Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

**Комплексна бакалаврська кваліфікаційна робота на тему:**

Кіберфізичний комплекс для автономної навігації з використанням маркерів.
Частина 2. Програмне забезпечення

08-54.КБКР.042.00.000 ПЗ

Виконав: студент 4 курсу, групи 2СП-21б
спеціальності 123 Комп'ютерна інженерія,
освітня програма «Системне програмування»
(шифр і назва напряму підготовки, спеціальності)
Шатайло В. А.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ОТ
Богомолов С. В.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф.МБІС
Шиян А. А.
(прізвище та ініціали)

**ДОПУЩЕНО**
Завідувач кафедри ОТ:
д.т.н., проф. Азаров О. Д
"30" 06 2025 р.

Вінниця ВНТУ — 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань — 12 — Інформаційні технології
Спеціальність — 123 — Комп'ютерна інженерія
Освітньо-професійна програма — Системне програмування

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ОТ

д.т.н., проф. Азаров О. Д

«21» березня 2025 р.

# З А В Д А Н Н Я
## НА КОМПЛЕКСНУ БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
## СТУДЕНТУ

Шатайлу В'ячеславу Андрійовичу

1 Тема роботи: «Кіберфізичний комплекс для автономної навігації з використанням маркерів. Частина 2. Програмне забезпечення», керівник роботи Богомолов С. В. затверджені наказом ВНТУ від «20» березня 2025 року №97.

2 Термін подання студентом роботи: 13.05.2025

3 Вихідні дані до роботи: Технічний опис мікроконтролерів Arduino та ESP32 для керування системою та обробки даних; середовище Arduino IDE для написання коду з використанням мови програмування C++; бібліотеки для роботи з Bluetooth.

4 Зміст розрахунково-пояснювальної записки: вступ, постановка завдання та аналіз системи, розробка програмного забезпечення, тестування і оцінка роботи системи, висновки, перелік використаних джерел, додатки.

5 Перелік графічного матеріалу: структурна схема системи.

Перелік ілюстративного матеріалу: алгоритм пошуку лінії, алгоритм

оминання перешкод.

6 Консультанти розділів роботи наведені у таблиці 1.

Таблиця 1 — Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | завдання прийняв |
| 1-3 | к.т.н., доц. каф. ОТ, Богомолов С. В. | 21.03.2025 | 13.05.2025 |
| Нормоконтроль | ас.. каф. ОТ, Швець С. І. | 14.05.2025 | 30.05.2025 |

7 Дата видачі завдання «21.03.2025 р. Календарний план наведений у таблиці 2.

Таблиця 2 — Календарний план

| № з/п | Назва етапів дипломної роботи | Строк виконання | Підпис |
|---|---|---|---|
| 1 | Постановка задачі роботи | 21.03.25 | вик. |
| 2 | Пошук матеріалів та інформаційних джерел | 21.03.25—30.03.25 | вик. |
| 3 | Аналіз існуючих прототипів та визначення вимог до створюваної системи | 21.03.25—30.03.25 | вик. |
| 4 | Обґрунтування структури та принципів роботи системи | 31.03.25—13.04.25 | вик. |
| 5 | Розробка й тестування програмної системи | 14.04.25—27.04.25 | вик. |
| 7 | Оформлення пояснювальної записки та ілюстративного матеріалу | 28.04.25—11.05.25 | вик. |
| 8 | Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків | 13.05.25 | вик. |

**Студент** В'ячеслав ШАТАЙЛО

**Керівник роботи** к.т.н., доц. Сергій БОГОМОЛОВ

# АНОТАЦІЯ

УДК 629.05

Шатайло В. А. Кіберфізичний комплекс для автономної навігації з використанням маркерів. Частина 2. Програмне забезпечення. Комплексна бакалаврська кваліфікаційна робота зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Системне програмування». Вінниця: ВНТУ, 2025. 81 с. На укр. мові. Бібліогр.: рис.: 21.

У комплексній кваліфікаційній роботі розроблено програмне забезпечення для автономної навігації кіберфізичного комплексу з використанням інфрачервоних датчиків. Створено алгоритми зчитування та обробки даних з ІЧ-датчиків, що дозволяють точно утримуватися на маршруті. Програмне забезпечення розподілено між мікроконтролерами Arduino Uno, ESP32-CAM та LOLIN D32. Окрему увагу приділено керуванню індикацією станів за допомогою світлодіодної стрічки WS2812B та відтворенню звукових сигналів через аудіомодуль з використанням інтерфейсу I2S. Уся логіка написана мовою C++ у середовищі Arduino IDE. Проведені тести підтвердили стабільність роботи, точність навігації та злагоджену взаємодію між усіма підсистемами.

Ключові слова: кіберфізичний комплекс, автономна навігація, Arduino, ESP32, інфрачервоні датчики, маркери, мікроконтролери.

# ABSTRACT

UDC 629.05

Shatailo V. A. Cyber-physical complex for autonomous navigation using markers. Part 2. Software. Comprehensive bachelor's qualification work in specialty 123 «Computer Engineering», educational program «System Programming». Vinnytsia: VNTU, 2025. 81 p. In Ukrainian. Bibliography: Fig. 21.

In this bachelor's thesis, software for autonomous navigation of a cyber-physical complex using infrared sensors was developed. Algorithms for reading and processing data from infrared sensors have been created to allow the vehicle to stay on the route accurately. The software is distributed between Arduino Uno, ESP32-CAM and LOLIN D32 microcontrollers. Particular attention is paid to controlling the status indication using the WS2812B LED strip and reproducing sound signals through the audio module using the I2S interface. All logic is written in C++ in the Arduino IDE environment. The tests have confirmed the stability of operation, navigation accuracy, and coordinated interaction between all subsystems. Translated with DeepL.com (free version)

Keywords: cyber-physical complex, autonomous navigation, Arduino, ESP32, infrared sensors, markers, microcontrollers.

**ЗМІСТ**

# ВСТУП

В умовах стрімкого розвитку технологій автономні системи займають одне з провідних місць у сфері робототехніки, логістики, промислової автоматизації та дослідницьких проєктів. Зростає потреба в компактних, доступних та розумних рішеннях, які здатні працювати без постійного втручання людини, орієнтуючись у середовищі за допомогою сенсорів, маркерів чи інших навігаційних засобів. Програмне забезпечення є ключовим елементом таких систем, адже саме воно забезпечує прийняття рішень, обробку даних у реальному часі та взаємодію між усіма складовими кіберфізичного комплексу.

**Актуальності** набувають рішення, що дозволяють здійснювати навігацію за допомогою візуальних або магнітних маркерів, а також стеженням за контрастними лініями за допомогою інфрачервоних датчиків. Такі підходи дозволяють значно спростити побудову траєкторій руху, забезпечити гнучкість маршруту і водночас — зменшити витрати на складні системи позиціонування. Застосування мікроконтролерів, зокрема Arduino та ESP32, робить ці технології доступними як для навчальних цілей, так і для практичних застосувань у промисловості.

На фоні зростаючого попиту на автономні рішення виникає необхідність у створенні програмного забезпечення, здатного не лише ефективно керувати їх рухом, але й забезпечувати стабільну роботу периферійних пристроїв — сенсорів, індикаторів, аудіомодулів. Важливим є також забезпечення взаємодії між різними мікроконтролерами, що працюють у межах однієї системи. Актуальним завданням стає інтеграція таких компонентів у єдиний керований комплекс.

Таким чином, тема розробки програмного забезпечення для кіберфізичного комплексу з функціями навігації, індикації та мультимедійної взаємодії є доволі актуальною. Вона охоплює ключові проблеми сучасної робототехніки — від алгоритмів руху до комунікації між електронними модулями — та відкриває широкі можливості для впровадження розробок у реальних умовах.

**Об'єктом дослідження** є кіберфізична система, що здійснює навігацію за допомогою візуальних маркерів.

**Предметом дослідження** є алгоритмічне та програмне забезпечення для автономної навігації комплексу, включаючи модулі обробки даних з маркерів і сенсорів, керування рухом, світловими індикаторами та звуковими ефектами.

**Метою роботи** є створення програмного забезпечення для автономного пересування кіберфізичної системи за маршрутом із використанням візуальних маркерів, із підтримкою світлової та аудіоіндикації.

У відповідності до поставленої мети необхідно вирішити такі **задачі**:

— проаналізувати апаратну частину системи

— розглянути сучасні підходи до програмної реалізації автономної навігації кіберфізичного комплексу;

— розробити алгоритми для зчитування та аналізу даних з маркерів і сенсорів;

— написати програмний код для керування рухом системи відповідно до заданого маршруту;

— забезпечити коректну взаємодію між мікроконтролерами Arduino та ESP32;

— реалізувати інтеграцію звукового супроводу та світлодіодної індикації станів системи;

— перевірити програмне забезпечення та оцінити коректність його роботи.

**Практичне значення** розробленої системи охоплює автономні мобільні платформи, призначені для навчання, досліджень і демонстрацій у сфері робототехніки. Використання візуальних маркерів, поєднане з мультимедійною взаємодією, дозволяє ефективно моделювати сценарії реального середовища. Реалізація системи на базі доступних мікроконтролерів та сенсорних модулів дає змогу знизити складність конструкції та зменшити вартість розробки у порівнянні з більш складними автономними рішеннями.

**Наукова новизна** полягає у створенні оригінальної програмно-апаратної

архітектури з розподілом функціоналу між кількома мікроконтролерами, що забезпечують навігацію за маркерами, аудіо та світлову індикацію. Система поєднує алгоритми обробки сенсорних даних, Bluetooth-аудіо та візуалізації спектру в реальному часі, що розширює можливості автономної взаємодії з користувачем.

**Апробація та публікація результатів** комплексної кваліфікаційної роботи була проведена на LIV Науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії (2025) [1].

**Публікації за темою роботи:** ОСОБЛИВОСТІ ЗАСТОСУВАННЯ МАРКЕРІВ У СИСТЕМАХ АВТОНОМНОЇ НАВІГАЦІЇ / В. А. Шатайло, С. В. Богомолов, Н. О. Черневський // Матеріали LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2025 р. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23995

# 1 ПОСТАНОВКА ЗАВДАННЯ І АНАЛІЗ СИСТЕМИ

## 1.1. Вимоги до функціональності програмного забезпечення

Функціональність руху є одним з основних компонентів програмного забезпечення автономного робота. Від якості реалізації цієї функції залежить здатність системи пересуватись у просторі згідно з визначеним маршрутом, уникати перешкод і реагувати на зміни навколишнього середовища в режимі реального часу.

Для досягнення впевненої та стабільної навігації використовуються дані з різних сенсорів, які зчитують характеристики поверхні, простору та об’єктів навколо. Особливе значення має система розпізнавання напрямку руху, яка може грунтуватись на інфрачервоному або оптичному відслідковуванні лінії чи маркерів. Такі маркери можуть слугувати умовними орієнтирами або маршрутами, за якими роботизований комплекс має рухатися [2].

Алгоритми навігації мають забезпечувати своєчасну обробку сенсорної інформації, точне керування приводами коліс або іншими виконавчими механізмами, а також стабільність поведінки при змінних умовах, таких як нерівності, повороти, або тимчасові перешкоди. Додатково може застосовуватись модуль виявлення перешкод, який дозволяє уникати зіткнень та приймати альтернативні рішення в ході руху.

Усі ці функції повинні працювати синхронно та без затримок, оскільки рух і навігація є основою автономності робота. Надійна програмна реалізація цих процесів дозволяє досягти точного й безпечного пересування, що є критично важливим у навчальних, промислових або дослідницьких застосуваннях.

Також для правильної та злагодженої роботи системи дуже важливо забезпечити надійний обмін інформацією між усіма функціональними модулями. У типовій конфігурації комплекс складається з кількох взаємопов’язаних частин: модуля керування рухом, сенсорного блоку, виконавчих пристроїв (наприклад, світлодіодів або динаміків), а також інтерфейсів зв’язку.

З технічної точки зору, комунікація між модулями може здійснюватися як за допомогою дротових інтерфейсів (наприклад, UART, I2C, SPI), так і бездротових технологій, таких як Wi-Fi або Bluetooth. Кожен з варіантів має свої переваги й недоліки, але незалежно від обраної технології головне завдання — забезпечити швидке, стабільне й надійне передавання інформації без втрат або затримок, особливо в умовах, коли робот працює автономно і не має зовнішнього контролю.

Ключова роль у цьому процесі належить програмному забезпеченню, яке має чітко визначити протоколи обміну даними, структуру переданих повідомлень, обробку запитів і відповідей, а також механізми синхронізації між модулями. Наприклад, коли сенсор виявляє зміну на маршруті, ця інформація має миттєво надходити до центрального контролера, який на основі отриманих даних ухвалює рішення щодо зміни напрямку руху або активації певних індикаторів.

Крім того, важливо враховувати можливість виникнення помилок або втрат даних у процесі обміну. Тому необхідно реалізувати базові механізми перевірки цілісності повідомлень, повторної передачі при необхідності, а також чітко визначити логіку поведінки системи у випадку втрати зв’язку між модулями.

Світлова індикація в системі призначена для візуального відображення станів роботи робота та створення додаткового інтерактивного ефекту. Вона реалізується за допомогою програмно керованих світлодіодів, які можуть змінювати колір, яскравість і режим роботи відповідно до поточного режиму роботи робота або зовнішніх впливів.

Основним завданням світлової індикації є підвищення зручності сприйняття інформації про стан пристрою, що особливо важливо під час автономної роботи або у складних умовах експлуатації. Вона також виконує функцію естетичного доповнення, роблячи робота більш привабливим і зрозумілим для користувача.

Програмне забезпечення забезпечує динамічне керування світловими ефектами, що можуть змінюватися в залежності від подій: запуску, зупинки, виявлення перешкод, руху за маршрутом тощо. Важливо, що індикація може бути синхронізована з іншими системами комплексу, наприклад, звуковою індикацією, що забезпечує комплексний зворотний зв’язок.

У реалізації світлової індикації застосовуються адресовані світлодіодні стрічки або окремі світлодіоди, що дає змогу ефективно налаштовувати їх поведінку в програмі. Для підтримки стабільної роботи індикації програмне забезпечення включає механізми управління яскравістю та частотою оновлення світлових сигналів.

Звукова індикація також є важливою складовою системи зворотного зв’язку між комплексом і користувачем. Вона дозволяє оперативно інформувати користувача про поточний стан пристрою, режим його роботи або виникнення певних подій. Такий тип індикації значно підвищує зручність у використанні системи, особливо в умовах, коли візуальний контроль є обмеженим або небажаним.

Звукові повідомлення можуть мати різну складність — від простих звукових сигналів до відтворення попередньо підготовлених аудіофайлів, які можуть включати голосові підказки, звукові ефекти або навіть музичний супровід. Такий підхід дає змогу реалізувати багаторівневу інформативну систему, де кожен звуковий сигнал відповідає конкретному стану або події.

Сучасні кіберфізичні системи часто використовують бездротові засоби передачі звуку, зокрема Bluetooth-модулі, що дозволяє передавати аудіо на зовнішні пристрої, такі як портативні динаміки або навушники. Це, в свою чергу, відкриває можливість для більш гнучкого налаштування інтерфейсу взаємодії користувача з комплексом. Таке рішення спрощує обслуговування та розширює можливості кастомізації під потреби конкретного користувача чи сценарію використання.

Програмне забезпечення системи виконує ключову роль у керуванні звуковими подіями. Звукова індикація синхронізується з іншими модулями —

такими як системи навігації, керування рухом, світловими ефектами — забезпечуючи узгоджене й цілісне функціонування пристрою. Наприклад, при переході робота в інший режим, спочатку змінюється поведінка підсвітки, активується відповідний аудіосигнал, і лише після цього виконується зміна алгоритму руху.

### 1.2. Обробка маркерів як орієнтирів руху

У системах автономного керування рухом, маркери виконують роль орієнтирів, які допомагають роботизованому комплексу визначати своє положення у просторі, реагувати на зміну траєкторії та приймати рішення щодо подальших дій. Обробка таких маркерів полягає у зчитуванні даних із відповідних сенсорів, аналізі отриманої інформації та формуванні команд для переміщення. Залежно від типу використаних маркерів, змінюється і спосіб їх обробки. Використання маркерів дозволяє зменшити потребу в складній навігаційній логіці, оскільки велика частина рішень приймається на основі простих візуальних або сенсорних сигналів, що значно спрощує побудову автономної поведінки. Окрім навігації, маркери також можуть використовуватись для активації певних дій або режимів роботи.

Оптичні маркери є одними з найпоширеніших засобів орієнтування для мобільних роботів у замкненому просторі. Їх головною перевагою є простота реалізації, легкість візуального виявлення, а також можливість гнучкого налаштування поведінки робота залежно від типу маркера. До оптичних маркерів можна віднести лінії, геометричні мітки, точки, стрілки, перехрестя, а також більш складні графічні елементи, такі як QR-коди або спеціалізовані візуальні ідентифікатори (наприклад, ArUco- або AprilTag-маркери).

Прикладом є траєкторна лінія, яку робот розпізнає за допомогою набору інфрачервоних сенсорів або фотодіодів [3]. Така лінія зазвичай наноситься чорною фарбою на світлу поверхню та виконує функцію маршруту, по якому має рухатися робот. Сенсори, встановлені на нижній частині шасі, постійно зчитують контрастність підлоги, і коли виявляється край лінії, алгоритм автоматично

коригує положення, змінюючи швидкість обертання одного з двигунів. Завдяки цьому робот здатен точно триматися заданого шляху. Такий підхід називається «line following» або слідкування за лінією. На рисунку 1.1 наведено приклад реалізації такої системи, розробленої за допомогою чорної клейкої стрічки, розміщеній у вигляді замкненого маршруту.

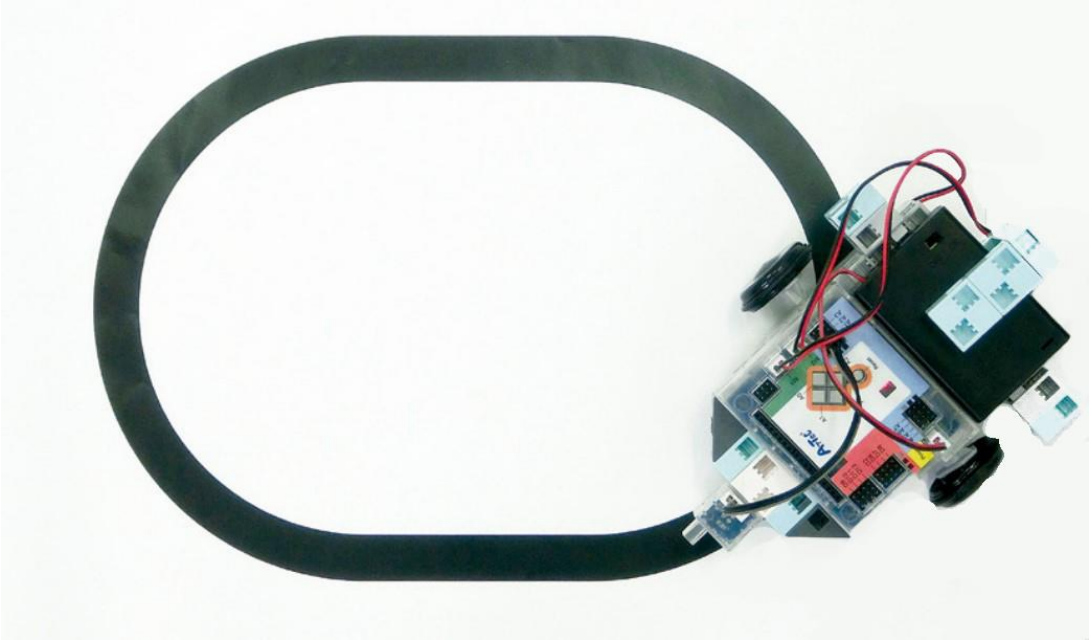

Рисунок 1.1 — Приклад реалізації системи слідкування за лінією

Для підвищення складності маршруту та гнучкості поведінки робота можуть використовуватись геометричні мітки або перехрестя, які несуть певну логічну інформацію. Наприклад, у місці перетину кількох ліній робот може «приймати рішення» — продовжити рух прямо, повернути, зупинитись або змінити режим роботи. Такі мітки можуть бути реалізовані у вигляді спеціальних форм, наприклад, квадратів, кіл, трикутників або послідовності точок, які розташовуються вздовж маршруту. Завдяки своїй наочності та простоті така система дозволяє швидко змінювати логіку маршруту без втручання в код — достатньо просто змінити розмітку на підлозі.

Більш складні оптичні маркери — це двовимірні коди, зокрема QR-коди або ArUco-маркери, які зчитуються за допомогою камери (рисунок 1.2). Вони дозволяють зашифрувати велику кількість інформації: ID місця, координати,

команду для робота або навіть короткий текст [4]. Для таких систем потрібно мати обчислювальні потужності для розпізнавання зображення — наприклад, камеру та мікроконтролер, здатний обробляти відеопотік або кадри. Завдяки цим маркерам робот може точно визначати свою позицію у просторі, змінювати логіку роботи відповідно до контексту або синхронізуватись з іншими пристроями.

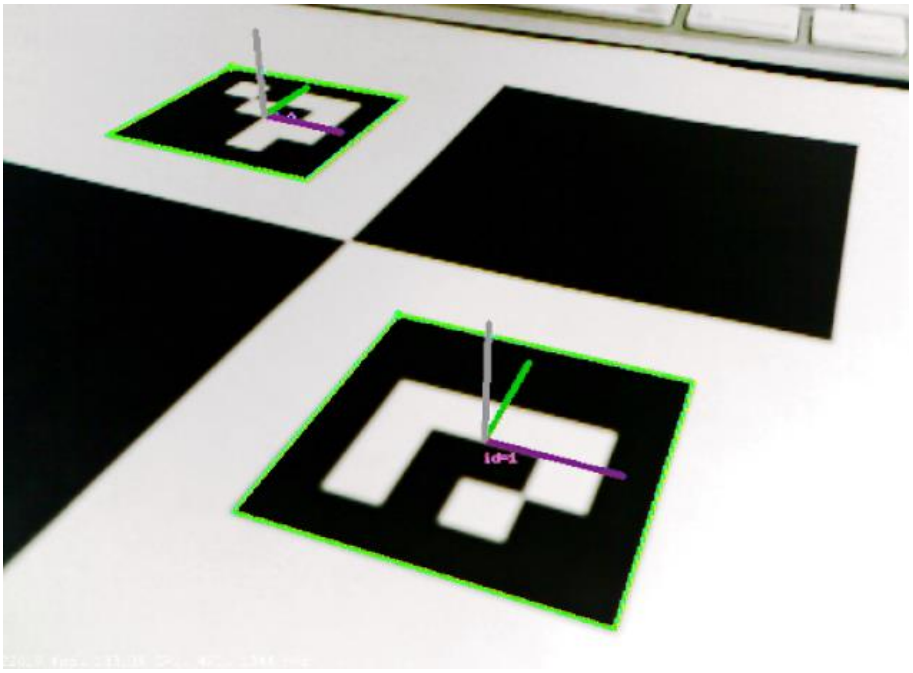


Рисунок 1.2 — Приклад ArUco-маркерів

Однією з переваг оптичних маркерів є їх невисока вартість та простота розгортання — достатньо лише надрукувати або наклеїти необхідні позначки на маршруті. Крім того, такі маркери не потребують живлення, мають високу довговічність і можуть легко змінюватися або оновлюватися в ході експлуатації системи.

Найпоширеніші технічні підходи до обробки оптичних маркерів включають використання порогової фільтрації, контурного аналізу, розпізнавання форм, а також комп’ютерного зору у поєднанні з машинним навчанням — у складніших реалізаціях. Наприклад, три інфрачервоні сенсори, розміщені в ряд, дозволяють роботу орієнтуватися відносно лінії: якщо

активується лише центральний сенсор — рух продовжується прямо, якщо активується один із бічних — виконується корекція траєкторії вліво або вправо.

Магнітні маркери використовуються як альтернатива оптичним орієнтирам, особливо там, де освітлення не завжди стабільне або де візуальне розпізнавання маркерів може ускладнюватись. У таких умовах робот не завжди здатен «побачити» лінію чи QR-код, зате чудово розпізнає магнітне поле. Саме тому магнітні маркери стають надійним рішенням для задач, пов’язаних із навігацією в складних середовищах.

Найчастіше вони реалізуються у вигляді тонкої магнітної стрічки, яка прокладається по поверхні маршруту. Іноді це можуть бути окремі невеликі магніти — наприклад, вставлені в ключових точках траєкторії, де потрібно звернути, зупинитись або змінити режим руху [5]. Робот, оснащений сенсорами на зразок датчиків Холла, зміна магнітного поля якими фіксується, і на основі цих сигналів робот приймає рішення: зупинитись, повернути, змінити швидкість або, наприклад, розпочати нову послідовність дій.

При цьому система не потребує ідеальної чистоти поверхні чи хорошого освітлення, як це часто буває у випадку з оптичними маркерами. Навіть якщо дорога забруднена або підлога зроблена з матеріалів, які візуально слабо контрастують, магнітна стрічка залишатиметься впізнаваною. Це особливо зручно для проєктів, що реалізуються в навчальних, лабораторних або виробничих умовах, де неможливо гарантувати стабільну візуальну інфраструктуру.

Важливо, що магнітні маркери можна легко заховати — наприклад, під тонкий шар плівки або підлогового покриття. Це дозволяє створити акуратну і майже невидиму систему навігації. Водночас, для правильної роботи необхідно уважно розміщувати магніти або стрічку, адже будь-яке зміщення може вплинути на точність зчитування. На рисунку 1.3 наведено приклад магнітної стрічки, яка корегує маршрут робота відповідно до її розміщення.

Рисунок 1.3 — Використання магнітної стрічки для корегування маршруту

Хоча магнітні маркери не є універсальними для всіх умов, у своїй ніші вони демонструють дуже високу ефективність. Їх зручно застосовувати там, де важлива надійність, а система має працювати без перебоїв навіть у ситуаціях, коли зовнішнє середовище не дозволяє використовувати камери або інфрачервоні сенсори.

Цифрові маркери, або електронні мітки, — це сучасний засіб орієнтації, що дозволяє забезпечити більш гнучку та інформативну взаємодію між роботом і навколишнім середовищем. На відміну від візуальних або фізичних маркерів, ці елементи не потребують безпосереднього контакту або зорової доступності — вони працюють на основі передачі даних за допомогою електронного сигналу.

Одним із найпоширеніших способів реалізації таких маркерів є використання технології RFID або NFC [6]. У цьому випадку робот оснащується зчитувачем, який, проходячи над електронною міткою, індукує в ній електричний струм, що дозволяє мітці відповісти унікальним радіосигналом, виявляючи її наявність у зоні дії та передаючи свій унікальний ідентифікатор або інші попередньо записані дані. Такі мітки можуть бути розміщені в ключових

точках маршруту — наприклад, на поворотах, розгалуженнях або контрольних точках. Після зчитування інформації мікроконтролер може приймати рішення щодо подальшого маршруту, активувати певний режим або передати дані іншим модулям системи. Приклад модулю для зчитування та програмування таких міток наведено на рисунку 1.4.


Рисунок 1.4 — Модуль для зчитування RFID-міток

Цифрові мітки мають суттєву перевагу в точності — відсутність необхідності у фізичному контакті або візуальній лінії огляду дозволяє зчитувати інформацію навіть у складних умовах, де, наприклад, оптичні сенсори можуть працювати ненадійно через пил, вологу або недостатнє освітлення. Крім того, дані на таких мітках можуть змінюватись — це дає змогу динамічно змінювати поведінку системи, оновлюючи логіку роботи без потреби змінювати сам код мікроконтролера.

Ще однією особливістю цих маркерів є можливість передавати не лише ідентифікаційний код, а й додаткову інформацію — наприклад, параметри маршруту, інструкції для виконання дій або статуси системи. Це значно розширює функціональні можливості таких орієнтирів і дозволяє створювати складні сценарії взаємодії між середовищем і автономним пристроєм.

Такі маркери особливо корисні в системах, де важлива точна ідентифікація місця, контексту або необхідна гнучкість у керуванні діями робота. Вони добре інтегруються в розподілені системи, в яких кілька мікроконтролерів взаємодіють між собою через електронні сигнали, та можуть використовуватись як елементи децентралізованого управління.

### 1.3. Розподіл функцій між апаратною частиною

Перед розробкою програмного забезпечення потрібно спочатку ознайомитися з апаратною платформою, яка буде використовуватись в кіберфізичному комплексі. Оскільки успішна робота системи значною мірою залежить від правильного розподілу функцій між мікроконтролерами, доцільно розпочати з аналізу ключового елемента керування — Arduino Uno.

Мікроконтролер Arduino Uno виступає як центральний елемент для управління базовими фізичними компонентами — сенсорами, двигунами та іншими виконавчими механізмами. Цей модуль зосереджений на обробці сигналів від простих аналогових і цифрових пристроїв, що забезпечує своєчасну реакцію на зміни у навколишньому середовищі. Зокрема, він відповідає за зчитування даних з інфрачервоних сенсорів, датчиків перешкод, а також маркерів, що використовуються для визначення позиції робота на маршруті (рисунок 1.5).


Рисунок 1.5 — Мікроконтролер Arduino Uno

Отримані дані одразу аналізуються мікроконтролером, і на їх основі формується логіка руху: обертання коліс, зміна напрямку або зупинка. З цією метою Uno керує драйверами двигунів, які безпосередньо впливають на оберти моторів та координують переміщення робота у просторі.

Окрім управління рухом, Arduino Uno також використовується для керування базовою індикацією — наприклад, включення світлодіодів при зміні режимів або сигналізації про зупинку. Крім того, він здатен приймати сигнали з кнопок чи перемикачів, що можуть використовуватись як елементи ручного керування або налаштування режимів роботи системи.

Модуль також реалізує передачу даних до інших компонентів, зокрема до модулів ESP32, через послідовний інтерфейс UART [7]. Це дає змогу делегувати складніші обчислювальні процеси зовнішнім контролерам, залишаючи Uno для виконання критичних задач у реальному часі. Завдяки стабільності, простоті використання та широкій підтримці спільноти, ця плата є надзвичайно зручною для реалізації саме таких апаратно-орієнтованих завдань.

Модуль ESP32-CAM поєднує в собі потужний мікроконтролер, вбудовану камеру та бездротові інтерфейси зв’язку, що робить його особливо зручним для реалізації функцій, пов’язаних із передачею даних, обробкою зображень та виконанням мультимедійних задач. Завдяки інтегрованій камері, цей модуль здатен здійснювати як фото, так і відеозйомку з можливістю передавання відеопотоку в реальному часі [8]. Це відкриває широкі можливості для створення систем відеомоніторингу, візуального контролю довкілля або орієнтування робота за допомогою зображень і маркерів. Окрім цього, ESP32-CAM може виконувати базову обробку зображення без участі зовнішніх пристроїв, що зменшує навантаження на інші елементи системи та пришвидшує реакцію на візуальні події. Крім роботи з відео, ESP32-CAM має розвинуті можливості бездротового зв’язку. Через Wi-Fi він може забезпечувати з’єднання з локальною мережею або створювати власну точку доступу для зв’язку з користувачем. Bluetooth дозволяє підключати зовнішні пристрої для керування, передачі даних або мультимедійного контенту. Наприклад, можливе використання Bluetooth для

передавання аудіо на зовнішні модулі з динаміком, що відкриває можливість реалізації звукових функцій або музичного супроводу (рисунок 1.6).


Рисунок 1.6 — Модуль ESP32-CAM

Ще однією поширеною функцією є керування світловими ефектами, зокрема світлодіодною індикацією. ESP32-CAM може аналізувати сигнали, отримані через Bluetooth або інші інтерфейси, і відповідно змінювати стан LED-елементів — створюючи, наприклад, світломузику, яка реагує на звук чи діє за заданим сценарієм.

Зв’язок між ESP32-CAM і іншими модулями часто реалізується через UART або інші послідовні інтерфейси. Така взаємодія дозволяє розподіляти завдання між мікроконтролерами: один відповідає за керування апаратними елементами, інший — за обробку даних та зв’язок із користувачем.

Мікроконтролер LOLIN D32, створений на базі чипа ESP32, є універсальним і достатньо потужним компонентом для виконання завдань, що вимагають швидкої обробки даних, управління зовнішніми пристроями та забезпечення бездротового зв’язку (рисунок 1.7). Завдяки високій продуктивності та великому об’єму оперативної пам’яті, цей модуль часто використовується для функцій, які потребують більшої обчислювальної потужності [9].


Рисунок 1.7 — Мікроконтролер LOLIN D32

LOLIN D32 підтримує Wi-Fi і Bluetooth, тому може використовуватись як окремий вузол зв’язку в системі. Це дозволяє, наприклад, організувати стабільну взаємодію з мобільним застосунком або забезпечити обмін даними між модулями через бездротову мережу. Також модуль може працювати як посередник, який приймає команди або дані з інших пристроїв і надсилає їх на мікроконтролери, що безпосередньо керують виконавчими механізмами.

Завдяки підтримці сучасних інтерфейсів — таких як UART, I2C, SPI — модуль легко інтегрується з різноманітними сенсорами та модулями. Наприклад, через I2C або UART можна реалізувати зв’язок з Arduino або ESP32-CAM, розділивши обов’язки між мікроконтролерами для досягнення кращої продуктивності та стабільності роботи.

Крім цього, модуль зручно використовувати для задач, що потребують постійної реакції на події в реальному часі. Наприклад, він може відповідати за синхронізацію світлових ефектів або за обробку потокових даних із сенсорів. Завдяки значному об’єму флеш-пам’яті, LOLIN D32 може зберігати складні програми, реалізовувати декілька режимів роботи, а також підтримувати динамічне конфігурування без потреби в перепрошивці.

### 1.4. Зв’язок між системою

Ефективна взаємодія між окремими модулями системи — одна з ключових умов її стабільної та гнучкої роботи. У проєктах, де використовується кілька мікроконтролерів, сенсорів і виконавчих пристроїв, важливо забезпечити

швидкий і надійний обмін даними між усіма компонентами. У цьому контексті використовуються кілька різних інтерфейсів — кожен із них виконує свою роль залежно від задачі та особливостей конкретного модуля.

UART (Universal Asynchronous Receiver-Transmitter) є одним із найпростіших і водночас найнадійніших способів організації обміну даними між мікроконтролерами. Цей інтерфейс працює на принципі асинхронної послідовної передачі, що не потребує окремої лінії синхронізації. Уся інформація передається по двох лініях — одна відповідає за надсилання сигналу (TX), інша — за його прийом (RX) (рисунок 1.8). Завдяки такому підходу, UART не потребує складного підключення чи великої кількості дротів, що робить його надзвичайно зручним для використання у компактних системах [10].

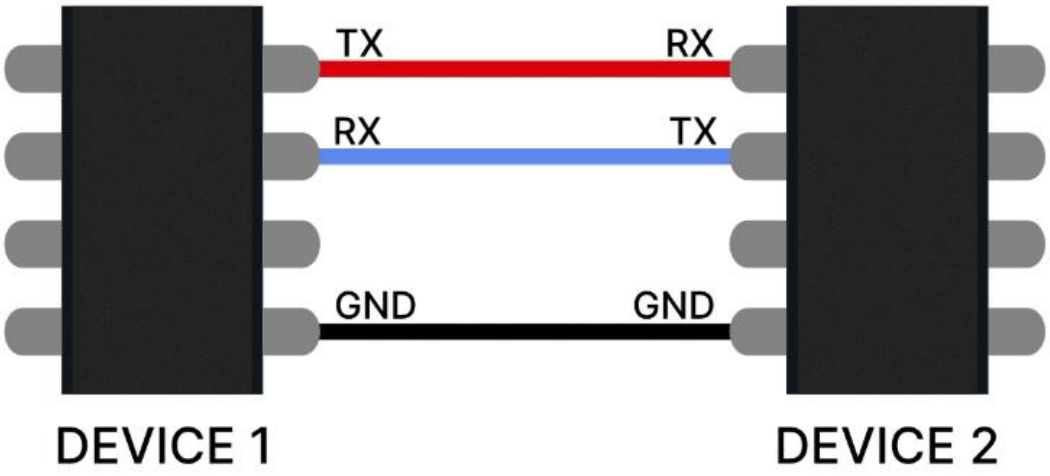


Рисунок 1.8 — Схема обміну даними за допомогою UART

Однією з ключових переваг UART є його незалежність від тактового сигналу, що дозволяє з’єднувати пристрої з різними системами синхронізації. На практиці UART дуже часто застосовується для зв’язку між двома пристроями, які безпосередньо взаємодіють між собою — наприклад, між мікроконтролером Arduino Uno та ESP32 або між ESP32-CAM і модулем, який обробляє інші сигнали. Перед передачею обидва пристрої узгоджують швидкість обміну (baud rate), що дозволяє синхронізувати прийом і передачу без додаткових сигналів. Це робить UART ідеальним вибором для сценаріїв, де важлива швидка

реакція на події в реальному часі, особливо коли мова йде про обробку даних з сенсорів або оперативне надсилання команд.

UART дозволяє передавати як окремі байти, так і цілі повідомлення у вигляді послідовностей, що можуть бути заздалегідь структуровані у вигляді простого протоколу. Наприклад, Arduino Uno може зчитувати інформацію з інфрачервоних або магнітних сенсорів, аналізувати її, а потім надсилати короткі повідомлення з результатами або командами далі — до ESP32, який уже відповідає за передачу даних користувачу або взаємодію з іншими модулями системи. Такий підхід дозволяє розділити функції між мікроконтролерами та забезпечити швидку обробку важливих подій без затримок.

Також слід враховувати певні обмеження: UART не підтримує багатоточковий обмін — тобто по одній лінії можуть працювати лише два пристрої, а також потребує чітко визначеної швидкості передачі на обох кінцях, інакше можливі збої в декодуванні сигналу.

Інтерфейс I2C (Inter-Integrated Circuit) є одним із ключових стандартів, що забезпечує ефективну та структуровану організацію зв’язку між мікроконтролерами та периферійними пристроями в сучасних цифрових системах. Його популярність обумовлена насамперед поєднанням простої апаратної реалізації з високою функціональністю. У своїй основі цей інтерфейс реалізує концепцію шинної архітектури, що дозволяє декільком пристроям одночасно взаємодіяти через спільну дволінійну комунікаційну магістраль. Завдяки цьому I2C знаходить широке застосування в багатьох галузях — від побутової електроніки та робототехніки до вбудованих систем автоматизації та приладобудування.

Фізично інтерфейс потребує лише дві сигнальні лінії — лінію передачі даних (SDA) та лінію синхронізації (SCL). Такий підхід значно спрощує трасування друкованих плат і дозволяє мінімізувати кількість з’єднань, що є критично важливим у компактних пристроях або в системах з великою кількістю функціональних модулів [11]. Водночас на логічному рівні шина I2C підтримує повноцінну багатоточкову архітектуру, де один пристрій виконує роль ведучого

(master), а решта — підлеглих (slaves). Це забезпечує централізоване управління обміном інформацією, дозволяючи майстру ініціювати передачу даних, адресувати конкретні пристрої та координувати послідовність операцій згідно з встановленим протоколом (рисунок 1.9).

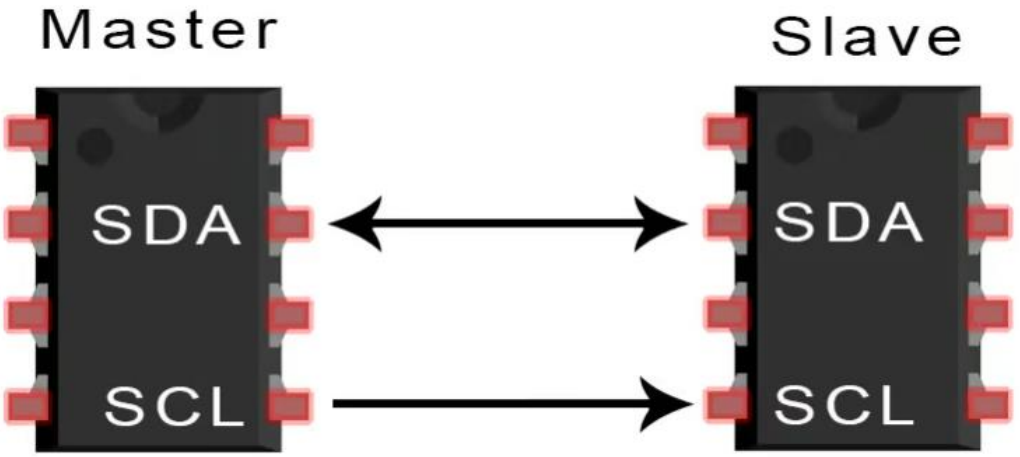


Рисунок 1.9 — Обмін даними за допомогою I2C

Однією з важливих характеристик інтерфейсу є його адресна модель. Кожен підлеглий пристрій у системі має унікальну 7-бітну або 10-бітну адресу, що дозволяє майстру індивідуально звертатися до кожного з них, не викликаючи конфліктів або помилок. Такий механізм створює передумови для гнучкого масштабування системи: до існуючої шини можна безперешкодно додавати нові модулі — сенсори, дисплеї, контролери — без потреби в змінах апаратної частини, достатньо лише оновити програмну конфігурацію, включивши нові адреси у логіку взаємодії.

Процес передачі даних в I2C відбувається у синхронному режимі, що означає використання єдиного тактового сигналу, сформованого ведучим пристроєм. Це дозволяє узгоджено працювати всім учасникам мережі незалежно від їхньої швидкодії або внутрішньої архітектури. Пакети даних передаються у чітко визначеній структурі, яка включає стартовий та стоповий сигнали, адресну частину, контрольні біти та підтвердження прийому (ACK/NACK). Такий підхід гарантує надійність та передбачуваність комунікації навіть у системах зі складною логікою.

I2C є особливо зручним у випадках, коли необхідно реалізувати періодичне опитування сенсорів або зчитування даних із зовнішніх джерел пам’яті. Наприклад, у багатьох мікроконтролерних застосунках ведучий пристрій регулярно звертається до датчика температури або тиску для оновлення показників, або здійснює синхронізацію часу з RTC-модулем. Крім того, цей інтерфейс дозволяє організовувати обмін між двома мікроконтролерами, коли один із них виконує допоміжну функцію — наприклад, відповідає за обробку даних, комунікацію з мережею або виконання фонових розрахунків — за запитом основного контролера.

Водночас, як і будь-який інтерфейс, I2C має певні обмеження. Серед них — чутливість до довжини сигнальних ліній, особливо при високих швидкостях передачі. Як правило, ефективна робота гарантована в межах декількох десятків сантиметрів, що обумовлено паразитними ємностями та індуктивностями провідників. Також слід враховувати обмежену швидкість обміну (від 100 кбіт/с у стандартному режимі до 3,4 Мбіт/с у high-speed-режимі), яка може бути недостатньою для високопродуктивних застосунків. Проте в межах локальних систем із помірним трафіком даних ці недоліки не мають суттєвого впливу.

Для взаємодії між користувачем та кіберфізичним комплексом доцільно використовувати бездротовий зв’язок, який дозволяє передавати дані без фізичного підключення кабелів, що особливо зручно в мобільних або розподілених системах. Одним із найпоширеніших і найбільш універсальних середовищ для бездротової комунікації є частотний діапазон 2,4 ГГц, у якому працює більшість сучасних протоколів зв’язку (рисунок 1.10).

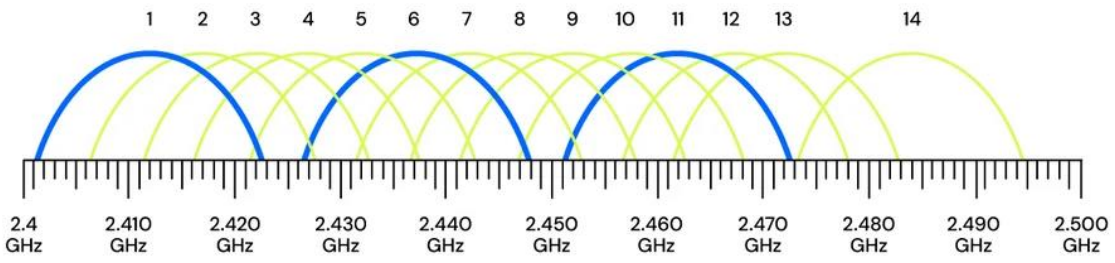


Рисунок 1.10 — Частотний діапазон технології Wi-Fi

Діапазон поділений на 14 каналів, з яких у Європі зазвичай використовуються 13, а в деяких країнах — усі 14. Канали частково перекриваються, тому для стабільного з'єднання часто обирають 1, 6 або 11, які не накладаються один на одного. Це дозволяє зменшити вплив взаємних перешкод і забезпечити кращу якість зв’язку.

До того ж цей діапазон не потребує ліцензування, має добру здатність проходження крізь перешкоди та підтримується переважною більшістю електронних пристроїв — від мікроконтролерів до смартфонів. У ньому функціонують такі популярні технології, як Wi-Fi, Bluetooth та ZigBee. Кожна з них має свої особливості, які визначають її придатність до певних задач — від передачі великих обсягів інформації до підтримки енергоефективних сенсорних мереж.

Wi-Fi є одним із найбільш гнучких і функціонально багатих способів організації бездротової комунікації між модулями в сучасних електронних системах. На відміну від дротових інтерфейсів, таких як UART або I2C, Wi-Fi дозволяє здійснювати передачу даних на значно більшу відстань та забезпечує зв’язок із пристроями, які можуть бути фізично розташовані в різних частинах системи або навіть у різних приміщеннях [12]. Він функціонує в діапазоні 2,4 ГГц або 5 ГГц (залежно від стандарту), має високу пропускну здатність — від 11 Мбіт/с (802.11b) до понад 150 Мбіт/с (802.11n) та навіть вище в сучасних версіях (ac/ax). Це відкриває широкі можливості для створення розподілених систем управління, моніторингу та взаємодії з користувачем.

Wi-Fi може бути використаний як для прямого обміну даними між модулями, так і для підключення до зовнішньої інфраструктури, наприклад, до локальної мережі або інтернету. Один з найбільш зручних способів застосування — це організація локального сервера на базі одного з модулів (наприклад, ESP32), до якого інші пристрої можуть підключатися в якості клієнтів. Таким чином, створюється внутрішній канал зв’язку, через який передаються команди, дані сенсорів, зображення або повідомлення про стан системи.

У системах з кількома модулями може виступати як міст між окремими частинами, які виконують специфічні функції. Наприклад, один з мікроконтролерів відповідає за збір даних, інший — за обробку або збереження, а третій — за відображення або передачу інформації користувачу. Усі вони можуть взаємодіяти через локальну бездротову мережу, синхронізуючи дії та обмінюючись повідомленнями у режимі реального часу.

Серед переваг Wi-Fi — висока швидкість передачі, можливість асинхронної взаємодії та підтримка стандартних мережевих протоколів. Проте для стабільної роботи потрібно враховувати якість сигналу, вплив перешкод, споживання енергії, особливо у мобільних пристроях або роботизованих платформах з автономним живленням. В більшості випадків, якщо система працює в межах приміщення або у стабільному середовищі, ці фактори не становлять критичної загрози.

Bluetooth — це один із найбільш поширених стандартів бездротового зв’язку ближнього радіуса дії, що функціонує в неліцензованому діапазоні частот 2,4 ГГц та базується на технології частотного перестрибування (FHSS — Frequency Hopping Spread Spectrum). Завдяки простоті реалізації, широкій підтримці на мобільних платформах і низькому енергоспоживанню, Bluetooth часто використовується в портативних, персоналізованих та вбудованих пристроях. У сучасних мікроконтролерах, таких як ESP32, Bluetooth підтримується апаратно, без потреби в зовнішніх модулях, що значно спрощує інтеграцію в готову систему [13].

У системі, реалізованій у межах даної роботи, Bluetooth можна використати для прийому аудіосигналу в режимі реального часу за допомогою профілю A2DP (Advanced Audio Distribution Profile). Цей профіль дозволяє здійснювати передавання стереоаудіо з мобільного пристрою (наприклад, смартфона або ноутбука) без втрати якості та з мінімальними затримками. Отриманий аудіопотік передається через I2S-інтерфейс до модуля цифрового підсилювача та виводиться на зовнішні динаміки. Одночасно сигнал аналізується для реалізації динамічної світлодіодної індикації — так званої

«світломузики», яка змінює колір та яскравість світлодіодів відповідно до ритму музики. Це забезпечує не лише функціональність, а й естетичний компонент взаємодії з користувачем.

Технічно Bluetooth (версії 4.2 та 5.0, які підтримує ESP32) забезпечує передачу даних зі швидкістю до 2 Мбіт/с на відстані до 30 метрів, залежно від класу передавача. Клас 2 (найбільш поширений у вбудованих пристроях) має радіус дії до 10 м і споживає до 2,5 мВт, що робить його енергоефективним навіть у системах з автономним живленням. При цьому Bluetooth не потребує складної інфраструктури: з'єднання встановлюється безпосередньо між двома пристроями за принципом «master-slave», де ініціатор (наприклад, смартфон) підключається до приймача (ESP32).

Крім мультимедійних функцій, Bluetooth також використовується для передавання службових даних — команд, параметрів конфігурації, повідомлень про стан системи. Це робить його універсальним каналом зворотного зв'язку між користувачем та пристроєм. Наприклад, Bluetooth можна використати для перемикання режимів роботи, керування яскравістю підсвітки, вибору аудіотреку або запуску певних сценаріїв без підключення до мережі Wi-Fi чи інтернету.

Порівняно з іншими бездротовими інтерфейсами, Bluetooth має очевидні переваги саме в контексті локальної взаємодії. Його перевага над Wi-Fi — у меншому енергоспоживанні, швидшому з'єднанні та відсутності потреби в зовнішньому маршрутизаторі. У порівнянні з ZigBee — у значно більшій швидкості передачі, підтримці потокового аудіо та наявності стандартної підтримки на всіх сучасних мобільних пристроях.

ZigBee — це бездротовий протокол ближнього радіуса дії, розроблений для передачі невеликих обсягів даних з мінімальним енергоспоживанням у розподілених мережах. Побудований на стандарті IEEE 802.15.4, ZigBee працює у діапазоні 2,4 ГГц, але також може функціонувати в інших частотних діапазонах [14]. Його основне призначення — забезпечити стійкий та масштабований обмін інформацією між численними пристроями в мережі типу

mesh, де кожен вузол може виступати не лише як передавач чи приймач, але й як ретранслятор.

Максимальна швидкість передачі даних у ZigBee становить до 250 Кбіт/с, чого цілком достатньо для обміну простими телеметричними даними, командами або повідомленнями стану. Завдяки цій характеристиці протокол ідеально підходить для систем автоматизації, моніторингу, розумного освітлення, охоронних комплексів та інших рішень, де важлива енергоефективність і надійна передача без високих вимог до пропускної здатності.

Однією з ключових переваг ZigBee є підтримка mesh-архітектури, яка дозволяє динамічно розширювати мережу за рахунок додавання нових вузлів без необхідності централізованого керування. У таких мережах сигнал автоматично знаходить альтернативні маршрути до адресата, що підвищує відмовостійкість і дозволяє працювати навіть у складних середовищах з перешкодами. Це робить ZigBee дуже ефективним у масштабних інсталяціях — наприклад, у промислових зонах або будівлях з багатьма фізичними перешкодами. Такі пристрої можуть працювати від батареї протягом кількох років без підзарядки, що робить їх ідеальними для автономних сенсорів, встановлених у важкодоступних місцях.

Однак поряд із перевагами ZigBee має і низку суттєвих обмежень, які обмежують його застосування в мобільних або мультимедійних системах. По-перше, обмежена швидкість передачі даних унеможливлює використання ZigBee для потокової передачі відео чи аудіо. По-друге, ZigBee не підтримується напряму більшістю мобільних пристроїв, тому для взаємодії з користувачем через смартфон або планшет потрібні додаткові шлюзи, конвертери або спеціалізовані додатки. Це значно ускладнює інтеграцію ZigBee у системи, де потрібна проста й швидка взаємодія.

У порівнянні з Wi-Fi або Bluetooth, ZigBee краще підходить для мереж великого масштабу, але не для систем, які орієнтовані на мультимедійні функції або інтерактивну роботу з користувачем. У контексті розробленої роботизованої платформи, де передбачено передачу аудіо, управління підсвіткою,

дистанційний контроль з мобільного пристрою, ZigBee є надмірно складним і обмеженим у функціональності. Наявність в ESP32 апаратної підтримки Wi-Fi та Bluetooth робить саме ці інтерфейси логічно виправданим вибором, тоді як використання ZigBee призвело б до ускладнення архітектури без суттєвих технічних переваг у даному випадку.

Таким чином, хоча ZigBee має важливе значення в галузі IoT, розумного дому та енергоефективних мереж, у межах даної роботи його застосування не є доцільним через специфіку задач, орієнтованість на мультимедійний контент і необхідність прямої взаємодії з мобільним користувачем.

# 2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Налаштування середовища розробки

Для розробки програмного забезпечення, яке керує автономним рухом робота та взаємодіє з апаратними модулями, необхідно попередньо підготувати середовище розробки. Основним інструментом для написання, компіляції та завантаження коду в мікроконтролери у даному проєкті є Arduino IDE — офіційне середовище програмування для платформ Arduino та сумісних пристроїв [15]. Його перевагами є відкритий код, простий інтерфейс, підтримка великої кількості бібліотек та повна сумісність із контролерами Arduino Uno, ESP32 та ESP32-CAM, що використовуються в системі.

На початковому етапі необхідно завантажити актуальну версію Arduino IDE з офіційного сайту. Для Windows рекомендується використовувати інсталятор Windows Installer, який спрощує процес установки (рисунок 2.1).


Рисунок 2.1 — Варіанти завантаження Arduino IDE

Після завантаження виконується стандартна процедура інсталяції з прийняттям ліцензійної угоди та вибором типових параметрів. У результаті на робочому столі з'являється ярлик для запуску середовища.

Після встановлення IDE важливо переконатися, що комп’ютер розпізнає підключену плату Arduino Uno. Для цього плата з’єднується з комп’ютером через USB-кабель, а в середовищі розробки через меню Tools → Board вибирається пункт Arduino/Genuino Uno, після чого у меню Tools → Port необхідно обрати відповідний COM-порт. Якщо пристрій не відображається автоматично, може знадобитись встановлення драйвера вручну через диспетчер пристроїв, зазначивши папку з драйверами в каталозі встановленої IDE. У випадку з Windows 10 та новіше драйвери, як правило, встановлюються автоматично.

Далі середовище готове до роботи. У програмному вікні доступна панель інструментів для перевірки коду, його завантаження, відкриття серійного монітора та керування проєктами (рисунок 2.2).

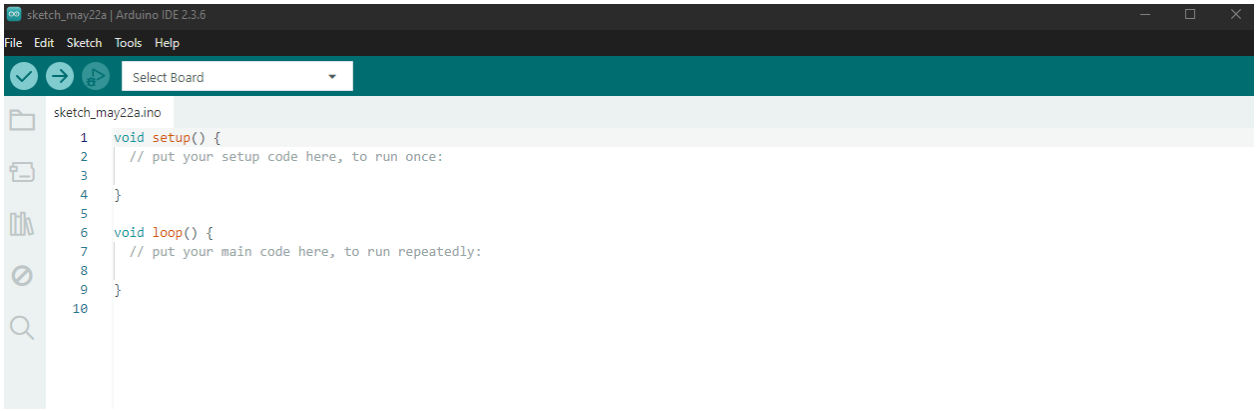


Рисунок 2.2 — Візуальний інтерфейс Arduino IDE

Важливо, що під час завантаження програми на плату потрібно перевести перемикач на самій платі у режим «Upload». Після завершення процесу слід повернути перемикач у положення «Cam» для нормальної роботи з мобільним застосунком.

Окрему увагу слід звернути на правильне підключення мікроконтролера ESP32 та модуля ESP32-CAM. Для роботи з ними в Arduino IDE необхідно встановити додаткові плати через Boards Manager, додавши посилання на платформи ESP32 до параметрів середовища. Після встановлення плат ESP32 відкривається можливість завантаження коду також і на ці модулі — наприклад, для реалізації Wi-Fi зв’язку або обробки зображень з камери.

## 2.2 Використання бібліотек

У процесі розробки програмного забезпечення для кіберфізичного комплексу важливу роль відіграло використання спеціалізованих бібліотек, що дозволили суттєво скоротити час реалізації функціоналу, забезпечити сумісність із апаратними модулями та підтримку складних протоколів зв'язку і обробки сигналів. Бібліотеки забезпечували прямий доступ до низькорівневих функцій, при цьому приховуючи складність реалізації внутрішніх механізмів, таких як PWM-керування, передача аудіо по Bluetooth або передача даних через Wi-Fi.

Для реалізації бездротового з'єднання через Wi-Fi використовувалась стандартна бібліотека WiFi.h, яка є частиною середовища ESP32 Arduino Core. Вона надає інтерфейс для підключення модуля ESP32 до локальної мережі у режимі клієнта або точки доступу (AP). У рамках проєкту цей функціонал був використаний для створення веб-сервера, який дозволяє користувачу переглядати потік з камери або надсилати команди керування.

Передача аудіосигналу через Bluetooth була реалізована за допомогою бібліотеки BluetoothA2DPSink, яка забезпечує повноцінну реалізацію профілю A2DP для прийому стереоаудіо [16]. Ця бібліотека дозволяє приймати потокові аудіодані з мобільного пристрою без додаткових компонентів, використовуючи лише ресурси ESP32. Для обробки аудіосигналу було реалізовано callback-функцію, яка зберігає зразки сигналу в масив для подальшого аналізу. Аудіосигнал потім подається на зовнішній аудіочип через інтерфейс I2S, що було налаштовано за допомогою структури i2s_pin_config_t.

Аналіз отриманого звукового сигналу в частотній області здійснювався за допомогою бібліотеки arduinoFFT. Вона дозволяє виконати швидке перетворення Фур'є (FFT), що перетворює часову вибірку сигналу на частотну [17]. Це дало змогу реалізувати світломузику — аналізуючи амплітуду частотних компонентів, програма визначає, які діоди потрібно засвітити і яким кольором. Отримані значення використовувались для побудови симетричного світлового шаблону, що оновлюється в реальному часі відповідно до музичного сигналу.

Для керування адресними світлодіодами WS2812B використовувалась бібліотека Adafruit_NeoPixel [18]. Вона забезпечує зручне програмне керування кожним світлодіодом у стрічці окремо, дозволяє задавати колір у форматі RGB або HSV, налаштовувати яскравість і використовувати ефекти затухання. Завдяки підтримці широкого спектру функцій ця бібліотека дозволила реалізувати як просту індикацію станів, так і складні анімації, пов’язані з аудіоаналізом.

У частині керування рухом роботизованої платформи Arduino Uno керувала драйвером двигунів TB6612, для чого використовувались стандартні засоби Arduino (analogWrite, digitalWrite). Бібліотека AFMotor.h дозволяє просто керувати напрямком обертання, швидкістю та зупинкою двигунів, абстрагуючи програміста від деталей налаштування ШІМ-сигналів.

Для зчитування даних з інфрачервоних сенсорів лінійного слідування використовувалась бібліотека QTRSensors, яка надає інтерфейс до масиву сенсорів і дозволяє зчитувати як цифрові, так і аналогові значення, а також використовувати калібрування. Цей функціонал забезпечив точне визначення положення чорної лінії відносно центра шасі, що є критично важливим для реалізації навігації.

Усі бібліотеки були підключені через менеджер бібліотек Arduino IDE або встановлені вручну з офіційних репозиторіїв GitHub. Перед використанням кожна бібліотека проходила тестування — перевірялась її працездатність у базових умовах, зокрема правильне налаштування апаратних пінів, відповідність версії плати, конфлікти між модулями.

## 2.3 Програмне керування рухом на базі Arduino Uno

Керування рухом роботизованої платформи реалізовано на базі мікроконтролера Arduino Uno. Уся логіка руху зосереджена у програмному коді, який отримує дані від сенсорів, аналізує їх, формує відповідні команди та передає ці команди на драйвер двигунів. У даній конструкції використовується драйвер TB6612, який забезпечує ефективне керування двома двигунами

постійного струму (рисунок 2.3). Цей драйвер дозволяє керувати напрямком обертання та швидкістю кожного з моторів за допомогою логічних сигналів і широтно-імпульсної модуляції (PWM) [19].

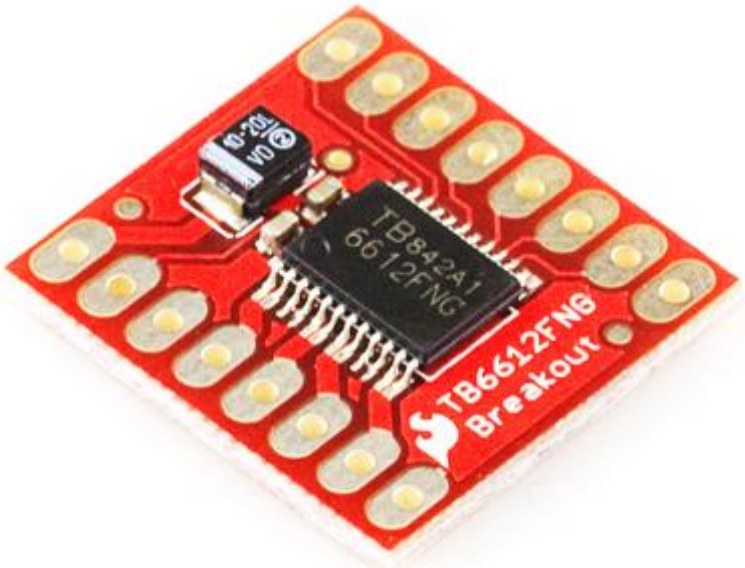


Рисунок 2.3 — Драйвер TB6612

Arduino Uno приймає сигнали з інфрачервоних сенсорів, що реагують на колір поверхні під колесами. Якщо сенсор фіксує чорну лінію, це означає, що робот знаходиться на маршруті. Відповідно до положення чорної лінії щодо центральної осі робота, програма визначає необхідний напрямок руху та виконує корекцію. Це дозволяє реалізувати базову поведінку типу слідування за лінією. Усі обчислення відбуваються на самій платі Arduino, що забезпечує швидку реакцію системи без потреби у зовнішньому аналізі.

Для реалізації обертання коліс Arduino генерує PWM-сигнали на відповідні порти, до яких підключено драйвер. Чим більша ширина імпульсу, тим швидше обертається мотор. Таким чином досягається не лише контроль над напрямком, а й над швидкістю руху. Коли необхідно змінити напрям, змінюється логічний рівень на контрольних пінових входах, що відповідають за полярність напруги на двигуні.

Програмний код також враховує ситуації, коли потрібно зробити поворот або виконати гальмування. Наприклад, якщо обидва крайні сенсори фіксують

чорну лінію, робот розпізнає це як перехрестя або маркер дії й може зупинитись або змінити напрям. Це дозволяє закладати в маршрут умовні точки прийняття рішень. Крім того, система підтримує налаштування швидкості для кожного двигуна окремо, що дає змогу точно повертати, оминати перешкоди або маневрувати на обмеженому просторі.

Arduino Uno виступає в цьому випадку як виконавчий модуль, який отримує інформацію ззовні (сенсори, інші мікроконтролери), приймає рішення в реальному часі та формує керуючі сигнали. Такий підхід дозволяє забезпечити повну автономність системи — робот здатен реагувати на зміну навколишнього середовища без втручання користувача.

## 2.4 Робота з ESP32-CAM

Модуль ESP32-CAM виконує функції бездротової передачі даних, обробки зображень та прийому зовнішніх команд (рисунок 2.4). Його апаратна архітектура поєднує високопродуктивний двоядерний мікроконтролер ESP32 із вбудованим модулем Wi-Fi та камерою OV2640, що робить його особливо придатним для реалізації задач відеоспостереження, дистанційного контролю та зв’язку з іншими елементами системи [20].

Рисунок 2.4 — Камера OV2640

Однією з ключових функцій модуля є створення Wi-Fi-з’єднання. У межах реалізованого проєкту ESP32-CAM може як підключатися до наявної локальної мережі, так і працювати в режимі точки доступу (Access Point), створюючи власну Wi-Fi-мережу. Це дає змогу організувати взаємодію між модулем та користувачем, наприклад — через браузер. У такому режимі користувач може підключитися до ESP32-CAM зі смартфона або комп’ютера та отримати доступ до веб-інтерфейсу, який відображає відеопотік із камери в реальному часі. Такий підхід не потребує додаткових мобільних застосунків і забезпечує простий і зручний канал керування.

Програмна частина модуля реалізується у середовищі Arduino IDE. З використанням відповідних бібліотек (esp_camera.h, WiFi.h, WebServer.h) налаштовується конфігурація камери, параметри Wi-Fi з’єднання, а також логіка генерації веб-інтерфейсу. У прошивці передбачено обробку HTTP-запитів, які ініціюються з боку користувача, що дозволяє не лише переглядати відео, а й надсилати команди, наприклад, змінювати режим роботи або ініціювати певну дію на стороні іншого модуля.

Крім трансляції відеопотоку, ESP32-CAM також виконує обробку запитів, які надходять через UART-інтерфейс від Arduino Uno або інших мікроконтролерів. Наприклад, Arduino може надсилати команду, що активує зйомку зображення, або запускає певний веб-сервіс. У таких випадках ESP32-CAM виступає як слухач UART-каналу і реагує на отримані дані згідно з логікою, закладеною в прошивці. Це дозволяє реалізувати синхронізацію між окремими модулями в межах єдиної логіки роботи пристрою.

Формат передавання зображень може бути як потоковим (MJPEG або JPEG-статичні кадри), так і покадровим з локальним збереженням у пам’яті контролера або передачею через веб-інтерфейс. За потреби, прошивка може бути доповнена можливістю збереження знімків на карту microSD (за наявності розширення плати) або передачею зображень у вигляді масиву байтів через HTTP-запити.

Окреме місце у логіці роботи ESP32-CAM займає обробка команд, які надходять із браузера користувача. Ці команди можуть надсилатися через кнопки у веб-інтерфейсі або спеціальні GET-запити. Наприклад, натискання кнопки «вперед» може викликати внутрішню функцію, яка надсилає відповідну команду по UART на Arduino Uno, що відповідає за рух. Таким чином, модуль відіграє роль посередника між користувачем і апаратною частиною робота.

Використання ESP32-CAM значно розширює функціональні можливості системи, оскільки дозволяє не лише передавати зображення, а й забезпечити двосторонній зв’язок — як у вигляді команд керування, так і зворотного відеозв’язку. Це створює основу для реалізації гнучкого інтерфейсу дистанційного контролю та візуального супроводу роботи автономного робота, що відповідає сучасним принципам побудови інтерактивних кіберфізичних систем.

## 2.5 Обробка аудіосигналу

Важливою складовою мультимедійної частини кіберфізичної системи є модуль відтворення звуку, який реалізується на основі ESP32 LOLIN D32 у поєднанні з цифровим аудіопідсилювачем MAX98357 (рисунок 2.5) .


Рисунок 2.5 — Аудіопідсилювач MAX98357

Даний підсилювач перетворює потік аудіоданих у аналоговий сигнал і подає його на динамік [21]. Це дозволяє відтворювати попередньо збережені аудіофайли (WAV або MP3), які зчитуються з флеш-пам’яті або microSD-карти. Для реалізації цієї функції використовуються Arduino-сумісні бібліотеки (ESP32-audioI2S, Audio.h), які забезпечують потокове декодування аудіо та передавання сигналу на I2S-інтерфейс. У налаштуваннях визначаються пін даних, тактовий пін і формат (моно/стерео).

Альтернативним і водночас більш універсальним способом є використання Bluetooth для бездротового аудіопідключення. ESP32 LOLIN D32 підтримує Bluetooth A2DP (Advanced Audio Distribution Profile), що дозволяє приймати аудіосигнал безпосередньо з мобільного пристрою — смартфона чи планшета. Таким чином, користувач може легко під’єднатися до робота, як до звичайної Bluetooth-колонки, і відтворювати улюблену музику в реальному часі. У прошивці ESP32 використовується бібліотека ESP32-A2DP, яка дозволяє модулю виступати у ролі Bluetooth-ресивера, обробляючи аудіопотік і передаючи його або на MAX98357 через I2S. Приклад коду наведено в лістингу 2.1. Повний код наведено в додатку Г.

Лістинг 2.1 — Обробка аудіосигналу

```
BluetoothA2DPSink a2dp_sink;
a2dp_sink.set_stream_reader(audio_data_callback);
a2dp_sink.start("BTVisualizer");
void audio_data_callback(const uint8_t *data, uint32_t len) {
  for (uint32_t i = 0; i + 3 < len && sampleIndex < SAMPLES; i += 4) {
    int16_t sample = (int16_t)(data[i] | (data[i + 1] << 8));
    vReal[sampleIndex] = (double)sample;
    vImag[sampleIndex] = 0.0;
    sampleIndex++;
    if (sampleIndex >= SAMPLES) readyToAnalyze = true;
```

Така реалізація відкриває широкі можливості: користувач може змінювати треки, регулювати гучність, запускати та зупиняти відтворення прямо з телефону без необхідності перезавантаження або повторного з’єднання. З технічного боку,

Bluetooth-з’єднання конфігурується як стандартний профіль A2DP Sink, а ESP32 при запуску транслює своє ім’я в ефір, після чого очікує на підключення.

Крім програвання музики, модуль ESP32 також може використовуватись для звукової індикації подій, таких як запуск системи, зміна режиму руху або досягнення цільової точки маршруту. Ці сигнали можуть запускатись як автоматично, згідно з логікою прошивки, так і за запитом від інших мікроконтролерів — наприклад, Arduino Uno передає через UART команду на відтворення певного звуку у відповідь на спрацьовування сенсора або проходження маркера.

У перспективі така аудіопідсистема може бути розширена: наприклад, реалізація розпізнавання голосових команд, синтез мови або передавання звуку на інші Bluetooth-пристрої. Архітектура ESP32 і його сумісність з численними аудіо бібліотеками відкриває широкий простір для подальших удосконалень.

Таким чином, ESP32 LOLIN D32 у даному проєкті виступає як гнучкий мультимедійний центр, здатний працювати як з проводовими I2S-пристроями, так і з бездротовими джерелами звуку через Bluetooth, що значно підвищує інтерактивність і зручність користування роботизованою системою.

## 2.6 Керування засобами індикації

Світлодіоди відіграють не лише декоративну роль, а й виконують функції сигнальної та статусної індикації, яка дозволяє відстежувати стан системи, режим роботи або напрямок руху. Для реалізації такої індикації в проєкті використовується програмоване світлодіодне оснащення на базі WS2812B — популярних RGB-світлодіодів, що підтримують індивідуальне керування кожним елементом через єдиний сигнальний пін [22]. WS2812B є інтегрованими адресованими світлодіодами, які об’єднуються в стрічку або матрицю і дозволяють задавати колір та яскравість кожного пікселя окремо (рисунок 2.6). Такий підхід значно розширює можливості для створення динамічних візуальних ефектів, що реагують на зміни в роботі пристрою. Наприклад, при зміні напрямку руху стрічка може змінювати колір або напрям візуальної анімації, при

зупинці — гаснути або пульсувати, а при виникненні помилки — блимати червоним кольором.

Рисунок 2.6 — Світлодіодна стрічка WS2812B

У програмному забезпеченні, що працює на ESP32 LOLIN D32, керування світлодіодами реалізується за допомогою спеціалізованих бібліотек, зокрема Adafruit NeoPixel або FastLED. Ці бібліотеки дозволяють легко ініціалізувати масив світлодіодів, змінювати їхні значення у циклі або за подією, створювати заздалегідь задані анімації, а також синхронізувати візуальні ефекти з іншими частинами системи — наприклад, зі звуковим супроводом або станом руху.

Керуючий мікроконтролер отримує сигнали від Arduino Uno або ESP32-CAM через UART або Wi-Fi, після чого, згідно з логікою програми, запускає потрібну індикацію. Наприклад, у момент запуску системи світлодіоди можуть поетапно загорятися в зеленому кольорі, що символізує готовність до роботи. У процесі руху відображення напрямку може реалізовуватись через зміщення анімації вліво або вправо, а при виконанні маневру — миттєво змінювати колір. У разі переходу в ручний режим, відсутності зв’язку або виникнення збоїв світлодіоди можуть подавати миготливий червоний сигнал.

Один з можливих варіантів застосування — реалізація світломузики, де індикація змінюється відповідно до частоти або гучності звуку, що відтворюється через ESP32. У такому випадку кольори або інтенсивність світіння регулюються в режимі реального часу, створюючи динамічний візуальний супровід до звукових ефектів.

Лістинг 2.2 — Налаштування роботи світлодіодів

```
uint16_t hue = map(band, 0, (NUM_LEDS / 2) - 1, 21845, 0);
uint32_t color = leds.ColorHSV(hue, 255, brightness);
color = leds.gamma32(color);
leds.setPixelColor(led1, color);
leds.setPixelColor(led2, color);
leds.show();
```

Перевагою WS2812B є те, що для керування навіть великою кількістю світлодіодів потрібен лише один цифровий вихід. Це дозволяє зберегти інші порти контролера для інших задач. Крім того, система дозволяє легко масштабувати індикацію — додавати нові елементи, змінювати кольорову логіку або адаптувати візуальні сигнали до нових режимів без змін у апаратній частині.

## 2.7 Реалізація взаємодії між підсистемами

Для забезпечення узгодженої роботи всіх функціональних модулів системи — включаючи керування рухом, обробку візуальних даних, світлову та звукову індикацію — необхідна надійна взаємодія між мікроконтролерами. У цьому проєкті для обміну інформацією між підсистемами використовується кілька способів передачі даних, серед яких центральне місце займає послідовний інтерфейс UART.

Модуль Arduino Uno, відповідальний за зчитування даних із сенсорів і керування рухом, взаємодіє з ESP32-CAM та ESP32 LOLIN D32 через UART, передаючи їм повідомлення про стан системи, маркери маршруту або запити на виконання певних дій. Наприклад, у разі проходження особливого маркера Arduino може надіслати сигнал до ESP32-CAM з інструкцією зафіксувати зображення або передати повідомлення оператору. Цей обмін даними

реалізується за допомогою простого текстового або байтового протоколу, який передбачає команду та параметри, що полегшує розбір отриманого повідомлення з боку приймача.

З іншого боку, модуль ESP32 LOLIN D32, що відповідає за звукове відтворення та динамічну світлову індикацію (наприклад, світломузику), отримує від Arduino або ESP32-CAM сигнали про зміну режимів, події або стан системи. Наприклад, при запуску руху або проходженні ключової точки маршруту система може передавати команду на вмикання анімації на світлодіодній стрічці або програвання аудіосигналу. Для цього також використовується UART або інші допоміжні інтерфейси зв’язку.

Щоб забезпечити коректну синхронізацію між підсистемами, важливо дотримуватися чіткого протоколу взаємодії. Зокрема, всі пристрої повинні використовувати однакову швидкість передачі (baud rate), однакову кількість бітів у кадрі, контроль чіткої послідовності повідомлень [23]. У випадку двосторонньої взаємодії модулі повинні мати змогу обробляти зворотні сигнали підтвердження або відповіді на запити. У більшості сценаріїв для простоти реалізації використовується односпрямований обмін із періодичним оновленням стану або подієвою передачею.

Також важливо враховувати розподіл завдань між підсистемами. Arduino Uno концентрується на низькорівневій обробці сенсорних даних і руху, не перевантажуючи себе мультимедійними задачами. Натомість ESP32-модулі виконують ресурсоємні функції, пов’язані із зображенням, звуком, керуванням Wi-Fi, Bluetooth та індикацією. Такий розподіл дозволяє ефективно використовувати апаратні ресурси кожного мікроконтролера та зменшити навантаження на критичні підсистеми.

Інтеграція між модулями забезпечується через чітко продумані точки обміну даними. UART-пари з’єднань фізично реалізуються між RX і TX піннами відповідних мікроконтролерів. При необхідності використовується буферна логіка, програмна обробка переривань або відкладене читання з UART, щоб уникнути втрати даних. У більш просунутих версіях реалізації передбачено

також взаємодію через Wi-Fi або Bluetooth — наприклад, ESP32-CAM може надсилати повідомлення ESP32 LOLIN D32 безпосередньо через локальну мережу, минаючи Arduino.

Результатом такої багаторівневої взаємодії є створення узгодженої кіберфізичної системи, де кожен модуль виконує свою функцію, але всі вони діють синхронно, реагуючи на сигнали та події, які виникають під час руху робота. Такий підхід дозволяє легко розширювати можливості пристрою, додаючи нові модулі, режими чи сценарії без суттєвої перебудови архітектури або перепрошивки всіх контролерів.

# 3 НАЛАШТУВАННЯ, ТЕСТУВАННЯ ТА ОЦІНКА

## 3.1 Тестування алгоритму руху за допомогою маркерів

У процесі реалізації та налагодження системи автономного руху особливу увагу було приділено етапу тестування розробленого алгоритму навігації за допомогою візуального орієнтира у вигляді суцільної чорної лінії. Алгоритм базувався на даних, отриманих із трьох інфрачервоних сенсорів, розташованих у передній частині платформи, які дозволяють визначати положення лінії відносно центра робота. Реалізоване програмне забезпечення на базі Arduino Uno приймає сигнали з сенсорів і формує відповідні команди для драйверів двигунів, коригуючи напрямок руху в режимі реального часу.

Для забезпечення точного, повторюваного та візуально наочного тестування було створено спеціальне демонстраційне середовище у вигляді повноформатного плакату, виготовленого за індивідуальним проєктом. На відміну від тимчасових трас із паперу чи скотчу, плакат є довготривалим рішенням, призначеним для постійного використання в навчальному процесі. Його основою стала векторна схема, розроблена у графічному середовищі Figma, з урахуванням габаритів мобільної платформи, відстані між сенсорами, ширини колії та мінімального радіуса повороту.

Траєкторія була розроблена у вигляді суцільної чіткої чорної лінії з плавними вигинами, без перехресть та розгалужень, що дозволило зосередитись на точності слідування, стабільності сенсорного зчитування та коректності алгоритмів обробки даних. Лінія мала достатній контраст із фоном, що гарантує коректне функціонування рефлекторних сенсорів незалежно від зовнішнього освітлення.

Готовий макет був надрукований у високій якості на матовому ламінованому матеріалі та змонтований на жорстку поверхню для подальшого розміщення у лабораторії кафедри (рисунок 3.1). Таким чином, плакат виконує не лише роль тестового середовища, а й демонстраційного навчального стенду, що може бути використаний у навчальному процесі, для ознайомлення студентів із основами лінійної навігації, сенсорної обробки та автономного управління.


Рисунок 3.1 — Маршрут для тестування руху робота, створений у Figma

У процесі проходження цієї траси робот зчитував положення чорної лінії за допомогою масиву інфрачервоних сенсорів, встановлених у передній частині шасі. Програмне забезпечення, реалізоване на Arduino Uno, аналізувало сигнали сенсорів і формувало команди на зміну напрямку або збереження поточного руху. Плавні вигини траси дозволили перевірити точність корекції курсу та реакцію робота на поступові зміщення лінії від центральної осі.

Завдяки відсутності розгалужень і зосередженості на одній цілісній лінії вдалося детально протестувати саме базову логіку навігації, оцінити чутливість системи до дрібних вигинів та підібрати оптимальні значення широтно-імпульсної модуляції для керування двигунами. Отримані результати засвідчили, що реалізований алгоритм ефективно справляється з підтриманням напрямку руху навіть за умов м'яких змін маршруту, що підтверджує його придатність для подальшого використання в складніших навігаційних сценаріях.

Для точного зчитування положення лінії відносно центра шасі під час тестування використовувалась стандартна лінійка з трьох інфрачервоних рефлекторних сенсорів, розміщених у передній частині робота на однаковій відстані один від одного. Кожен із сенсорів складався з інфрачервоного світлодіода та фототранзистора, який фіксує відбите від поверхні випромінювання (рисунок 3.2). Чорна лінія поглинає інфрачервоне світло, тоді як біла поверхня його відбиває, що дозволяє сенсору генерувати відповідний логічний сигнал.

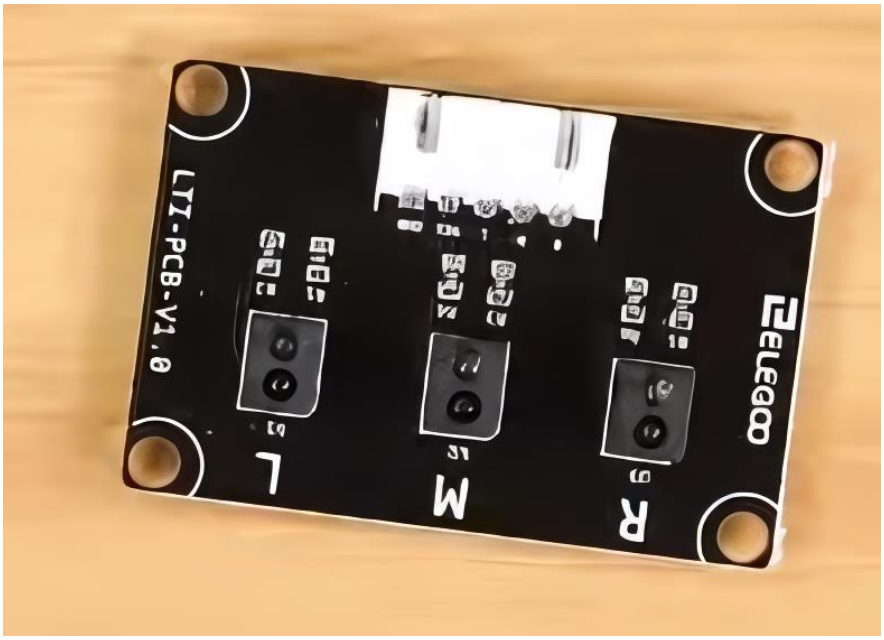


Рисунок 3.2 — Інфрачервоний фотоелектричний датчик

Під час налаштування було обрано цифровий режим зчитування, де кожен сенсор передає 0 або 1 залежно від наявності чорного кольору під ним. Це значно спрощувало обробку даних і забезпечувало швидку реакцію у реальному часі. Для уникнення хибних спрацьовувань було використано програмну фільтрацію, яка виключала короткочасні сигнали тривалістю менше 50 мілісекунд. Також було підібрано оптимальну висоту сенсорів над поверхнею (7 мм), що забезпечило стабільне розрізнення фону й лінії незалежно від зовнішнього освітлення.

Показники з трьох сенсорів формували шаблон, за яким алгоритм визначав напрямок руху. Якщо активним був центральний сенсор — рух залишався прямолінійним. Якщо лінія фіксувалась на лівому або правому сенсорі, Arduino генерувала команду на корекцію траєкторії у відповідний бік. Такий підхід дозволив реалізувати просту та надійну логіку слідування, яка показала стійку роботу на трасі з плавними вигинами. Детальну логіку роботи пошуку лінії наведено на рисунку 3.3.

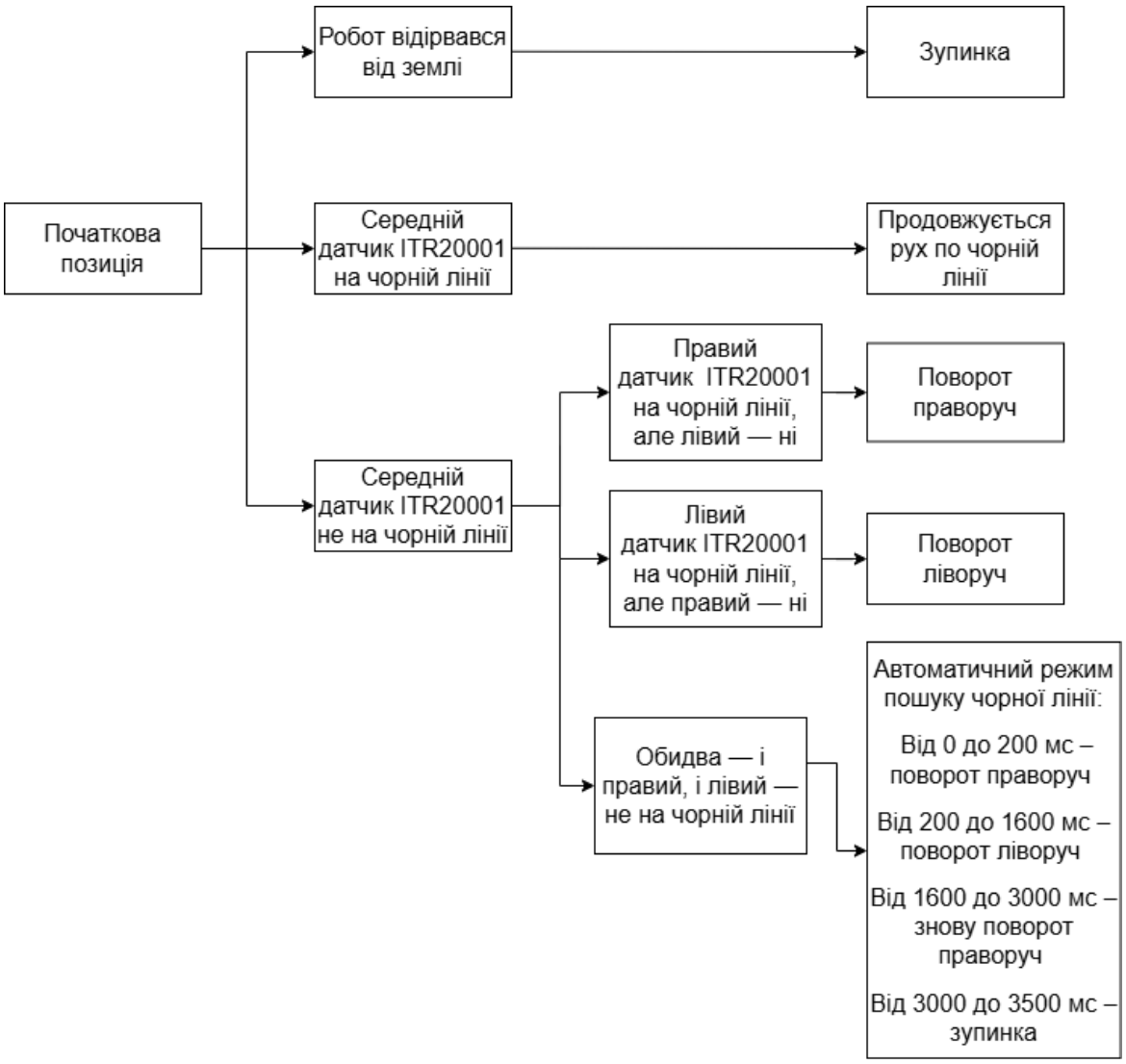


Рисунок 3.3 — Схема логіки роботи пошуку лінії

Ще одним важливим функціональним режимом, реалізованим у даному кіберфізичному комплексі, є автоматичне оминання перешкод. Цей режим дає змогу роботу самостійно реагувати на об’єкти, які опиняються на його шляху, та

змінювати траєкторію руху, щоб уникнути зіткнень. Така поведінка реалізується завдяки використанню ультразвукового датчика відстані, розміщеного у передній частині шасі (рисунок 3.4).


Рисунок 3.4 — Ультразвуковий датчик вимірювання відстані

У процесі роботи система безперервно надсилає короткі ультразвукові імпульси, які відбиваються від навколишніх об’єктів. Отриманий зворотний сигнал дозволяє обчислити відстань до перешкоди, виходячи з часу, що минув між передачею та прийомом імпульсу. Вбудований у прошивку алгоритм постійно порівнює отримані значення з наперед заданим пороговим значенням. У разі перевищення цього порогу система вважає шлях вільним і продовжує рух вперед. Якщо ж перешкода знаходиться ближче, комплекс призупиняється і переходить до виконання маневру об’їзду.

Процедура оминання перешкоди реалізована у вигляді послідовності дій, які можуть включати зупинку, рух назад, обертання вбік і повторну перевірку маршруту. Така логіка дозволяє роботу знайти альтернативний шлях та поновити рух у напрямку основного маршруту (рисунок 3.5). Весь процес відбувається автономно, без участі користувача, що є важливою перевагою при використанні робота в динамічному або невідомому середовищі. Завдяки цьому комплекс здатен адаптуватися до змін у навколишньому середовищі в реальному часі.

Особливо ефективним такий режим є під час експлуатації в обмеженому просторі, де ймовірність зіткнення з перешкодами досить висока.

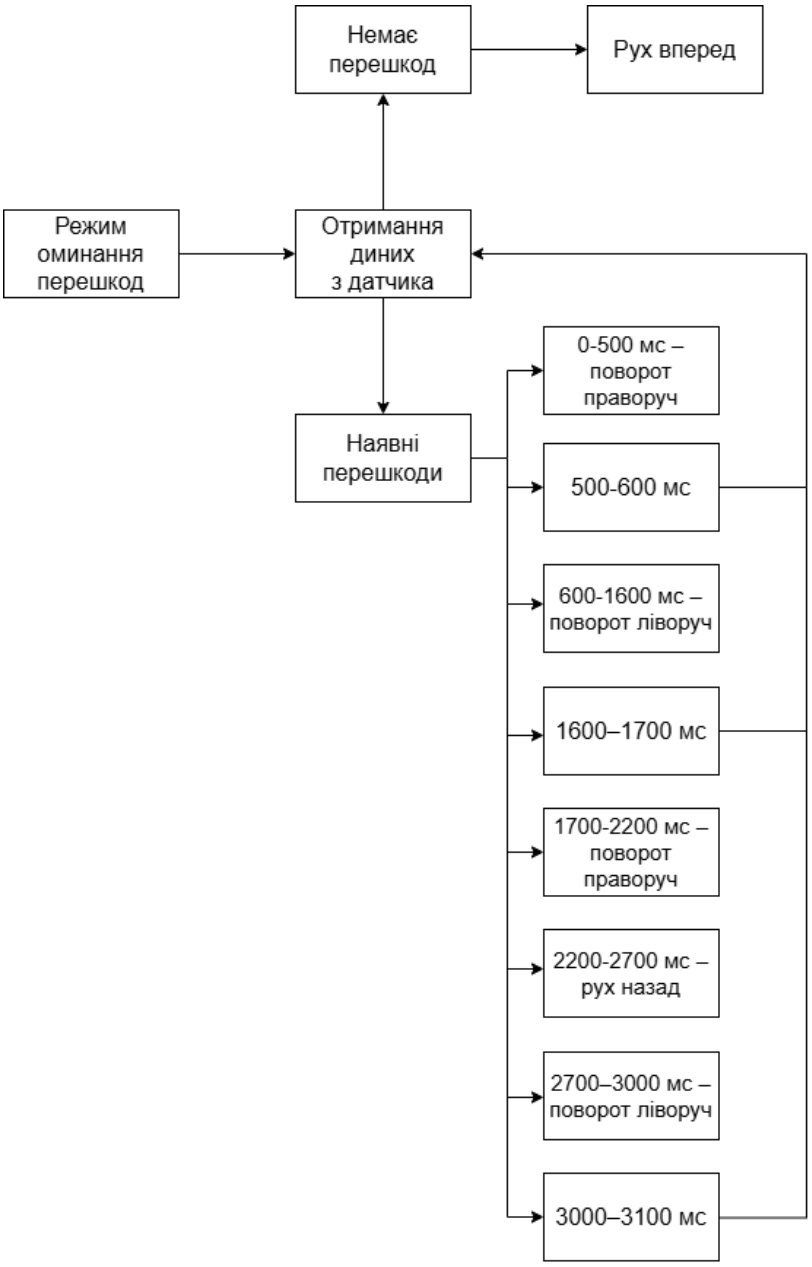


Рисунок 3.5 — Алгоритм оминання перешкод

У проєкті логіка режиму оминання перешкод реалізована у вигляді окремої функціональної процедури, що виконується паралельно з основною логікою навігації. Такий підхід дозволяє комбінувати відстеження лінії або маркерів із динамічним ухиленням від об’єктів, що виникають на шляху. У майбутньому можливе удосконалення цієї системи шляхом поєднання даних з декількох

сенсорів або використання алгоритмів машинного навчання для передбачення руху об’єктів у полі зору.

### 3.2 Перевірка зв’язку та роботи модулів

У реалізованій системі застосовано три окремі мікроконтролерні модулі, кожен із яких виконує специфічні функції: Arduino Uno відповідає за управління основним рухом та обробку даних з сенсорів, ESP32-CAM реалізує відеозв’язок і прийом команд по Wi-Fi, а ESP32 LOLIN D32 забезпечує обробку аудіосигналу та керування світловими ефектами. З огляду на поділ функцій між платами, критично важливим етапом стала перевірка коректності та стабільності обміну даними між цими модулями.

Основним каналом зв’язку між Arduino Uno та модулями на базі ESP32 виступає послідовний інтерфейс UART. Arduino Uno передає сигнали у вигляді текстових або бінарних команд, які зчитуються іншими модулями у відповідних послідовних портах. Наприклад, Arduino може передавати інформацію про зміну режиму, стан сенсорів або команди для активації певної поведінки, які обробляє ESP32 LOLIN D32. Для перевірки цього каналу було реалізовано просту систему запит—відповідь, де кожен модуль після прийому команди надсилав зворотне підтвердження. Це дозволило протестувати як односторонній, так і двосторонній обмін інформацією.

Зв’язок із ESP32-CAM був перевірений як через UART (для команд з Arduino Uno), так і через Wi-Fi, який забезпечував передачу зображення та взаємодію з користувачем. UART-інтерфейс використовувався для надсилання внутрішніх сигналів — наприклад, активації або зупинки відеопотоку. У той час як Wi-Fi-інтерфейс забезпечував підключення до локальної мережі або створення точки доступу для зовнішнього управління через браузер. Для тестування Wi-Fi-зв’язку використовувався браузер з підключенням до IP-адреси ESP32-CAM, де запускалась вбудована вебсторінка. Її функціональність включала доступ до відео, а також надсилання команд через кнопки або HTTP-запити.

Усі плати живляться від одного джерела — літієвої батареї на 7,4 В із вбудованими стабілізаторами живлення, що дозволяє забезпечити узгоджене функціонування без електричних конфліктів на лініях UART. Для підключення UART між модулями було застосовано логічне узгодження рівнів сигналу, оскільки Arduino Uno працює з 5 В логікою, тоді як ESP32 — з 3.3 В.

Окрему увагу було приділено синхронізації комунікації: затримки, що виникають при передаванні даних, були оброблені програмно через використання черг або флагів підтвердження прийому. Це дозволило уникнути ситуацій, коли модуль не встигав опрацювати повідомлення або приймав неповні дані.

За результатами перевірки було підтверджено, що UART-зв’язок стабільно функціонує між Arduino Uno та обома ESP32, без втрати даних або спотворення команд. Wi-Fi-інтерфейс ESP32-CAM також показав стабільну роботу в режимі точки доступу, з достатньою швидкістю оновлення зображення та прийому команд.

Підключення до джерела звуку реалізовано за допомогою бібліотеки BluetoothA2DPSink, яка дозволяє ESP32 приймати стерео-аудіо потік по Bluetooth у форматі A2DP. Для цифрового зчитування звуку використовується інтерфейс I2S, який забезпечує стабільне та точне передавання аудіосигналу до мікроконтролера. У проєкті застосовано власну конфігурацію I2S-пінів, де дані надходять через вхід data_out_num, а обробка відбувається у callback-функції audio_data_callback.

Обробка аудіосигналу виконується шляхом дискретизації вхідного потоку на фіксовану кількість семплів (256 точок) із частотою 44,1 кГц. Для виділення частотних компонентів звуку використано алгоритм швидкого перетворення Фур’є (FFT), реалізований через бібліотеку arduinoFFT. Цей алгоритм дозволяє розкласти складний аудіосигнал на окремі частотні компоненти, тобто визначити, які саме частоти присутні у конкретному моменті часу та з якою інтенсивністю. Такий аналіз критично важливий для реалізації ефекту

світломузики, оскільки дає змогу відобразити не просто загальну гучність, а саме структуру звуку за частотними діапазонами.

У реалізованій системі вхідний Bluetooth-аудіосигнал приймається у цифровому форматі через інтерфейс I2S, після чого кожна порція сигналу (розміром 256 семплів) зберігається у масив vReal[] для обробки. Ці дані представляють собою миттєві значення звукової хвилі (амплітуди в часі). Щоб перевести сигнал у частотну область, на ці значення накладається віконна функція Геммінга, яка зменшує ефект спектрального «витікання» і дозволяє точніше виділити частотні піки.

Після цього виконується сам процес швидкого перетворення Фур'є, який обчислює комплексні спектральні коефіцієнти — кожен із них відповідає за певну частоту в діапазоні від 0 до половини частоти дискретизації (тобто до 22050 Гц при 44,1 кГц). З комплексних значень отримується модуль спектра — амплітуда кожної частотної компоненти. Ці амплітуди далі використовуються для побудови світлової реакції на частоти: наприклад, низькі частоти (бас) активують центральні світлодіоди, середні — ближчі до країв, а високі — крайні.

Особливістю реалізації є поділ спектра на кілька логарифмічних діапазонів (за допомогою функції getFrequencyBand()), що відповідають реальному сприйняттю звуку людиною — оскільки ми розрізняємо частоти не лінійно, а логарифмічно.

Щоб уникнути стрибкоподібного миготіння, в системі також реалізовано згладжування амплітуд за допомогою коефіцієнта альфа, тобто кожне нове значення амплітуди обчислюється як комбінація попереднього значення і нового за принципом експоненційного середнього. Це робить світлову реакцію більш плавною і приємною для сприйняття. Програмну реалізацію згладжування наведено в лістингу 3.1.

Лістинг 3.1 — Реалізація згладжування амплітуд

```
int band = getFrequencyBand(freq);
smoothedAmplitude[band] = alpha * smoothedAmplitude[band] + (1 - alpha) * amplitude;
uint8_t brightness = map((int)smoothedAmplitude[band], 100, 8000, 10, 255);
```

Отриманий спектр поділяється на кілька частотних діапазонів, які відображаються у вигляді симетричних візуальних ефектів на 24 світлодіодах (12 пар зліва і справа). Для створення більш плавної анімації використовується експоненційне згладжування амплітуд (alpha blending), а для колірної палітри — градієнт на основі HSV-моделі кольору. Кожен сегмент світлодіодної стрічки реагує на свою частотну смугу, формуючи ефект динамічного спектру.

Крім цього, у прошивці реалізована фільтрація низькоамплітудних коливань — усі значення нижче за адаптивний поріг (залежно від шумової підлоги) відкидаються. Це дозволяє уникнути хаотичного миготіння світлодіодів при низькому рівні сигналу або на паузах у треку.

Світлова реакція на звук реалізується у дзеркальному вигляді: частотні смуги симетрично відображаються зліва і справа, що створює ефект рівномірного «дихання» світлодіодної стрічки в ритмі музики. Кольори змінюються залежно від частоти — від холодних відтінків для низьких частот до теплих для високих. Яскравість змінюється пропорційно амплітуді звукового сигналу, і додатково обробляється функцією gamma32() для компенсації сприйняття людським оком.

Ця підсистема працює незалежно від інших компонентів робота, але може бути інтегрована з ними для індикації станів або створення мультимедійного супроводу під час руху. Таким чином, реалізоване програмне рішення забезпечує повноцінну аудіо-візуальну інтеракцію на основі живого звукового потоку та підвищує привабливість системи в навчальному або демонстраційному середовищі.

## 3.3 Обробка збоїв у системі

У складних багатокомпонентних системах, до яких належать кіберфізичні платформи, особливу увагу необхідно приділити стабільності роботи та здатності системи відновлюватися після збоїв. З огляду на це, в межах розробки програмного забезпечення було передбачено базові механізми резервування функцій і відновлення критичних процесів у випадку нештатної ситуації.

Оскільки система включає три основні мікроконтролери, частина функціональності між ними розподілена з можливістю часткового дублювання. Наприклад, основне керування рухом реалізується на Arduino Uno, але при втраті з’єднання або відсутності відповіді від цієї плати інші модулі можуть реагувати на затримку сигналу та припиняти активність — наприклад, зупинити передачу відео, вимкнути світлодіоди або згенерувати сигнал тривоги. Така логіка реалізується за допомогою таймерів, контрольних фреймів і станів.

Ще одним типом збоїв є порушення комунікації між модулями. У таких випадках використовується логіка повторного з'єднання. При втраті Wi-Fi-зв’язку ESP32-CAM може автоматично перезапустити мережу або переключитись у режим точки доступу. Для UART-каналу, якщо протягом певного інтервалу часу не отримано жодної команди, система інтерпретує це як втрату зв’язку й ініціює безпечний стан — зупинка двигунів, вимкнення активних режимів та очікування на відновлення.

У прошивці Arduino Uno реалізована перевірка коректності вхідних даних із сенсорів. Якщо датчики повертають неконсистентні або недопустимі значення (наприклад, сигнал відсутній на всіх сенсорах одночасно), програма не приймає миттєвих рішень, а вводить коротку затримку, виконує повторне зчитування і лише потім переходить до дій. Така проста перевірка дозволяє уникнути помилок у навігації через випадкові збої або короткочасні електричні перешкоди.

У випадках зупинки або «зависання» системи з боку користувача передбачено можливість перезапуску вручну, наприклад, через фізичну кнопку RESET або подачу спеціальної команди через UART чи веб-інтерфейс. Це дозволяє оперативно повернути робота до стабільного стану без необхідності повного вимкнення живлення.

Для збереження базової працездатності після короткочасного збою або відключення живлення деякі параметри, як-от останній відомий режим роботи, можуть бути збережені у енергонезалежній пам’яті. Це дозволяє після відновлення живлення повернутись у той самий режим без необхідності повторної конфігурації або втручання ззовні.

## 3.4 Тестування роботи та рекомендації щодо вдосконалення

Для оцінки роботи кіберфізичної системи було проведено стрес-тестування в умовах тривалого безперервного функціонування. Це тестування дозволило виявити потенційні слабкі місця в апаратній та програмній реалізації, оцінити поведінку системи за межами звичайного сценарію використання, а також перевірити стійкість компонентів до перегріву, просідання напруги, втрати зв’язку та накопичення помилок у пам’яті.

Тестування проводилось у режимі автономного слідування за лінією, під час якого робот безперервно переміщався по замкненій трасі протягом кількох годин. У ході випробування всі підсистеми працювали одночасно: інфрачервоні датчики здійснювали зчитування лінії, Arduino Uno обробляв дані й генерував сигнали керування, а ESP32-модулі паралельно підтримували функції зв’язку, обробки звуку, відтворення світлових ефектів та передавання команд.

Також увагу було приділено температурному режиму роботи компонентів, зокрема ESP32, драйвера двигунів TB6612 та елементів живлення. Візуальна та температурна оцінка (за допомогою інфрачервоного термометра) показала, що в межах однієї години безперервної роботи плата не перегрівається критично, а живлення залишалося стабільним. Після цього часу робот продовжував роботу, підключений до зовнішнього джерела живлення, щоб оцінити його поведінку за межами стандартного робочого циклу.

Ще одним аспектом було спостереження за якістю обміну даними через UART та Wi-Fi. У процесі тривалого функціонування перевірялась наявність затримок, втрат пакетів або зависань у передаванні команд. Була реалізована логіка повторної ініціалізації зв’язку у разі його втрати, і протягом тесту жодного критичного збою в обміні інформацією зафіксовано не було.

Оцінка швидкодії показала, що затримка між виявленням зміни траєкторії та реакцією виконавчого механізму не перевищувала кількох мілісекунд, що забезпечує плавний рух. Команди з ESP32 також оброблялись оперативно завдяки ефективному обміну даними та розподілу функцій між модулями.

Проведене тестування програмного забезпечення та окремих функціональних підсистем автономного комплексу дало змогу зробити висновки щодо загальної надійності системи, її стійкості до збоїв, а також ефективності реалізованих алгоритмів навігації, обробки сигналів і мультимедійної взаємодії.

У ході випробувань підтверджено, що система здатна працювати в автономному режимі протягом тривалого часу без критичних помилок чи зависань. Комунікація між модулями відбувалася стабільно, як через інтерфейс UART, так і по Wi-Fi, забезпечуючи ефективний обмін даними. Алгоритми слідування за лінією продемонстрували високу точність, стійкість до незначних дефектів маркерної розмітки та швидку реакцію на зміну положення орієнтира.

Водночас було виявлено кілька потенційних напрямів для подальшого вдосконалення системи:

— удосконалення фільтрації сигналів від інфрачервоних сенсорів для кращої роботи при зміненому освітленні або на складніших маршрутах;

— додавання логіки самодіагностики та автоматичного перезапуску модулів у разі втрати зв’язку або помилок UART;

— можливість збереження даних телеметрії у внутрішній пам’яті ESP32 або на карті microSD для подальшого аналізу;

— розширення функціональності веб-інтерфейсу ESP32-CAM: наприклад, інтерактивне налаштування режимів роботи або вибір аудіо/світлових шаблонів;

— оптимізація використання енергоресурсів, зокрема переведення модулів у режим сну у випадках простою або за відсутності активності.

Таким чином, розроблена система демонструє достатній рівень надійності для використання в навчальних, демонстраційних або експериментальних умовах.

# ВИСНОВКИ

У даній роботі було розроблено кіберфізичну систему для автономної навігації мобільного робота з використанням візуальних маркерів. В основі апаратної частини лежить платформа ELEGOO Smart Robot Car Kit V4.0, з використанням модулів Arduino Uno, ESP32-CAM та ESP32 LOLIN D32, що спільно реалізують функції руху, обробки сенсорної інформації, аудіо- та світлової індикації, а також дистанційної взаємодії.

Було створено ефективне програмне забезпечення, яке забезпечує зчитування лінії за допомогою інфрачервоних сенсорів, формування алгоритму слідування, керування двигунами, а також комунікацію між мікроконтролерами через інтерфейси UART та Wi-Fi. Впроваджено підтримку мультимедійних функцій — відтворення аудіо через Bluetooth та візуалізація спектру на основі швидкого перетворення Фур’є з виведенням на адресні світлодіоди.

У результаті тестування підтверджено стабільну роботу системи при тривалому функціонуванні, високу точність проходження маршруту та швидку реакцію на зовнішні сигнали. Компоненти системи ефективно взаємодіють між собою, що забезпечує узгоджену поведінку всієї платформи в реальному часі.

Таким чином, поставлені у роботі завдання були успішно вирішені, а мета — створення програмно-апаратного рішення для автономного пересування робота за заданим маршрутом із розширеними функціями індикації — досягнута. Отримані результати можуть бути використані для подальших досліджень, розширення функціональності або навчальних демонстрацій у сфері робототехніки та вбудованих систем.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. LIV Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2025). URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/ (дата звернення 01.05.2025).

2. Створення робота, який їздить по лінії. URL: https://arduino.ua/art185-stvorennya-robota-yakii-izdit-po-linii (дата звернення 01.05.2025).

3. Line Following Robot. URL: https://www.instructables.com/LineFollowing-Robot/ (дата звернення 03.05.2025).

4. Detection of ArUco Markers. URL: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html (дата звернення 03.05.2025).

5. Orientation process of acicular oxides in the magnetic tape layer. URL: https://ieeexplore.ieee.org/document/1066873 (дата звернення 04.05.2025).

6. Чим RFID відрізняється від NFC? URL: https://idcard.com.ua/ua/blog/chem-rfid-otlichaetsya-ot-nfc/ (дата звернення 04.05.2025).

7. Adruino UNO Documentation. URL: https://docs.arduino.cc/hardware/uno-rev3/ (дата звернення 01.05.2025).

8. ESP32CAM Development Board. URL: https:media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf (дата звернення 02.05.2025).

9. D32 — WEMOS documentation. URL: https://www.wemos.cc/en/latest/d32/d32.html (дата звернення 02.05.2025).

10. UART протокол. URL: https://itmaster.biz.ua/directory/standarts/uart.html (дата звернення 05.05.2025).

11. I2C інтерфейс. URL: https://itmaster.biz.ua/directory/standarts/i2c.html (дата звернення 05.05.2025).

12. IEEE 802.11. URL: https://uk.wikipedia.org/wiki/IEEE_802.11 (дата звернення 05.05.2025).

13. Bluetooth. URL: https://uk.wikipedia.org/wiki/Bluetooth (дата звернення 06.05.2025).

14. Zigbee 101: посібник для початківців. URL: https://dou.ua/forums/topic/38080/ (дата звернення 06.05.2025).

15. Arduino IDE. URL: https://www.arduino.cc/en/software/ (дата звернення 07.05.2025).

16. ESP32-A2DP. URL: https://github.com/pschatzmann/ESP32A2DP (дата звернення 06.05.2025).

17. Швидке перетворення Фур'є. URL: https://uk.wikipedia.org/wiki/Швидке_перетворення_Фур%27є (дата звернення 07.05.2025).

18. Adafruit NeoPixel. URL: https://github.com/adafruit/Adafruit_NeoPixel (дата звернення 08.05.2025).

19. TB6612FNG Dual Motor Driver Carrier. URL: https://www.pololu.com/product/713 (дата звернення 08.05.2025).

20. ESP32CAM Pinout Reference. URL: https://lastminuteengineers.com/esp32-cam-pinout-reference/ (дата звернення 08.05.2025).

21. Adafruit MAX98357 I2S ClassD Mono Amp. URL: https://cdnlearn.adafruit.com/downloads/pdf/adafruit-max98357-i2s-class-d-mono-amp.pdf (дата звернення 09.05.2025).

22. WS2812B Datasheet. URL: https://cdnshop.adafruit.com/datasheets/WS2812B.pdf (дата звернення 09.05.2025).

23. Baud Rate — an overview. URL: https://www.sciencedirect.com/topics/computer-science/baud-rate (дата звернення 09.05.2025).

# ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ проф.,

д.т.н.. Азаров О.Д.

«21» березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання комплексної бакалаврської кваліфікаційної роботи

«Кіберфізичний комплекс для автономної навігації з використанням маркерів

Частина 2. Програмне забезпечення»

08-54.КБКР.042.00.000.ТЗ

Науковий керівник: к.т.н., доц. каф. ОТ

Богомолов С. В.

Студент групи 2СП-21б

Шатайло В. А.

ВНТУ 2025

1 Підстава для виконання комплексної бакалаврської кваліфікаційної роботи

1.1 Актуальність розробки програмного забезпечення обумовлена потребою у створенні ефективних, гнучких і доступних інструментів для керування автономними системами. Сучасні виклики в освітній та експериментальній робототехніці вимагають програмних рішень, здатних взаємодіяти з різними типами сенсорів, здійснювати обробку даних у реальному часі та забезпечувати зв’язок між підсистемами. Реалізація такого програмного забезпечення на основі відкритих платформ, таких як Arduino та ESP32, дозволяє поєднати простоту, розширюваність і доступність, що є критичним фактором для впровадження кіберфізичних систем у навчальний процес;

1.2 Наказ про затвердження теми КБКР.

2 Мета КБКР і призначення розробки

2.1 Мета роботи — розробка та тестування програмного забезпечення для управління автономним мобільним пристроєм, який рухається за маркерами та виконує супутні функції індикації, звукового супроводу і передачі даних;

2.2 Призначення розробки — забезпечення роботи кіберфізичної системи в реальному середовищі з демонстрацією принципів програмного управління, аналізу сенсорної інформації та взаємодії між кількома мікроконтролерами. Система може використовуватись як освітній стенд або експериментальний зразок для вивчення програмування та основ навігації.

3 Вихідні дані для виконання БКР

3.1 Ознайомлення з сучасними підходами до програмування мікроконтролерів у системах реального часу, вивчення бібліотек Arduino IDE для роботи з інфрачервоними сенсорами, драйверами моторів, LED-індикацією, аудіо та бездротовими інтерфейсами;

3.2 Створення модульної структури програмного коду для окремих

функціональних підсистем: керування рухом, світловою індикацією, обробкою Bluetooth-даних та комунікацією між платами;

3.3 Реалізація алгоритмів автономної навігації за допомогою сенсорних даних, отриманих із системи слідкування за лінією, обробка сигналів у цифровому вигляді;

3.4 Тестування роботи взаємодії між Arduino Uno, ESP32-CAM і LoLin D32, налагодження UART-зв'язку, обробка команд, передача повідомлень;

3.5 Верифікація правильності виконання всіх підсистем програмного забезпечення, внесення змін до логіки роботи або налаштувань бібліотек при потребі.

4 Вимоги до виконання БКР

Основна вимога — створення стабільного, масштабованого та зручного для налагодження програмного забезпечення, яке забезпечує автономну роботу кіберфізичної системи, підтримує реакцію на зміну зовнішніх умов, виконує аналіз сенсорної інформації та надає користувачеві можливість інтерактивного керування або моніторингу через бездротові інтерфейси.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка КБКР, графічні і ілюстративні матеріали, протокол попереднього захисту КБКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до КБКР українською та іноземною мовами, довідка про відповідність оформлення КБКР діючим вимогам.

7 Порядок контролю виконання та захисту КБКР

Виконання етапів графічної та розрахункової документації КБКР

контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

Таблиця А.1 — Етапи БКР

| № етапу | Назва етапу | початок | кінець | Очікувані результати |
|---|---|---|---|---|
| 1 | Аналіз задачі | 01.03.2025 | 15.03.2025 | Огляд теми та існуючих рішень |
| 2 | Публікація результатів досліджень | 15.03.2025 | 21.03.2025 | Тези доповідей |
| 3 | Аналіз апаратної бази для розробки програмного забезпечення | 21.03.2025 | 10.04.2025 | Розділ 1 |
| 4 | Програмна реалізація системи | 10.04.2025 | 25.04.2025 | Розділ 2 |
| 5 | Тестування та налагодження, опис виявлених недоліків і внесених змін | 25.04.2025 | 06.05.2025 | Розділ 3 |
| 6 | Оформлення пояснювальної записки та презентації | 07.05.2025 | 26.05.2025 | ПЗ, графічний матеріал, презентація |

8 Вимоги до оформлювання та порядок виконання КБКР

8.1 При оформлювання БКР використовуються:

– ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

– ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

– міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

– методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія» (освітня програма «Системне програмування»). Кафедра обчислювальної техніки ВНТУ 2024 р.;

– документами на які посилаються у вище вказаних.

8.2 Порядок виконання КБКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

# ДОДАТОК Б

Протокол перевірки навчальної (кваліфікаційної) роботи

Назва роботи:

Кіберфізичний комплекс для автономної навігації з використанням маркерів. Частина 2. Програмне забезпечення

Тип роботи: комплексна бакалаврська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

**Показники звіту подібності StrikePlagiarism**

Оригінальність 99,5% Схожість 0,5%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С. М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою StrikePlagiarism щодо роботи.

Автор роботи _____ Шатайло В. А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Богомолов С. В.
(підпис) (прізвище, ініціали)

# ДОДАТОК В

Графічна частина

КІБЕРФІЗИЧНИЙ КОМПЛЕКС ДЛЯ АВТОНОМНОЇ НАВІГАЦІЇ З ВИКОРИСТАННЯМ МАРКЕРІВ. ЧАСТИНА 2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

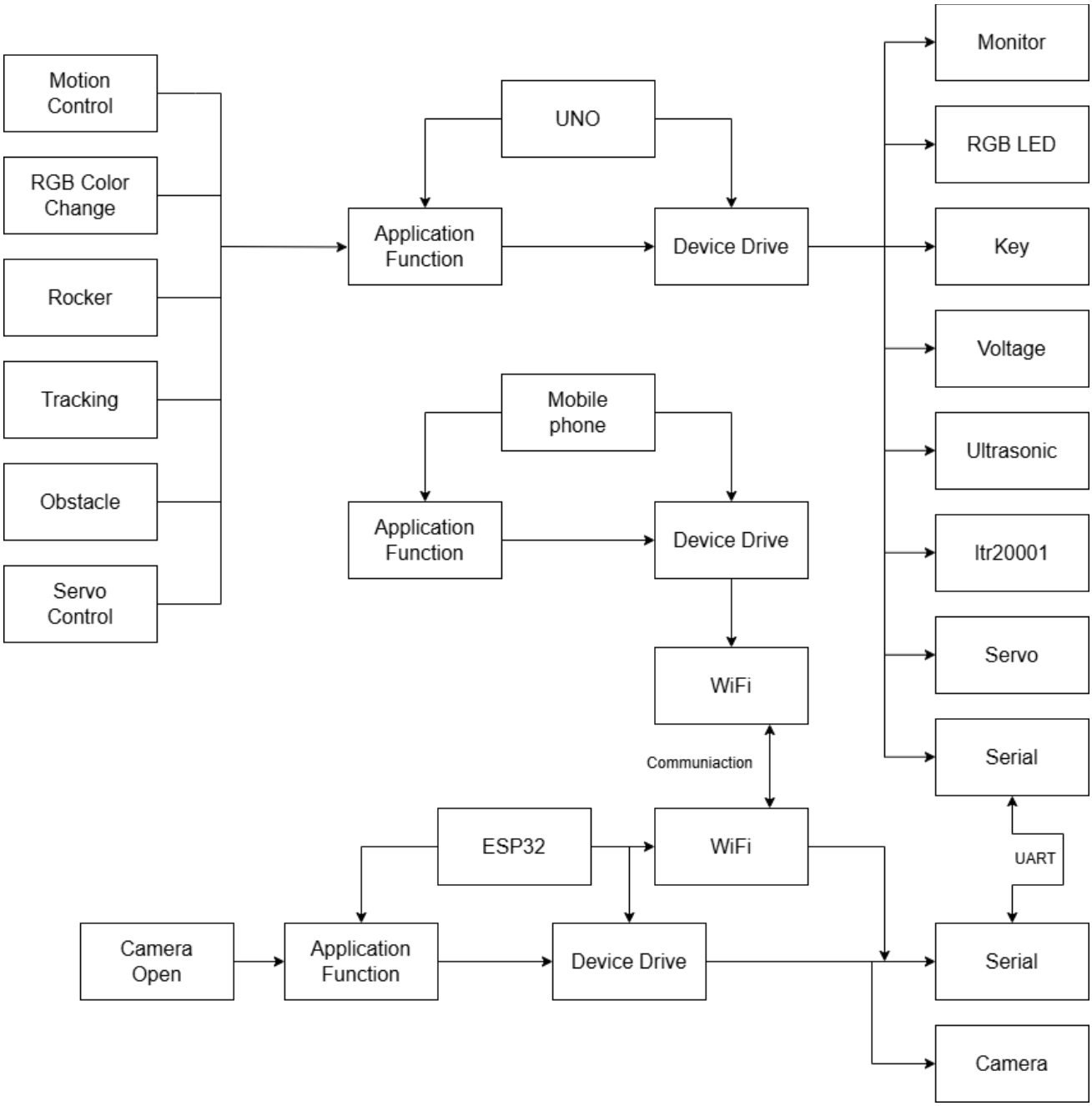


Рисунок В.1 — Структурна схема роботизованого комплексу

# ДОДАТОК Г

Ілюстративна частина

КІБЕРФІЗИЧНИЙ КОМПЛЕКС ДЛЯ АВТОНОМНОЇ НАВІГАЦІЇ З ВИКОРИСТАННЯМ МАРКЕРІВ. ЧАСТИНА 2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

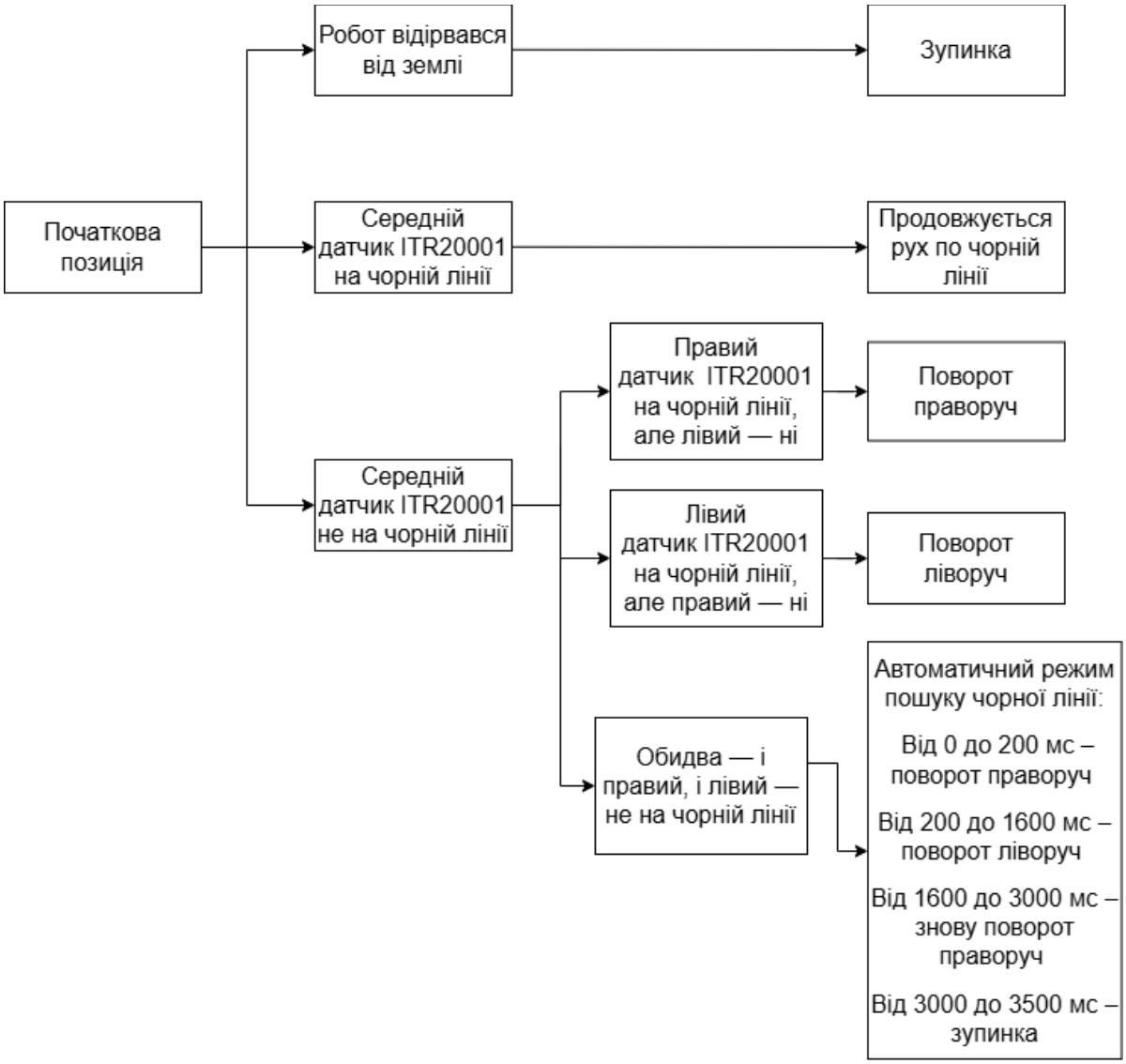


Рисунок Г.1 — Алгоритм роботи пошуку лінії

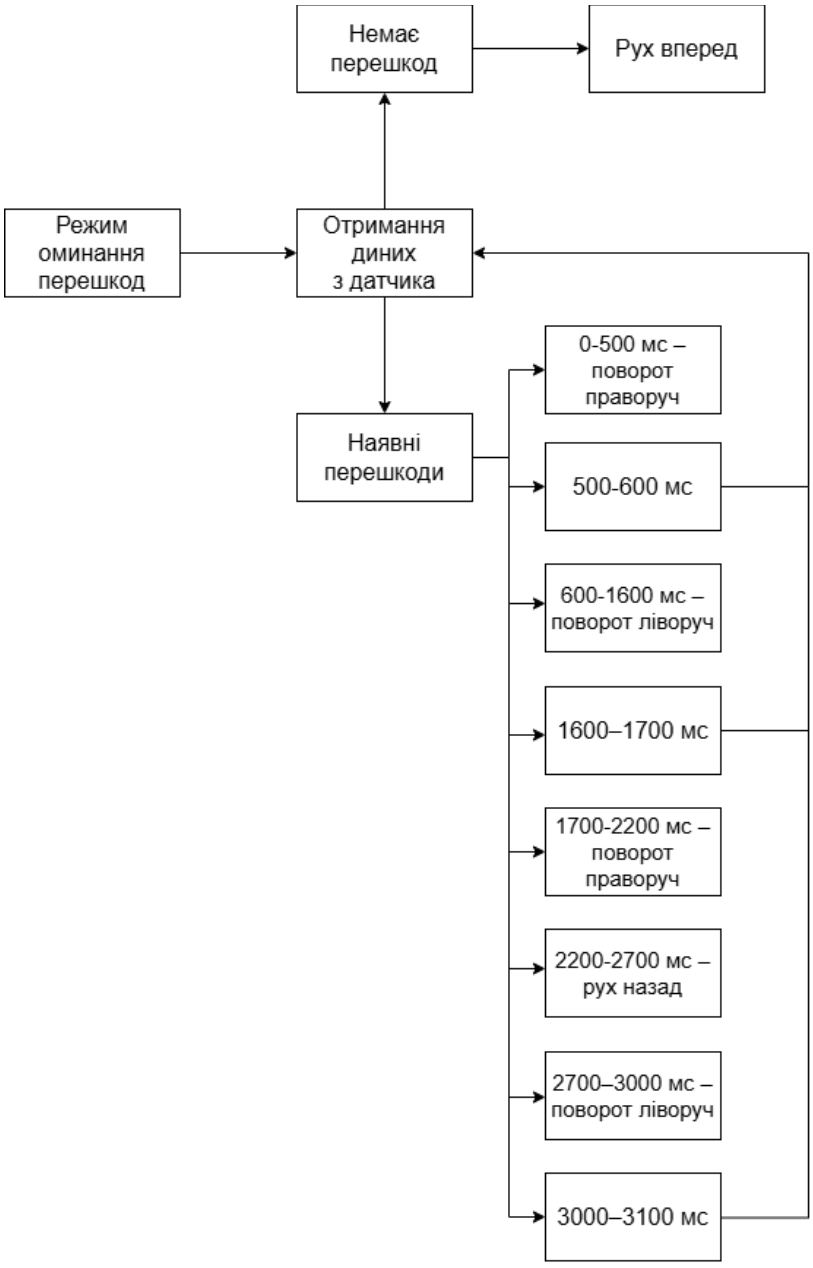


Рисунок Г.2 — Алгоритм роботи оминання перешкод

## ДОДАТОК Д

Лістинг програмного забезпечення

Лістинг файлу main.cpp

```
#include <avr/wdt.h>
#include <hardwareSerial.h>
#include <stdio.h>
#include <string.h>
#include "ApplicationFunctionSet_xxx0.h"
#include "DeviceDriverSet_xxx0.h"
#include "ArduinoJson-v6.11.1.h" //ArduinoJson
#include "MPU6050_getdata.h"
#define _is_print 1
#define _Test_print 0
ApplicationFunctionSet Application_FunctionSet;
MPU6050_getdata AppMPU6050getdata;
DeviceDriverSet_RBGLED AppRBG_LED;
DeviceDriverSet_Key AppKey;
DeviceDriverSet_ITR20001 AppITR20001;
DeviceDriverSet_Voltage AppVoltage;
DeviceDriverSet_Motor AppMotor;
DeviceDriverSet_ULTRASONIC AppULTRASONIC;
DeviceDriverSet_Servo AppServo;
DeviceDriverSet_IRrecv AppIRrecv;
/*f(x) int */
static boolean
function_xxx(long x, long s, long e) //f(x)
{
  if (s <= x && x <= e)
    return true;
  else
    return false;
}
static void
delay_xxx(uint16_t _ms)
{
  wdt_reset();
  for (unsigned long i = 0; i < _ms; i++)
  {
    delay(1);
  }
}
enum SmartRobotCarMotionControl
```

```
{
  Forward,       //(1)
  Backward,      //(2)
  Left,          //(3)
  Right,         //(4)
  LeftForward,   //(5)
  LeftBackward,  //(6)
  RightForward,  //(7)
  RightBackward, //(8)
  stop_it        //(9)
};
enum SmartRobotCarFunctionalModel
{
  Standby_mode,
  TraceBased_mode,
  ObstacleAvoidance_mode
  Follow_mode,
  Rocker_mode,
  CMD_inspect,
  CMD_Programming_mode,
  CMD_ClearAllFunctions_Standby_mode,
  CMD_ClearAllFunctions_Programming_mode,
  CMD_MotorControl,
  CMD_CarControl_TimeLimit,
  CMD_CarControl_NoTimeLimit,
  CMD_MotorControl_Speed,
  CMD_ServoControl,
  CMD_LightingControl_TimeLimit,
  CMD_LightingControl_NoTimeLimit
};
struct Application_xxx
{
  SmartRobotCarMotionControl Motion_Control;
  SmartRobotCarFunctionalModel Functional_Mode;
  unsigned long CMD_CarControl_Millis;
  unsigned long CMD_LightingControl_Millis;
};
Application_xxx Application_SmartRobotCarxxx0;

bool ApplicationFunctionSet_SmartRobotCarLeaveTheGround(void);
void
ApplicationFunctionSet_SmartRobotCarLinearMotionControl(SmartRobotCarMotio
nControl direction, uint8_t directionRecord, uint8_t speed, uint8_t Kp, uint8_t
UpperLimit);
```

```
void
ApplicationFunctionSet_SmartRobotCarMotionControl(SmartRobotCarMotionContr
ol direction, uint8_t is_speed);

void ApplicationFunctionSet::ApplicationFunctionSet_Init(void)
{
  bool res_error = true;
  Serial.begin(9600);
  AppVoltage.DeviceDriverSet_Voltage_Init();
  AppMotor.DeviceDriverSet_Motor_Init();
  AppServo.DeviceDriverSet_Servo_Init(90);
  AppKey.DeviceDriverSet_Key_Init();
  AppRBG_LED.DeviceDriverSet_RBGLED_Init(20);
  AppIRrecv.DeviceDriverSet_IRrecv_Init();
  AppULTRASONIC.DeviceDriverSet_ULTRASONIC_Init();
  AppITR20001.DeviceDriverSet_ITR20001_Init();
  res_error = AppMPU6050getdata.MPU6050_dveInit();
  AppMPU6050getdata.MPU6050_calibration();
  Application_SmartRobotCarxxx0.Functional_Mode = Standby_mode;
}
static bool ApplicationFunctionSet_SmartRobotCarLeaveTheGround(void)
{
  if (AppITR20001.DeviceDriverSet_ITR20001_getAnalogue xxx_R() >
Application_FunctionSet.TrackingDetection_V &&
      AppITR20001.DeviceDriverSet_ITR20001_getAnalogue xxx_M() >
Application_FunctionSet.TrackingDetection_V &&
      AppITR20001.DeviceDriverSet_ITR20001_getAnalogue xxx_L() >
Application_FunctionSet.TrackingDetection_V)
  {
    Application_FunctionSet.Car_LeaveTheGround = false;
    return false;
  }
  else
  {
    Application_FunctionSet.Car_LeaveTheGround = true;
    return true;
  }
}
static void
ApplicationFunctionSet_SmartRobotCarLinearMotionControl(SmartRobotCarMotio
nControl direction, uint8_t directionRecord, uint8_t speed, uint8_t Kp, uint8_t
UpperLimit)
{
  static float Yaw;
  static float yaw_So = 0;
```

```
  static uint8_t en = 110;
  static unsigned long is_time;
  if (en != directionRecord || millis() - is_time > 10)
  {
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_void,
/*speed_A*/ 0,
                                           /*direction_B*/ direction_void, /*speed_B*/ 0,
/*controlED*/ control_enable); //Motor control
    AppMPU6050getdata.MPU6050_dveGetEulerAngles(&Yaw);
    is_time = millis();
  }
  //if (en != directionRecord)
  if (en != directionRecord || Application_FunctionSet.Car_LeaveTheGround ==
false)
  {
    en = directionRecord;
    yaw_So = Yaw;
  }
  int R = (Yaw - yaw_So) * Kp + speed;
  if (R > UpperLimit)
  {
    R = UpperLimit;
  }
  else if (R < 10)
  {
    R = 10;
  }
  int L = (yaw_So - Yaw) * Kp + speed;
  if (L > UpperLimit)
  {
    L = UpperLimit;
  }
  else if (L < 10)
  {
    L = 10;
  }
  if (direction == Forward)
  {
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_just,
/*speed_A*/ R,
                                           /*direction_B*/ direction_just, /*speed_B*/ L,
/*controlED*/ control_enable);
  }
  else if (direction == Backward)
  {
```

```
   AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_back,
/*speed_A*/ L,
                                  /*direction_B*/ direction_back, /*speed_B*/ R,
/*controlED*/ control_enable);
  }
}
static void
ApplicationFunctionSet_SmartRobotCarMotionControl(SmartRobotCarMotionContr
ol direction, uint8_t is_speed)
{
  ApplicationFunctionSet Application_FunctionSet;
  static uint8_t directionRecord = 0;
  uint8_t Kp, UpperLimit;
  uint8_t speed = is_speed;
)
  switch (Application_SmartRobotCarxxx0.Functional_Mode)
  {
  case Rocker_mode:
    Kp = 10;
    UpperLimit = 255;
    break;
  case ObstacleAvoidance_mode:
    Kp = 2;
    UpperLimit = 180;
    break;
  case Follow_mode:
    Kp = 2;
    UpperLimit = 180;
    break;
  case CMD_CarControl_TimeLimit:
    Kp = 2;
    UpperLimit = 180;
    break;
  case CMD_CarControl_NoTimeLimit:
    Kp = 2;
    UpperLimit = 180;
    break;
  default:
    Kp = 10;
    UpperLimit = 255;
    break;
  }
  switch (direction)
  {
```

```
  case /* constant-expression */
      Forward:
    /* code */
    if (Application_SmartRobotCarxxx0.Functional_Mode == TraceBased_mode)
    {
      AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_just,
/*speed_A*/ speed,
                                      /*direction_B*/ direction_just, /*speed_B*/ speed,
/*controlED*/ control_enable); //Motor control
    }
    else
    {
      ApplicationFunctionSet_SmartRobotCarLinearMotionControl(Forward,
directionRecord, speed, Kp, UpperLimit);
      directionRecord = 1;
    }
    break;
  case /* constant-expression */ Backward:
    /* code */
    if (Application_SmartRobotCarxxx0.Functional_Mode == TraceBased_mode)
    {
      AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_back,
/*speed_A*/ speed,
                                       /*direction_B*/ direction_back, /*speed_B*/ speed,
/*controlED*/ control_enable); //Motor control
    }
    else
    {
      ApplicationFunctionSet_SmartRobotCarLinearMotionControl(Backward,
directionRecord, speed, Kp, UpperLimit);
      directionRecord = 2;
    }
    break;
  case /* constant-expression */ Left:
    /* code */
    directionRecord = 3;
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_just,
/*speed_A*/ speed,
                                     /*direction_B*/ direction_back, /*speed_B*/ speed,
/*controlED*/ control_enable); //Motor control
    break;
  case /* constant-expression */ Right:
    /* code */
    directionRecord = 4;
```

```
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_back,
/*speed_A*/ speed,
                                /*direction_B*/ direction_just, /*speed_B*/ speed,
/*controlED*/ control_enable); //Motor control
    break;
  case /* constant-expression */ LeftForward:
    /* code */
    directionRecord = 5;
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_just,
/*speed_A*/ speed,
                                /*direction_B*/ direction_just, /*speed_B*/ speed / 2,
/*controlED*/ control_enable); //Motor control
    break;
  case /* constant-expression */ LeftBackward:
    directionRecord = 6;
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_back,
/*speed_A*/ speed,
                                /*direction_B*/ direction_back, /*speed_B*/ speed / 2,
/*controlED*/ control_enable); //Motor control
    break;
  case /* constant-expression */ RightForward:
    directionRecord = 7;
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_just,
/*speed_A*/ speed / 2,
                                /*direction_B*/ direction_just, /*speed_B*/ speed,
/*controlED*/ control_enable); //Motor control
    break;
  case /* constant-expression */ RightBackward:
    directionRecord = 8;
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_back,
/*speed_A*/ speed / 2,
                                /*direction_B*/ direction_back, /*speed_B*/ speed,
/*controlED*/ control_enable); //Motor control
    break;
  case /* constant-expression */ stop_it:
    directionRecord = 9;
    AppMotor.DeviceDriverSet_Motor_control(/*direction_A*/ direction_void,
/*speed_A*/ 0,
                                /*direction_B*/ direction_void, /*speed_B*/ 0,
/*controlED*/ control_enable); //Motor control

    break;
  default:
    directionRecord = 10;
    break;
```

```
  }
}
void ApplicationFunctionSet::ApplicationFunctionSet_SensorDataUpdate(void)
{
  // AppMotor.DeviceDriverSet_Motor_Test();
  {
    static unsigned long VoltageData_time = 0;
    static int VoltageData_number = 1;
    if (millis() - VoltageData_time > 10) //10ms
    {
      VoltageData_time = millis();
      VoltageData_V = AppVoltage.DeviceDriverSet_Voltage_getAnalogue();
      if (VoltageData_V < VoltageDetection)
      {
        VoltageData_number++;
        if (VoltageData_number == 500)
        {
          VoltageDetectionStatus = true;
          VoltageData_number = 0;
        }
      }
      else
      {
        VoltageDetectionStatus = false;
      }
    }
  }
  // {
  //   AppULTRASONIC.DeviceDriverSet_ULTRASONIC_Get(&UltrasoundData_c
m /*out*/);
  //   UltrasoundDetectionStatus = function_xxx(UltrasoundData_cm, 0,
ObstacleDetection);
  // }
  {
    TrackingData_R =
AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_R();
    TrackingDetectionStatus_R = function_xxx(TrackingData_R,
TrackingDetection_S, TrackingDetection_E);
    TrackingData_M =
AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_M();
    TrackingDetectionStatus_M = function_xxx(TrackingData_M,
TrackingDetection_S, TrackingDetection_E);
    TrackingData_L =
AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_L();
```

```
    TrackingDetectionStatus_L = function_xxx(TrackingData_L,
TrackingDetection_S, TrackingDetection_E);
    ApplicationFunctionSet_SmartRobotCarLeaveTheGround();
  }
```

Лістинг файлу lolind32_sound.cpp

```
#include "BluetoothA2DPSink.h"
#include <arduinoFFT.h>
#include <Adafruit_NeoPixel.h>
#define SAMPLES 256
#define SAMPLING_FREQUENCY 44100
#define LED_PIN     33
#define NUM_LEDS    24
BluetoothA2DPSink a2dp_sink;
Adafruit_NeoPixel leds(NUM_LEDS, LED_PIN, NEO_GRB + NEO_KHZ800);
double vReal[SAMPLES];
double vImag[SAMPLES];
ArduinoFFT<double> FFT = ArduinoFFT<double>(vReal, vImag, SAMPLES,
SAMPLING_FREQUENCY);
volatile uint16_t sampleIndex = 0;
volatile bool readyToAnalyze = false;
double smoothedAmplitude[NUM_LEDS / 2] = {0};
void audio_data_callback(const uint8_t *data, uint32_t len) {
    for (uint32_t i = 0; i + 3 < len && sampleIndex < SAMPLES; i += 4) {
        int16_t sample = (int16_t)(data[i] | (data[i + 1] << 8));
        vReal[sampleIndex] = (double)sample;
        vImag[sampleIndex] = 0.0;
        sampleIndex++;
        if (sampleIndex >= SAMPLES) {
            readyToAnalyze = true;
        }
int getFrequencyBand(double freq) {
    double logFreq = log10(freq);
    int band = map(logFreq * 100, 250, 400, 0, (NUM_LEDS / 2) - 1);
    if (band < 0) band = 0;
    if (band >= NUM_LEDS / 2) band = (NUM_LEDS / 2) - 1;
    return band;
}
void setup() {
    Serial.begin(115200);
    Serial.println("Bluetooth FFT LED Analyzer");
    leds.begin();
    leds.clear();
    leds.show();
```

```
    i2s_pin_config_t my_pin_config = {
        .mck_io_num = I2S_PIN_NO_CHANGE,
        .bck_io_num = 27,
        .ws_io_num = 14,
        .data_out_num = 26,
        .data_in_num = I2S_PIN_NO_CHANGE};
    a2dp_sink.set_pin_config(my_pin_config);
    a2dp_sink.set_stream_reader(audio_data_callback);
    a2dp_sink.start("BTVisualizer")
;}
void loop() {
    if (readyToAnalyze) {
        FFT.windowing(FFTWindow::Hamming, FFTDirection::Forward);
        FFT.compute(FFTDirection::Forward);
        FFT.complexToMagnitude();
        double noiseFloor = 0;
        for (int i = 1; i < SAMPLES / 2; i++) {
            noiseFloor += vReal[i];
        }
        noiseFloor /= (SAMPLES / 2);
        double threshold = noiseFloor * 2;
        for (int i = 0; i < NUM_LEDS; i++) {
            leds.setPixelColor(i, 0); }
        for (uint16_t i = 1; i < (SAMPLES / 2); i++) {
            double freq = (i * 1.0 * SAMPLING_FREQUENCY) / SAMPLES;
            double amplitude = vReal[i];
            if (amplitude < threshold) continue;
            int band = getFrequencyBand(freq);
            double alpha = 0.6;
            smoothedAmplitude[band] = alpha * smoothedAmplitude[band] + (1 - alpha)
* amplitude;
            uint8_t brightness = map((int)smoothedAmplitude[band], 100, 8000, 10,
255);
            if (brightness > 255) brightness = 255;
            uint16_t hue = map(band, 0, (NUM_LEDS / 2) - 1, 21845, 0);
            uint32_t color = leds.ColorHSV(hue, 255, brightness);
            color = leds.gamma32(color);
            int led1 = band;
            int led2 = NUM_LEDS - 1 - band;
            Serial.println(led1);
            leds.setPixelColor(led1, color);
            leds.setPixelColor(led2, color);}
        leds.show();
        sampleIndex = 0;
        readyToAnalyze = false;}
```