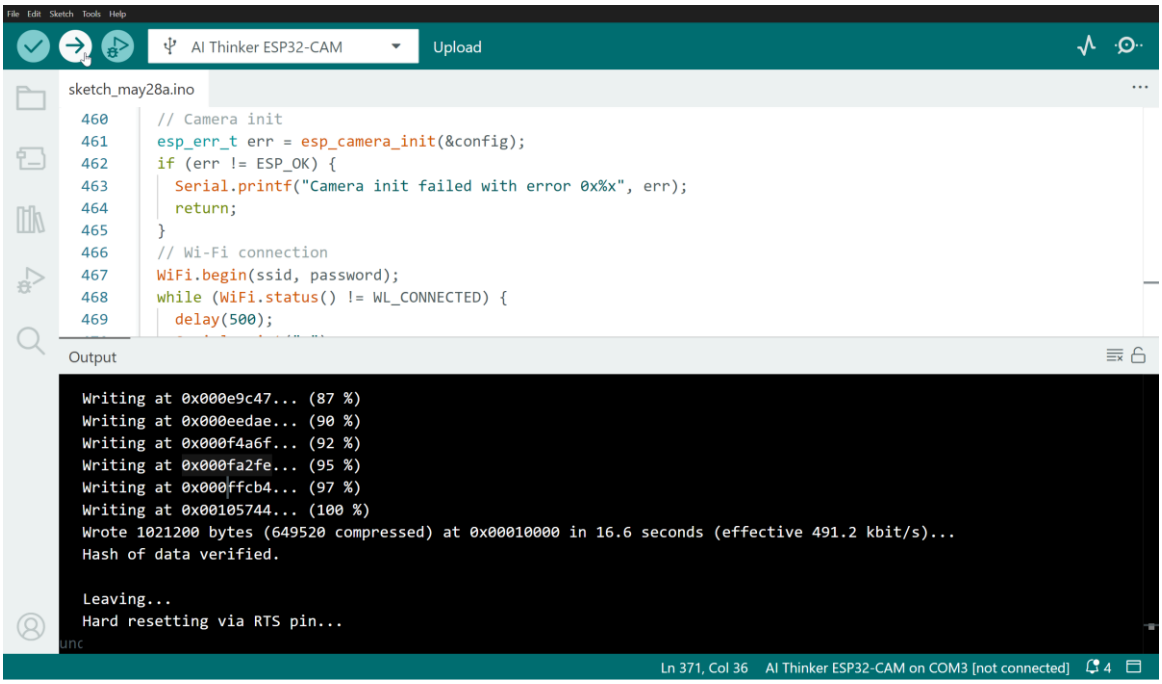


Рисунок 3.21 – Завантаження прошивки на ESP32-CAM

Після успішного завантаження з'явиться повідомлення «Hash of data verified», яке підтверджує, що прошивка записана в пам’ять контролера. Щоб запустити програму, ESP32-CAM потрібно перезавантажити. Перед цим важливо розімкнути контакт між IO0 та GND, адже інакше мікроконтролер знову завантажиться в режим очікування прошивки. Після натискання Reset плата почне виконання користувацького коду – саме того, який було щойно прошито. Це означає, що вона автоматично підключиться до Wi-Fi, запустить сервер і почне передавати зображення з камери, а також реагуватиме на сигнали керування.

3.3 Тестування АПК

Після завершення етапів проєктування апаратної частини та розробки програмного забезпечення, було проведено повне тестування АПК в умовах, максимально наближених до реальних. Основною метою тестування було перевірити працездатність усіх функціональних модулів системи: передусім стійкість з’єднання з мережею, коректну роботу сервера, якість відеопотоку, а також точність та оперативність реагування на команди керування.

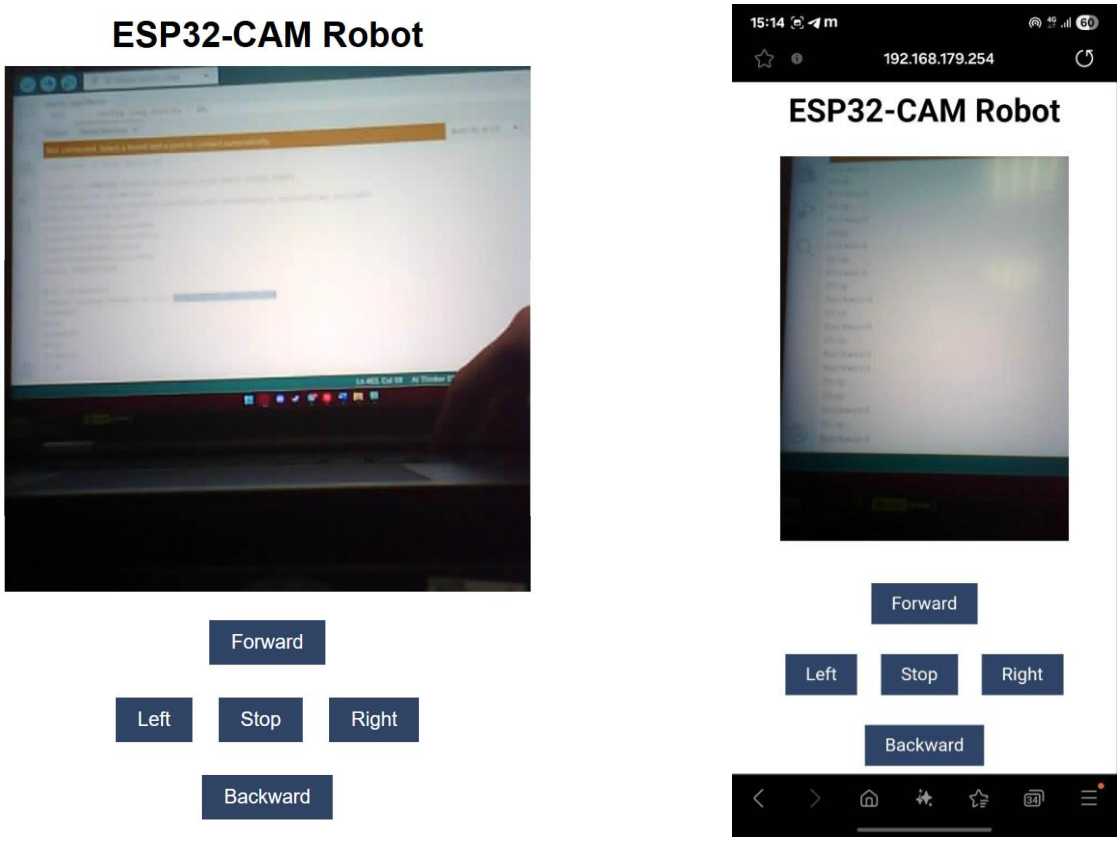
Процедура тестування розпочалася з підключення живлення до ESP32-CAM та перевірки, чи модуль успішно підключається до WiFi-мережі. Вивід на серійному моніторі підтверджував призначення IP-адреси, за якою клієнт міг звернутися до сервера. У випадку, коли не вдається підключитися до зовнішньої мережі, плата автоматично активувала режим точки доступу, що дозволяло під’єднатися до неї напряму зі смартфона або ноутбука.

```
...
WiFi connected
Camera Stream Ready! Go to: http://192.168.179.254
```

Рисунок 3.22 – Вивід IP-адреси

Після введення IP-адреси в адресний рядок браузера на пристрої користувача, автоматично відкривався вебінтерфейс. У ньому відображалося

зображення з камери, яке оновлювалося в реальному часі. Це підтверджувало коректну реалізацію MJPEG-стрімінгу та стабільну передачу даних по HTTP-з’єднанню.



а) б)

Рисунок 3.23 – Перевірка роботи вебсервера: а – підключення з ноутбука; б – підключення з телефона

Після цього відбувалося випробування функціональності керування двигунами. Через вебінтерфейс користувач надсилав команди на зміну напрямку руху або повороту. За допомогою серійного монітору також фіксувалася інформація про те, які саме команди надходили на плату, що дозволяло зіставити час натискання кнопки в інтерфейсі з реальною реакцією виконавчого механізму. Тестування показало, що модуль керування двигунами реагує з мінімальною затримкою, що важливо для точного позиціонування та маневрування.

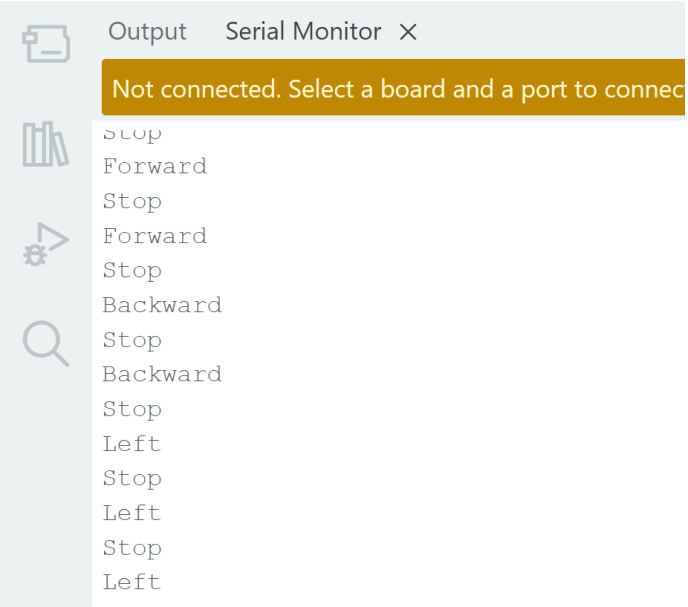


Рисунок 3.24 – Вивід в серійний монітор

У процесі тестування також оцінювалася стабільність тривалої роботи системи. АПК залишався ввімкненим упродовж декількох годин, протягом яких не було виявлено суттєвих збоїв або перегріву. Це свідчило про стабільну роботу як апаратної частини, так і програмної реалізації. Особливу увагу приділено сценаріям повторного з’єднання після втрати мережі – ESP32-CAM успішно відновлював з’єднання та автоматично перезапускав сервер.

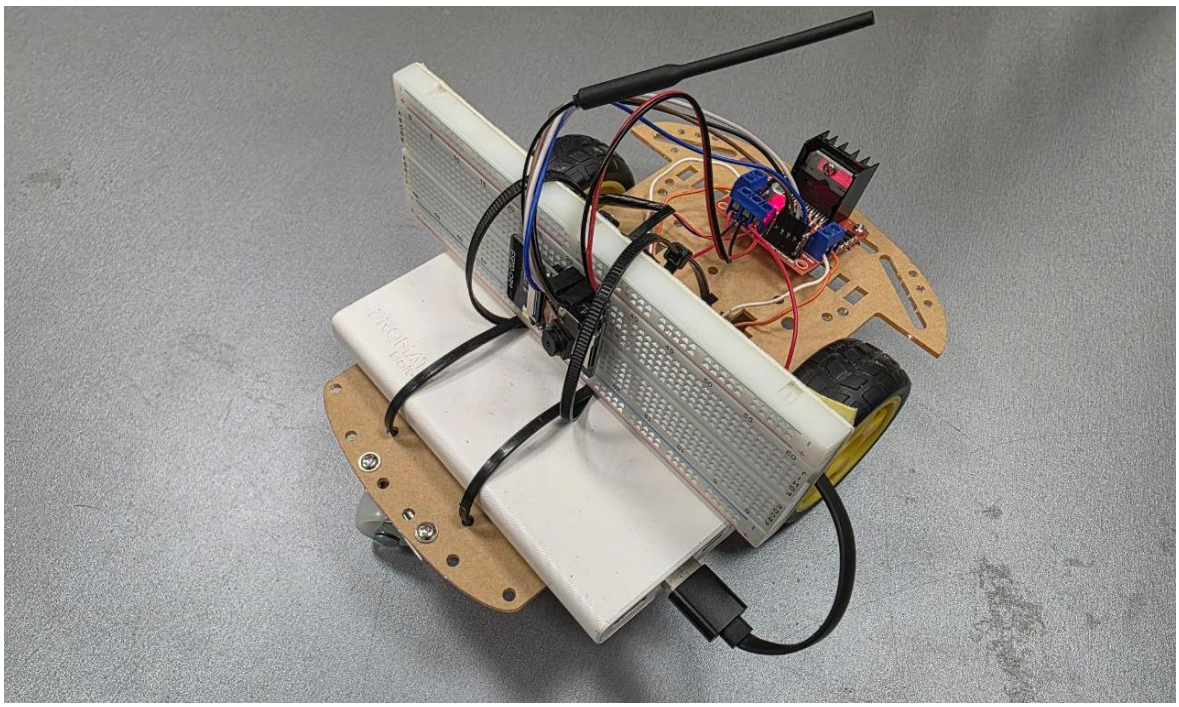


Рисунок 3.25 – Перевірка стабільності роботи за тривалий проміжок

Загалом, результати тестування підтвердили працездатність АПК відповідно до поставлених вимог. Комплекс виявився достатньо надійним для автономної або дистанційної роботи, демонструючи стабільну взаємодію між усіма складовими системи.

3.4 Потенціальні напрямки розвитку системи

У ході розробки та тестування автоматизованого програмно-апаратного комплексу було виявлено низку аспектів, які можуть бути вдосконалені для підвищення ефективності, стабільності та безпеки роботи системи. Одним із критичних напрямів є модернізація системи живлення. Поточне джерело забезпечує обмежену потужність, якої вистачає лише для роботи логіки мікроконтролера та базових навантажень, проте вона виявляється недостатньою під час одночасного живлення кількох електродвигунів або при роботі під навантаженням.

Для забезпечення повноцінного обертового моменту та стабільної швидкості обертання моторів необхідно підвищити напругу або струм, що подається на виконавчі механізми. Одним із варіантів є заміна джерела на потужніший блок живлення з вихідною напругою 6–12 В, який здатен витримувати високі пікові навантаження. Загалом, підвищення потужності живлення безпосередньо впливає на продуктивність виконавчих механізмів і розширює функціональні можливості комплексу, особливо в завданнях, де потрібна більша сила або швидкість руху.

Для покращення дальності дії системи, особливо в умовах відсутності стабільного WiFi-з’єднання, доцільно розглянути використання модулів з підтримкою зовнішніх антен або альтернативних бездротових технологій, таких як LoRa чи GSM/3G-модулі. Це дозволить розширити сферу застосування пристрою – наприклад, для моніторингу у важкодоступних або віддалених місцевостях.

Питання безпеки також потребує уваги. На поточному етапі вебсервер на базі ESP32 працює без шифрування, що створює ризик перехоплення

переданих даних. Впровадження захищених протоколів (HTTPS) або хоча б базової авторизації значно підвищило б рівень безпеки. Окрім того, доцільно реалізувати механізми контролю доступу, як-от списки дозволених клієнтів або змінні паролі, які оновлюються з певною періодичністю.

Ще одним із ключових напрямів удосконалення є заміна основи конструкції, на якій закріплено всі апаратні компоненти. Поточна легка або гнучка основа є недостатньо жорсткою, що знижує стабільність під час руху. Це спричиняє вібрації, які негативно впливають на роботу камери, точність позиціювання та зносостійкість системи. Також вона є вразливою до механічних пошкоджень, що обмежує її придатність до реального використання.

Для підвищення надійності конструкції слід замінити основу на більш міцний і жорсткий матеріал – наприклад, алюмінієву пластину, акрил товщиною понад 4 мм або текстоліт. Така модернізація зменшить рівень вібрацій і поліпшить фіксацію компонентів: двигунів, камери, контролера, джерела живлення. Окрім цього, жорстка основа полегшить подальші розширення системи – наприклад, додавання сенсорів, акумулятора більшої ємності або навіть другого поверху.

З погляду зручності користування варто також реалізувати окремий вебінтерфейс або мобільний застосунок. Це дозволить не лише здійснювати керування пристроєм, але й переглядати поточні показники з сенсорів, змінювати режими роботи та адаптувати функціонал під конкретні задачі. Особливо це буде корисно при експлуатації системи в польових умовах або за участі користувачів без технічної підготовки.

У підсумку, реалізація наведених покращень дозволить суттєво підвищити загальну якість, надійність і функціональність розробленого програмно-апаратного комплексу, відкриваючи нові можливості для його практичного застосування.

Висновок до розділу 3

У цьому розділі було детально описано процес розробки, складання та тестування автоматизованого програмно-апаратного комплексу на базі мікроконтролера ESP32. Розглянуто структуру проєкту, ключові програмні компоненти, зокрема реалізацію бездротового з’єднання, запуск вбудованого вебсервера, стрімінг відеосигналу з камери та управління електродвигунами. Також описано процес прошивки пристрою через USB-UART, що забезпечило зручне програмування та налагодження.

У ході тестування було виявлено як функціональні переваги, так і низку недоліків, що впливають на стабільність та надійність роботи системи. Зокрема, було визначено необхідність покращення системи живлення, посилення механічної основи конструкції, впровадження додаткових засобів безпеки при передачі даних, а також можливість використання альтернативних модулів зв’язку для збільшення зони покриття. Запропоновано низку напрямів для подальшої модернізації з метою підвищення ефективності, зручності та розширення функціональності комплексу.

Таким чином, проведена розробка підтвердила життєздатність обраної архітектури системи та створила основу для подальшого вдосконалення й масштабування.

ВИСНОВКИ

У результаті виконання КБР розроблено та протестовано АПК на базі мікроконтролера ESP32, що забезпечує функції відеоспостереження, дистанційного керування виконавчими механізмами та передачі даних через Wi-Fi. Під час виконання було:

– проведено аналіз сучасних рішень та технологій у галузі мобільних систем дистанційного спостереження;

– обґрунтовано вибір апаратного забезпечення для побудови мобільної платформи;

– розроблено архітектуру АПК;

– реалізовано програмне забезпечення для управління платформою та передавання відеоданих;

– проведено експериментальні дослідження функціональності комплексу.

Система пройшла повний цикл реалізації: від технічного обґрунтування до практичного тестування. Реалізований АПК продемонстрував стабільну роботу в умовах локальної мережі, можливість перегляду відео в реальному часі та дистанційного керування рухомими компонентами через вбудований вебінтерфейс.

Порівняння з аналогічними рішеннями свідчить про конкурентоспроможність запропонованої моделі завдяки її відкритості, універсальності, простоті масштабування та доступності компонентної бази. Досягнутий ступінь новизни проявляється у поєднанні недорогих апаратних засобів з функціональністю, притаманною більш складним системам. Практичне значення результатів КБР підтверджується можливістю застосування АПК у сфері моніторингу, безпеки, освіти або автоматизації локальних процесів.

Протягом розробки було також виявлено низку обмежень, які не завадили досягти мети, однак окреслили вектори подальшого вдосконалення

системи. Зокрема, було встановлено, що стабільність роботи АПК залежить від потужності живлення та конструкційної жорсткості основи. Це дало підстави для формування конкретних рекомендацій щодо модернізації – таких як заміна джерела живлення на більш потужне, використання жорсткішого матеріалу основи, впровадження захищених протоколів зв’язку та створення більш зручного користувацького інтерфейсу.

Таким чином, усі поставлені у вступі завдання були виконані, а функціональні, технічні й програмні характеристики АПК відповідають заявленим вимогам. Результати КБР можуть бути впроваджені у практичну діяльність або використані як основа для подальших досліджень і розробок у сфері вбудованих систем, робототехніки чи Інтернету речей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Raspberry Pi Camera. URL: https://www.raspberrypi.com/documentation/accessories/camera.html (Last accessed: 08.05.2025).

2. TurtleBot3. URL: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/ (Last accessed: 08.05.2025).

3. ESP 32 M5Stack. URL: https://docs.m5stack.com/en/unit/esp32cam (Last accessed: 08.05.2025).

4. RoboMaster S1. URL: https://www.dji.com/global/downloads/products/robomaster-s1#doc (Last accessed: 08.05.2025).

5. mBot Ranger 3-in-1 Educational Robot Kit. URL: https://qiniu.makeblock.com/makeblock-download/ranger/resource/mBot-Ranger_Blue_STD_EN_D2.1.3_7.40.3703_Edit-1.pdf (Last accessed: 08.05.2025).

6. ActivityBot 360. URL: https://www.parallax.com/activitybot-360/ (Last accessed: 08.05.2025).

7. Throwbot 2 Robot. URL: https://reconrobotics.com/products/throwbot-2-robot/# (Last accessed: 08.05.2025).

8. TTGO T-Journal ESP32 Camera: Built-in Programmer, OLED, Antenna and Project Examples. URL: https://randomnerdtutorials.com/ttgo-t-journal-esp32-camera-getting-started/ (Last accessed: 08.05.2025).

9. Модуль камери OV7670. URL: https://itmaster.biz.ua/directory/sensors/ov7670.html (дата звернення: 08.05.2025).

10. ESP32-CAM URL: https://docs.sunfounder.com/projects/galaxy-rvr/en/latest/hardware/cpn_esp_32_cam.html (Last accessed: 08.05.2025).

11. Робо-платформа однопалубна (3 колеса, 2 провідних) URL: https://arduino.ua/prod228-robo-platforma-odnopalybnaya-3-kolesa-2-vedyshhih (дата звернення: 08.05.2025).

12. L298N Dual H-Bridge Motor Driver. URL: https://cnc1.lv/PDF%20FILES/L298N%20Motor%20Driver%20manual.pdf?srsltid=AfmBOopMNQ2LyMZB6YnvN53S-qHBUnfooY1xOI7u4XUd4GW9YAqJlMgI (Last accessed: 08.05.2025).

13. Kumar B. Robot Vehicle with ESP32 CAM. *International Journal for Research in Applied Science and Engineering Technology*. 2024. Vol. 12, Is. 8. P. 1028–1031. DOI: 10.22214/ijraset.2024.64054.

14. Hasibuan A., Nasution T., Azis, P. MPU-6050 Wheeled Robot Controlled Hand Gesture Using L298N Driver Based on Arduino. *Journal of Physics Conference Series*. 2023. Vol. 2421, Is. 1. P. 1–10. Conference Series. DOI:10.1088/1742-6596/2421/1/012022.

15. Робо-платформа 4х колісна двопалубна повнопривідна. URL: https://arduino.ua/prod1908-robo-platforma-4-h-kolesnaya-dvyhpalybnaya-polnoprivodnaya (дата звернення: 08.05.2025).

16. Робо-платформа Міні 2-х палубна (4 колеса, 2 ведучих). URL: https://arduino.ua/prod278-roboplatforma-mini-2-h-palybnaya-4-kolesa-2-vedyshhih (дата звернення: 08.05.2025).

17. DIY-гусеничне шасі робота-танка TP101. URL: https://arduino.ua/prod4485-diy-gysenichnoe-shassi-robota-tanka-tp101 (дата звернення: 08.05.2025).

18. Arduino Uno. URL: https://doc.arduino.ua/ru/hardware/Uno (Last accessed: 08.05.2025).

19. ESP-IDF Programming Guide. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html (Last accessed: 08.05.2025).

20. PlatformIO. IDE URL: https://docs.platformio.org/en/latest/integration/ide/pioide.html (Last accessed: 08.05.2025).

21. Arduino Documentation URL: https://docs.arduino.cc (Last accessed: 08.05.2025).

22. Гуляєв І.С., Дарнапук Є.С., Комплекс дистанційного спостереження на базі колісної робоплатформи та ESP32-CAM. Могилянські читання – 2024 : тези доп. XXVII Всеукр. наук.-практ. конф. Миколаїв, 6–10 листоп. 2024 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2024. С. 108- 111 URL: https://dspace.chmnu.edu.ua/jspui/bitstream/123456789/2507/1/Технічні%20науки.pdf (дата звернення: 22.05.2025).

23. Fandidarma B., Dwi Laksono R.. Rancang Bangun Mobil Remote Control Pemantau Area berbasis IoT menggunakan ESP 32 Cam. *ELECTRA Electrical Engineering Articles*. 2021. Vol. 2, Is. 1. P. 31–38. Conference Series. DOI:10.25273/electra.v2i1.10522.

24. Downloads Arduino IDE 2.3.6. URL: https://www.arduino.cc/en/software/ (Last accessed: 02.06.2025).

25. Cameron N. ESP32-CAM Camera. *ESP32 Formats and Communication*. 2023. Vol. 1, Is 1. P. 447–488. DOI:10.1007/978-1-4842-9376-8_11.

26. Barrett S. Arduino PLC IDE and Ladder Logic. *Arduino VII*. 2025. Vol. 1, Is 1. P. 87–106. DOI:10.1007/978-3-031-68609-2_3.

ДОДАТОК А

Код прошивки для ESP32-CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_http_server.h"

const char* ssid = "Sw";
const char* password = "00000000";

#define PART_BOUNDARY "123456789000000000000987654321"

#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM_B
//#define CAMERA_MODEL_WROVER_KIT

#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #define SIOD_GPIO_NUM    26
  #define SIOC_GPIO_NUM    27

  #define Y9_GPIO_NUM      35
  #define Y8_GPIO_NUM      34
  #define Y7_GPIO_NUM      39
  #define Y6_GPIO_NUM      36
  #define Y5_GPIO_NUM      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
```

```
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27

  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM_B)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     22
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#else
  #error "Camera model not selected"
#endif
```

```
      #define MOTOR_1_PIN_1    14
      #define MOTOR_1_PIN_2    15
      #define MOTOR_2_PIN_1    13
      #define MOTOR_2_PIN_2    12

      static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-
replace;boundary=" PART_BOUNDARY;
      static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
      static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

      httpd_handle_t camera_httpd = NULL;
      httpd_handle_t stream_httpd = NULL;

      static const char PROGMEM INDEX_HTML[] = R"rawliteral(
      <html>
        <head>
          <title>ESP32-CAM Robot</title>
          <meta name="viewport" content="width=device-width, initial-scale=1">
          <style>
            * {
              box-sizing: border-box;
            }

            body {
              font-family: Arial, sans-serif;
              margin: 0;
              padding: 0;
              display: flex;
              flex-direction: column;
              align-items: center;
              height: 90vh;
              overflow: hidden;
            }
            h1 {
              margin: 10px 0;
            }
            .camera-container {
              flex: 1;
              display: flex;
              justify-content: center;
              align-items: center;
              overflow: hidden;
              max-height: 70vh;
              width: 100%;
            }
            .camera-container img {
              transform: rotate(90deg);
              max-height: 100%;
              max-width: 100%;
              object-fit: contain;
            }
            .controls {
              padding: 10px;
              width: 100%;
              display: flex;
              flex-direction: column;
              align-items: center;
            }

            table {
```

```
            margin: 0 auto;
          }

          td {
            padding: 8px;
          }

          .button {
            background-color: #2f4468;
            border: none;
            color: white;
            padding: 10px 20px;
            text-align: center;
            font-size: 18px;
            margin: 6px 3px;
            cursor: pointer;
            user-select: none;
            -webkit-tap-highlight-color: rgba(0,0,0,0);
          }
        </style>
      </head>
      <body>
        <h1>ESP32-CAM Robot</h1>

        <div class="camera-container">
          <img src="" id="photo">
        </div>

        <div class="controls">
          <table>
            <tr><td colspan="3" align="center"><button class="button"
onmousedown="toggleCheckbox('forward');" ontouchstart="toggleCheckbox('forward');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Forward</button></td></tr>
            <tr>
              <td align="center"><button class="button"
onmousedown="toggleCheckbox('left');" ontouchstart="toggleCheckbox('left');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Left</button></td>
              <td align="center"><button class="button"
onmousedown="toggleCheckbox('stop');"
ontouchstart="toggleCheckbox('stop');">Stop</button></td>
              <td align="center"><button class="button"
onmousedown="toggleCheckbox('right');" ontouchstart="toggleCheckbox('right');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Right</button></td>
            </tr>
            <tr><td colspan="3" align="center"><button class="button"
onmousedown="toggleCheckbox('backward');" ontouchstart="toggleCheckbox('backward');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Backward</button></td></tr>
          </table>
        </div>

        <script>
          function toggleCheckbox(x) {
            var xhr = new XMLHttpRequest();
            xhr.open("GET", "/action?go=" + x, true);
            xhr.send();
          }
          window.onload = function() {
```

```
            document.getElementById("photo").src = window.location.href.slice(0, -
1) + ":81/stream";
          };
        </script>
      </body>
    </html>
    )rawliteral";

    static esp_err_t index_handler(httpd_req_t *req){
      httpd_resp_set_type(req, "text/html");
      return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
    }

    static esp_err_t stream_handler(httpd_req_t *req){
      camera_fb_t * fb = NULL;
      esp_err_t res = ESP_OK;
      size_t _jpg_buf_len = 0;
      uint8_t * _jpg_buf = NULL;
      char * part_buf[64];

      res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
      if(res != ESP_OK){
        return res;
      }

      while(true){
        fb = esp_camera_fb_get();
        if (!fb) {
          Serial.println("Camera capture failed");
          res = ESP_FAIL;
        } else {
          if(fb->width > 400){
            if(fb->format != PIXFORMAT_JPEG){
              bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
              esp_camera_fb_return(fb);
              fb = NULL;
              if(!jpeg_converted){
                Serial.println("JPEG compression failed");
                res = ESP_FAIL;
              }
            } else {
              _jpg_buf_len = fb->len;
              _jpg_buf = fb->buf;
            }
          }
        }
        if(res == ESP_OK){
          size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART,
_jpg_buf_len);
          res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
        }
        if(res == ESP_OK){
          res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
        }
        if(res == ESP_OK){
          res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
        }
        if(fb){
          esp_camera_fb_return(fb);
          fb = NULL;
          _jpg_buf = NULL;
```

```
        } else if(_jpg_buf){
          free(_jpg_buf);
          _jpg_buf = NULL;
        }
        if(res != ESP_OK){
          break;
        }
        //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
      }
      return res;
    }

    static esp_err_t cmd_handler(httpd_req_t *req){
      char*  buf;
      size_t buf_len;
      char variable[32] = {0,};

      buf_len = httpd_req_get_url_query_len(req) + 1;
      if (buf_len > 1) {
        buf = (char*)malloc(buf_len);
        if(!buf){
          httpd_resp_send_500(req);
          return ESP_FAIL;
        }
        if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
          if (httpd_query_key_value(buf, "go", variable, sizeof(variable)) ==
ESP_OK) {
          } else {
            free(buf);
            httpd_resp_send_404(req);
            return ESP_FAIL;
          }
        } else {
          free(buf);
          httpd_resp_send_404(req);
          return ESP_FAIL;
        }
        free(buf);
      } else {
        httpd_resp_send_404(req);
        return ESP_FAIL;
      }

      sensor_t * s = esp_camera_sensor_get();
      int res = 0;

      if(!strcmp(variable, "forward")) {
        Serial.println("Forward");
        digitalWrite(MOTOR_1_PIN_1, 0);
        digitalWrite(MOTOR_1_PIN_2, 1);
        digitalWrite(MOTOR_2_PIN_1, 1);
        digitalWrite(MOTOR_2_PIN_2, 0);
      }
      else if(!strcmp(variable, "left")) {
        Serial.println("Left");
        digitalWrite(MOTOR_1_PIN_1, 1);
        digitalWrite(MOTOR_1_PIN_2, 0);
        digitalWrite(MOTOR_2_PIN_1, 1);
        digitalWrite(MOTOR_2_PIN_2, 0);
      }
      else if(!strcmp(variable, "right")) {
        Serial.println("Right");
```

```
    digitalWrite(MOTOR_1_PIN_1, 0);
    digitalWrite(MOTOR_1_PIN_2, 1);
    digitalWrite(MOTOR_2_PIN_1, 0);
    digitalWrite(MOTOR_2_PIN_2, 1);
  }
  else if(!strcmp(variable, "backward")) {
    Serial.println("Backward");
    digitalWrite(MOTOR_1_PIN_1, 1);
    digitalWrite(MOTOR_1_PIN_2, 0);
    digitalWrite(MOTOR_2_PIN_1, 0);
    digitalWrite(MOTOR_2_PIN_2, 1);
  }
  else if(!strcmp(variable, "stop")) {
    Serial.println("Stop");
    digitalWrite(MOTOR_1_PIN_1, 0);
    digitalWrite(MOTOR_1_PIN_2, 0);
    digitalWrite(MOTOR_2_PIN_1, 0);
    digitalWrite(MOTOR_2_PIN_2, 0);
  }
  else {
    res = -1;
  }
  if(res){
    return httpd_resp_send_500(req);
  }
  httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
  return httpd_resp_send(req, NULL, 0);
}
void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 1;
  config.ctrl_port += 1;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
```

```
  pinMode(MOTOR_1_PIN_1, OUTPUT);
  pinMode(MOTOR_1_PIN_2, OUTPUT);
  pinMode(MOTOR_2_PIN_1, OUTPUT);
  pinMode(MOTOR_2_PIN_2, OUTPUT);

  Serial.begin(115200);
  Serial.setDebugOutput(false);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  if(psramFound()){
    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.println(WiFi.localIP());

  // Start streaming web server
  startCameraServer();
}
void loop() {

}
```

ДОДАТОК Б

Матеріали апробації роботи

XXVII Всеукраїнська науково-практична конференція «Могилянські читання – 2024»

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 – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

Підсекція:
- **КОМП'ЮТЕРНА ІНЖЕНЕРІЯ**

7. Distributed function chaining with anycast routing / Wion Adrien et al. *Proceedings of the 2019 ACM Symposium on SDN Research.* 2019. P. 91–97. DOI: 10.1145/3314148.3314355.

УДК 004.9:007.52

Дарнапук Є. С.,
старший викладач кафедри комп'ютерної інженерії, аспірант,
Гуляєв І. С.,
бакалаврант,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

КОМЛЕКС ДИСТАНЦІЙНОГО СПОСТЕРЕЖЕННЯ НА БАЗІ КОЛІСНОЇ РОБОПЛАТФОРМИ ТА ESP32-CAM

В умовах сучасних загроз і викликів, з якими зіштовхується наша країна, зростає потреба у розвитку автономних та надійних систем для дистанційного спостереження і моніторингу небезпечних територій. Зокрема, це актуально для забезпечення безпеки на об’єктах інфраструктури та контрольованих зонах. Використання колісної робоплатформи з інтегрованим мікроконтролером ESP32-CAM [1–3] дозволяє створити ефективний та мобільний комплекс для оперативного отримання візуальної інформації з віддалених або небезпечних місць, мінімізуючи ризик для життя людей.

Концептуально, комплекс складається з наступних компонентів: центральний мікроконтролер ESP32-CAM, що відповідає за керування платформою та отриманням зображень навколишнього середовища з подальшим поширенням їх на клієнтські пристрої; колісна робоплатформа, L298N драйвер моторів та Li-Po батарей 18650 з контролером заряду.

Мікроконтролер ESP32-CAM (рис. 1) є ключовим елементом комплексу дистанційного спостереження, який використовується для отримання візуальних даних навколишнього середовища. Завдяки своїм можливостям, ESP32-CAM дозволяє інтегрувати відеоспостереження та аналіз зображень у мобільні роботизовані платформи, створюючи ефективну та економічно вигідну систему. Його двоядерний процесор Xtensa LX6 з тактовою частотою до 240 МГц забезпечує достатню обчислювальну потужність для обробки відеопотоків у режимі реального часу, що є критично важливим для оперативного реагування.

Рис.1. ESP32-CAM

Завдяки вбудованій 2-мегапіксельній камері OV2640, ESP32-CAM може захоплювати високоякісні зображення та відео з роздільною здатністю до 1600x1200 пікселів, що дозволяє системі точно ідентифікувати перешкоди або небезпечні об'єкти. Модуль Wi-Fi, який працює на частоті 2.4 ГГц, дозволяє бездротово передавати ці дані на сервер, де вони обробляються за допомогою OpenCV для виявлення потенційних загроз. Це дозволяє суттєво підвищити мобільність та гнучкість системи, оскільки рішення можуть прийматися в режимі реального часу без необхідності присутності оператора на місці.

Додатково, ESP32-CAM має слот для microSD карт, що дозволяє локально зберігати великі обсяги даних для подальшого аналізу. Режими енергозбереження мікроконтролера також сприяють тривалій автономній роботі, що важливо для роботи в зонах з обмеженим доступом до джерел живлення.

Завдяки своїй доступності, гнучкості та можливостям обробки відеоданих, ESP32-CAM є ідеальним рішенням для реалізації дистанційного спостереження в умовах сучасних загроз, забезпечуючи ефективний моніторинг та виявлення небезпечних предметів та локацій.

В якості робоплатформи використовується Smart Robot Chassis Kit (рис. 2), але можна використати будь-яку подібну платформу за функціоналом. Плюсом використання цієї платформи є її ціна – саме шасі коштує до 15 EUR, з мінусів – на нерівних поверхнях та на територіях з великою кількістю незначних перешкод, як-от гілки дерев та елементи побутового сміття, прохідність комплексу падає.

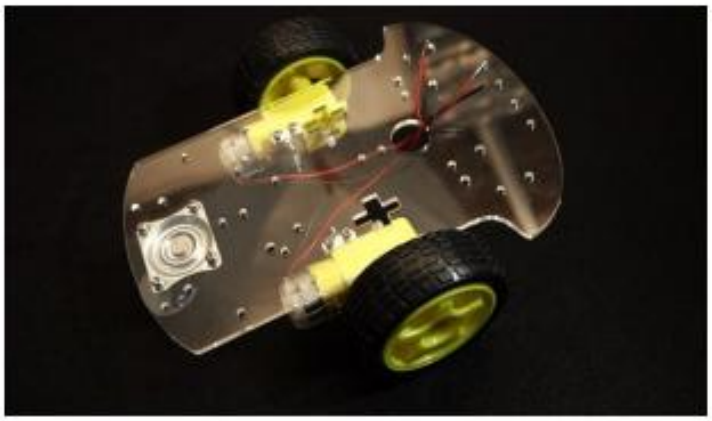


Рис.2. Smart Robot Chassis Kit

L298N драйвер мотора відіграє важливу роль в управлінні рухом колісної робоплатформи в системі дистанційного спостереження на базі ESP32-CAM [4]. Цей компонент дозволяє керувати двома двигунами постійного струму, забезпечуючи незалежне управління кожним колесом, що критично важливо для маневреності платформи в умовах реального часу. Завдяки можливості працювати з двигунами, які потребують живлення від 5 В до 35 В, L298N підтримує широкий діапазон варіантів живлення, дозволяючи використовувати як легкі, так і потужні двигуни в залежності від умов експлуатації.

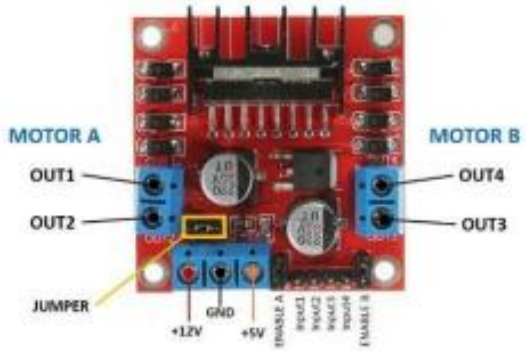


Рис.3. Драйвер мотора L298N

Вбудований Н-міст у драйвері забезпечує точний контроль над напрямком обертання моторів, що дозволяє платформі змінювати траєкторію руху для обходу перешкод або наближення до визначених цілей.

110

Крім того, надійний захист від перевантажень по струму і перегріву гарантує стабільну роботу навіть в екстремальних умовах, коли платформа може працювати без постійного нагляду. L298N разом із ESP32-CAM забезпечує плавну інтеграцію між системою збору даних і рухомими механізмами платформи, що дозволяє їй реагувати на аналізовані зображення та адаптувати свої дії відповідно до умов навколишнього середовища.

Розроблений комплекс дистанційного спостереження на базі колісної робоплатформи та мікроконтролера ESP32-CAM є важливим елементом системи спостереження та виявлення небезпек. Поєднуючи мобільну платформу з потужними можливостями відеозахоплення і аналізу зображень, система здатна виявляти перешкоди, небезпечні об'єкти та здійснювати спостереження за визначеними територіями без необхідності постійної присутності оператора. Використання ESP32-CAM у ролі основного обчислювального модуля дозволяє передавати зображення на сервер для подальшої обробки за допомогою OpenCV, що забезпечує швидку реакцію на загрози в реальному часі.

Список використаних джерел

1. Kumar B. A. Robot Vehicle with ESP32 CAM. *International Journal for Research in Applied Science and Engineering Technology*. 2024. Vol. 12, no. 8. P. 1028-1031. URL: https://doi.org/10.22214/ijraset.2024.64054.

2. IoT based social distance checking robot using Esp32-Cam / V. S. N. Reddy et al. *1st International Conference on Advances in Signal Processing, VLSI, Communications and Embedded Systems:* ICSVCE-2021, Hyderabad, India. 2021. DOI: 10.1063/5.0074056.

3. Remotely Controlled Firefighting Robot with ESP32-CAM Module / A. Yahya et al. *2023 2nd International Engineering Conference on Electrical, Energy, and Artificial Intelligence (EICEEAI),* Zarqa, Jordan, 27–28 December 2023. DOI: 10.1109/eiceeai60672.2023.10590289.

4. Application of drive circuit based on L298N in direct current motor speed control system / L. Yin et al. *International Symposium on Optoelectronic Technology and Application 2016*, Beijing, China / ed. by B. Lu, H. Wang. 2016. DOI: 10.1117/12.2246555.