Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Модель автомобіля на базі мікроконтролера серії ESP з дистанційною системою керування

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-1
Антон БУГРИМЕНКО
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Роман ЯРОШЕВИЧ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Бугрименку Антону Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель автомобіля на базі мікроконтролера серії ESP з дистанційною системою керування

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи
1. Завдання на кваліфікаційну роботу
2. Перелік вимог до проєктованої системи
3. Методичні вказівки з виконання кваліфікаційної роботи
4. Дані з додаткових джерел
5. Документація до компонентів

4. Перелік питань, що потрібно опрацювати у роботі
1. Аналіз предметної області
2. Розробка структурної схеми
3. Розробка програмного забезпечення (прошивки)
4. Розробка схеми електричного з’єднання
6. Тестування та налагодження
7. Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 11 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз існуючих рішень та постановка	27.05.25-28.05.25	
	завдання		
2	Вибір програмних та апаратних компонентів	29.05.25-30.05.25	
3	Розробка алгоритмів керування та структури	31.05.25-02.06.25	
	програмного забезпечення		
4	Проєктування та реалізація системи	03.06.25-07.06.25	
5	Тестування та відлагодження системи	07.06.25-08.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	09.06.25-12.06.25	
7	Подання кваліфікаційної роботи керівникові та	13.06.25-14.06.25	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	15.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Роман ЯРОШЕВИЧ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 84 с., 30 рис., 1 табл., 2 дод., 11 джерел.

ESP32, BLUETOOTH, РАДІОКЕРОВАНИЙ АВТОМОБІЛЬ, МІКРОКОНТРОЛЕР, КЕРУВАННЯ ЧЕРЕЗ BLUETOOTH, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ.

Метою кваліфікаційної роботи є розробка моделі автомобіля, керованої за допомогою Bluetooth, на основі мікроконтролера серії ESP. Ідея полягає в тому, щоб створити прототип радіокерованого авто, яке буде виконувати команди (рух вперед та назад, повороти) надіслані з додатка на мобільному пристрої. Актуальність теми є досить високою, особливо зараз, коли в країні триває війна. Розробка повітряних і наземних дронів є дуже важливим завданням для розвідувальних, коригувальних, пошуково – рятувальних, а також логістичних операцій. Я вважаю, що ця робота дає можливість отримати практичні навички в розробці систем дистанційного керування, які будуть корисними в майбутньому.

У ході виконання кваліфікаційної роботи було пройдено декілька важливих етапів, від аналізу предметної області до розробки, тестування та налагодження роботи. Під час першого етапу, аналізу предметної області та постановки задачі було проаналізовано наявні рішення та способи реалізації. На етапі проєктування було проаналізовано, які компоненти та програмне забезпечення слід використовувати, а також було розроблено загальну апаратну схему. Третій етап – реалізація моделі, під час якого заплановані компоненти було реалізовано та поєдано у готову, робочу модель, головні функції якої було протестовано задля перевірки функціональності та визначення основних характеристик моделі.

ABSTRACT

Bachelor’s thesis: 84 pages, 30 figures, 1 tables, 2 appendices, 11 sources.

ESP32, BLUETOOTH, RADIO-CONTROLLED CAR, MICROCONTROLLER, BLUETOOTH CONTROL, SOFTWARE, HARDWARE.

The purpose of the qualification work is to develop a Bluetooth-controlled car model based on the ESP series microcontroller. The idea is to create a prototype of a radio-controlled car that will execute commands (forward and reverse, turns) sent from an application on a mobile device. The relevance of the topic is quite high, especially now that the country is at war. The development of air and ground drones is a very important task for reconnaissance, corrective, search and rescue, and logistics operations. I believe that this work gives me the opportunity to gain practical skills in the development of remote control systems that will be useful in the future.

In the course of the qualification work, several important stages were passed, from the analysis of the subject area to the development, testing, and debugging. During the first stage, the analysis of the subject area and the problem statement, we analyzed existing solutions and implementation methods. The design phase analyzed which components and software should be used and developed a general project outline. The third stage is the project implementation, during which the planned components were implemented and combined into a ready – made, working model, the main functions of which were tested to verify the functionality and determine the main characteristics of the model.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ШІМ – широтно-імпульсна модуляція

ADC – аналого-цифровий перетворювач (англ. Analog-to-Digital Converter)

BLE – технологія Bluetooth з низьким енергоспоживанням (англ. Bluetooth Low Energy)

I2C – послідовний інтерфейс передачі даних між інтегральними схемами (англ. Inter-Integrated Circuit)

I2S – стандарт інтерфейсу для передачі цифрових аудіоданих між інтегральними схемами (англ. Inter-IC Sound)

IMU – інерційний вимірювальний блок (англ. Inertial Measurement Unit)

MAC – управління доступом до середовища передачі даних (англ. Media Access Control)

MCU – мікроконтролерний пристрій (англ. Microcontroller Unit)

RC – радіокерований (англ. Radio-Controlled)

SCL – лінія тактового сигналу для послідовних інтерфейсів (англ. Serial Clock)

SDA – лінія даних для послідовних інтерфейсів (англ. Serial Data)

SDK – набір засобів для розробки програмного забезпечення (англ. Software Development Kit)

SPI – послідовний периферійний інтерфейс (англ. Serial Peripheral Interface)

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

UUID – універсальний унікальний ідентифікатор (англ. Universally Unique Identifier)

ВСТУП

У наш час радіоелектроніка проникла у кожну сферу людської діяльності та повсякденного життя. Від виробничих ліній до побутових приладів, від медицини до транспорту – здавалося б, що немає жодної галузі, де вона б не відігравала ключову роль. Кожен з нас, іноді замислюється, як би ми взагалі жили без усіх цих електронних приладів? З кожним роком ми все більше удосконалюємо технології, створюючи прилади та технології, що полегшують життя, роблять його безпечнішим, а також прискорюють величезну кількість процесів, на які б в наших предків пішла б купа часу.

Початком ери дистанційно керованих моделей транспорту можна вважати перші кроки в розробці радіокерованих моделей, але на початку ці моделі були доволі примітивними. Так, на перших етапах для керування моделями використовувалися не точні механічні системи, сам радіокерований транспорт було досить складно придбати, а ціна була досить високою. Перші радіокеровані машини з’явилися в 1960 – х роках і були досить простими пристроями. Але все змінювалося з розвитком електроніки. У 1970 – х і 1980 – х роках радіокеровані машини почали розвиватися стрімкими темпами, компанії, такі як Team Losi, вивели на ринок моделі з поліпшеною підвіскою і потужними двигунами, та, що саме важливо, з новітніми на той час електронними технологіями – для підвищення точності керування моделями почали використовувати стабільніші електронно – механічні джерела опорних коливань, що дозволило робити моделі набагато більш точними та керованими [1].

Розвиток електронних технологій зробив можливим втілювати в моделях більш складні електронні системи, що будуть займати менше місця (тобто збільшувати продуктивність двигуна коштом зменшення ваги моделі в цілому). З’явилися мікросхеми з малою споживаною потужністю. Це дозволило зробити моделі більш компактними та автономними, а новітні

технології у сфері акумуляторних батарей (технології Li – ion та LiPo) дозволили зробити автономність моделей ще більше. Також змінилася й структура системи керування: джерело опорних коливань стало повністю електронним, на основі кварцевих резонаторів. З’явилася можливість більш точного керування двигуном та сервоприводами за допомогою програмованих мікроконтролерів.

У якості головного елементу керування у розроблюваній моделі було вирішено використовувати мікроконтролер, керувати яким можна буде за допомогою розповсюджених стандартів бездротового зв’язку, що пришвидшить розробку, та дозволить більш гнучко розроблювати пристрої для керування моделлю.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність теми

Розробка моделі автомобіля з бездротовим керуванням та мікроконтролером у якості головного компонента є важливою в сучасних умовах. Це пов'язано з розвитком Інтернету речей (IoT) та робототехніки. Існує постійна потреба у бездротовому керуванні пристроями, і такі проєкти дозволяють демонструвати роботу з мікроконтролерами та їх інтеграцію з бездротовими мережами.

Моделі, керовані дистанційно за допомогою мікроконтролерів та бездротових технологій, можуть слугувати навчальним цілям, а також мати практичне застосування, наприклад, для демонстрації базових принципів робототехніки та дистанційного керування. Перехід до різних типів бездротового зв'язку, таких як WI-FI або Bluetooth, розширює спектр можливих застосувань та навчальних аспектів роботи. У перспективі, подібні розробки можуть бути масштабовані та використані для комерційних, промислових або військових цілей.

1.2 Огляд існуючих рішень і технологій

Сучасні проєкти часто базуються на доступних мікроконтролерних платформах, таких як Arduino або ESP8266/ESP32. Ці платформи можуть бути доповнені спеціальними модулями для забезпечення бездротового зв’язку, або ж мати вбудовані бездротові технології, такі як Wi-Fi або Bluetooth.

Існують різні підходи до бездротового керування моделями. Так, традиційні радіокеровані моделі застосовують спеціалізовані RF – модулі. Ці моделі стали першими кроками в розробці дистанційно керованих моделей

транспорту. З розвитком електроніки вони еволюціонували від примітивних механічних систем до більш точних електронних систем керування. У таких моделях джерела опорних коливань на основі кварцевих резонаторів та програмовані мікроконтролери використовуються для більш точного керування двигунами та сервоприводами. Спеціалізовані RF – модулі дозволяють передавати команди від передавача (пульта керування) до приймача на моделі. Прикладами таких моделей є велика кількість комерційних рішень, різноманітних наземних, літальних, та плаваючих дронів, які можуть призначатися для дуже широкого спектра завдань – від дитячих іграшок у формі машинок, квадрокоптерів, вертольотів, катерів та роботів до різноманітних дронів, що призначені для зрошення полів, фільмування відеоматеріалів для кіноіндустрії та навіть для військових цілей.

Сучасні проєкти у галузі робототехніки, що базуються на мікроконтролерах, можуть використовувати Wi-Fi або Bluetooth – зв'язок, що розширює можливості для керування – можна побудувати модель, керування якою може здійснюватися за допомогою саморобних та готових пультів, застосунків на мобільних пристроях, веб додатків. Також перевагою такого підходу є відносно проста реалізація – для вбудованих у мікроконтролери серії ESP модулів Bluetooth та Wi-Fi є готові бібліотеки, які дозволяють швидко писати та налаштовувати код, а також з легкістю передавати дані – це може бути корисно, якщо у моделі є багато сенсорів або камер, інформацію з яких потрібно оброблювати у режимі реального часу. Приклад такого готового рішення можна побачити на рисунку 1.1.

Рисунок 1.1 – Комерційне рішення моделі автомобіля компанії LAFVIN

Для реалізації апаратної частини в подібних проєктах можуть використовуватись досить різні компоненти, це залежить від типу проєкта та функціоналу, який очікується від моделі. Основним компонентом є мікроконтролер (MCU), як головний компонент, що слугує для обробки вхідних сигналів для керування, показників з сенсорів, та для керування моторами, дисплеями, світлодіодами тощо. Також досить важливим компонентом, особливо для моделей автомобілів або літальних дронів, є драйвер двигуна, який оброблює сигнали від мікроконтролера, підсилює та конвертує їх у сигнали, що можуть керувати двигунами напряму. Необхідним компонентом для рухомих моделей є двигуни постійного струму (DC) та сервоприводи. Також, дуже важливим, але не обов'язковим для моделей, які не рухаються у просторі, є джерело живлення, яким можуть виступати акумулятори або батарейки. Сенсори, датчики, і камери є важливими складовими моделей, яким потрібна покращена орієнтація у просторі. Так, наприклад, у літальні моделі можуть вбудовуватися лідари, ультразвукові та інфрачервоні датчики, радари, а необхідним компонентом є IMU – інерційні вимірювальні блоки, що містять у своєму складі акселерометр, гіроскоп, та магнітометр (компас). Наземні ж моделі не потребують обов'язкової наявності сенсорів, але вони можуть сильно покращити орієнтацію моделі у просторі та її функціонал.

1.3 Постановка задачі

Основною метою кваліфікаційної роботи є створення моделі автомобіля на базі мікроконтролера із можливістю керування з мобільного пристрою. В результаті буде отримана модель, яка зможе виконувати базові рухи (рух вперед, назад, повороти), а також отримувати команди від користувача через мобільний застосунок за допомогою технологій бездротового з'єднання. Основними задачами, які буде потрібно виконати під час розробки для досягнення мети є:

- розробка корпусу моделі;
- розробка апаратної частини;
- розробка програмного забезпечення;

Розробка корпусу моделі – це розробка каркаса моделі, у який надалі будуть встановлені всі інші компоненти. Потрібно спроєктувати, як усі необхідні елементи будуть розташовані, та як буде відбуватися їх взаємодія. Корпус має забезпечувати взаємодію електронних компонентів з фізичними, основними з яких є дві віхи коліс – задня, яка буде рухати автомобіль, а також передню, що дозволить автомобілю повертати вправо або вліво. Проєктування цих осей також є важливою задачею, оскільки задня вісь повинна забезпечувати достатню потужність для руху авто та всіх його компонентів, а передня – можливість змінювати напрям руху за допомогою зміни кута повороту коліс.

Розробка апаратної частини – ця частина розробки передбачає вибір електронних компонентів моделі, а також проєктування взаємодії між ними. Основними апаратними компонентами є головний блок керування – мікроконтролер, драйвер для двигуна, двигуни для реалізації роботи осей, джерело живлення та плати для його заряджання та перетворення напруг, а також дисплею, що буде виводити корисну інформацію про поточний стан автомобіля. Потрібно продумати, як ці компоненти будуть взаємодіяти між собою, зробити попередню схему реалізації, а також, продумати, як зробити охайний кабель – менеджмент.

Розробка програмної частини – ця задача передбачає розробку необхідної для мікроконтролера прошивки, що буде реалізовувати функції бездротового з’єднання, контролю двигунів, а також вивід корисної інформації на дисплей та у Serial monitor для налагодження. Другою частиною цієї задачі є розробка мобільного додатку, за допомогою якого й буде виконуватися дистанційне керування – потрібно продумати, як буде виглядати інтерфейс користувача, яким чином буде реалізовано з’єднання та як будуть передаватися команди на мікроконтролер.

2 ПРОЄКТУВАННЯ СИСТЕМИ МОДЕЛІ АВТОМОБІЛЯ

2.1 Проєктування корпусу

Корпус має реалізовувати функції утримання усіх компонентів разом, а також переміщення моделі у просторі, що і є однією з основних ідей розроблюваної моделі.

Є два основних шляху реалізації, які були взяті до уваги – виготовлення деталей засобами 3D друку, або використання конструктору Lego. Ці варіанти відрізняються ціною, зовнішніми виглядом, а також можливістю додавання нових функцій, модернізації та гнучкістю розробки моделі у майбутньому.

Використання 3D принтеру для створення деталей корпусу – найкращий варіант з погляду того, як буде виглядати фінальний результат, але він вимагає чіткого розуміння того, як модель буде виглядати у кінці, бо інакше неправильно спроєктовану деталь доведеться викинути. Також постає питання ціни у вигляді наявності потрібного обладнання та матеріалів. Оскільки не було можливості скористатися засобами 3D друку, єдиним варіантом залишається робити деталі на замовлення, що було б ще дорожче. Отже, було вирішено перейти до другого варіанту.

Для побудови корпусу було вирішено обрати конструктор Lego, тому що він ідеально підходить для створення будь – якої моделі, будь – яких форм, розмірів та рівня складності, а також дозволяє швидко додавати нові функції, модернізувати та гнучко підлаштовувати фізичні параметри моделі під конкретні вимоги (у випадку подальшої модернізації можна буде з відносною легкістю додавати нові компоненти, такі як сенсори або камери, або ж замінювати ті, що вже існують) [2]. Планується використовувати різні деталі, як зі звичайних наборів, так й з Lego Technic – вони використовуються для складніших конструкцій, що містять в собі багато

рухомих частин (наприклад, на рисунку 2.1 можна побачити субмарину, в якої рухаються мотори, а також клешні для захоплення предметів), що ідеально підходить під концепцію розробки роботизованих систем – рухомими частинами будуть передня вісь, яка повинна повертати колеса у потрібному напрямку, та задня, що повинна мати у своєму складі редуктор, який дозволить конвертувати швидкість обертів мотора у потужність для більшої вантажності. Деталі ж зі звичайних наборів краще підходять для закріплення на них апаратного обладнання, що дозволить з легкістю їх знімати для подальшої відладки, модернізації, або ж заміни на нові, кращі компоненти. Для прикріплення апаратного обладнання до деталей Lego використовується термоклей.

Рисунок 2.1 – Набор Lego Technic Глибоководний підводний човен

Також варто зазначити, що для розробки осей будуть використовуватися компоненти зі старих DVD – дисководів, бо наявні у них шківи досить добре підходять для передачі крутного моменту від мотора до деталей Lego, що дозволяє поєднати несумісні компоненти.

2.2 Проєктування апаратної частини

2.2.1 Мікроконтролер

Головним компонентом апаратної частини є мікроконтролер. Головними ж критеріями вибору мікроконтролера є енергоспоживання, потужність вбудованого процесора, розмір програмної пам’яті та кількість пінів з погляду потенціалу подальшого розвитку моделі, наявність вбудованих бездротових технологій для полегшення розробки, а також ціна. Отже, розглядалися три основних варіанти – Arduino Nano, Espressif ESP8266 та Espressif ESP32.

Arduino – це класика розробки саморобних проєктів. Саме з цим сімейством мікроконтролерів асоціюється розробка роботизованих систем та систем розумного будинку. Однак, для реалізації сучасних проєктів, що вимагають високої обчислювальної потужності та інтегрованих бездротових комунікацій, Arduino, попри свою історичну роль та доступність, у контексті даних критеріїв вважається застарілим. Його процесор має обмежену потужність, а відсутність вбудованих модулів бездротового зв'язку (таких як Wi-Fi або Bluetooth) вимагає використання зовнішніх модулів, що збільшує складність апаратної реалізації та загальну вартість системи. Таким чином, Arduino не відповідає ключовим критеріям вибору для даної роботи. [3]

Наступними розглядалися мікроконтролери компанії Espressif – ESP8266 та ESP32. ESP8266 був першим поколінням мікроконтролерів серії ESP та здобув широке визнання завдяки інтегрованому модулю Wi-Fi, що значно спрощує реалізацію мережевих функцій у галузі розробки роботизованих систем та систем розумного будинку з погляду керування та комунікації. Саме його було використано у попередній версії моделі. Однак, було вирішено, що керування за допомогою Bluetooth є більш доречним, оскільки це – оптимальне рішення з погляду енергоспоживання, простоти реалізації та достатності функціоналу для базового керування моделлю

автомобіля без камер і датчиків. В якості третього та, виходячи з вищенаведених міркувань, найбільш відповідного варіанту, розглядався Espressif ESP32. Цей мікроконтролер є наступним поколінням від Espressif та пропонує значно розширені можливості порівняно з ESP8266. ESP32 оснащений потужнішим двоядерним процесором, має більший обсяг пам'яті, а також має істотно більшу кількість GPIO пінів, що забезпечує значну гнучкість для підключення численних периферійних пристроїв та датчиків, а також надає широкий потенціал для подальшого розвитку та розширення функціоналу моделі. Крім того, ESP32 має вбудовані модулі як Wi-Fi, так і Bluetooth, що відповідає вимогам роботи щодо бездротового керування, а також надає додаткові можливості для подальшого розширення функціоналу – додавання камер та сенсорів, інформацію з яких буде більш доречно передавати за допомогою технології Wi-Fi. Вартість ESP32 не є суттєво вищою за ESP8266, що робить його привабливим вибором з погляду співвідношення ціни, продуктивності, та потенціалу для розвитку. Завдяки своїм розширеним можливостям та гнучкості, ESP32 виглядає найбільш перспективним варіантом для цієї роботи. Детальніше порівняння цих двох моделей можна побачити у таблиці 2.1. [4]

Таблиця 2.1 – Порівняння двох поколінь мікроконтролерів

Характеристика	ESP8266	ESP32
MCU	Xtensa Single – core 32 – bit L106	Xtensa Dual – core 32 – bit LX6 with 600 DMIPS
802.11 b/g/n Wi-Fi	HT20	HT40
SPI/I2C/I2S/UART	Не підтримується	Підтримується
GPIO	17	36
Bluetooth	Не підтримується	Підтримується (Bluetooth 4.2 and BLE)

Продовження таблиці 2.1

Апаратний/Програмний PWM	Не підтримується	Підтримується
ADC	Не підтримується	Підтримується
CAN	Не підтримується	Підтримується
Інтерфейс Ethernet MAC	Не підтримується	Підтримується
Сенсор дотику	Не підтримується	Підтримується
Сенсор температури	Не підтримується	Підтримується
Датчик Холла	Не підтримується	Підтримується
Типова частота	80 МГц	160 МГц
SRAM	Не підтримується	Підтримується
Flash	Не підтримується	Підтримується
Робоча температура	– 40°C до 125°C	– 40°C до 125°C

2.2.2 Драйвер двигуна

Драйвер двигунів необхідний у цій роботі, оскільки він дозволяє зручно керувати двигунами, дозволяючи живити двигуни від акумулятора, а керувати ними через ШІМ сигнал з мікроконтролера (що дозволяє регулювати їх швидкість). Також він дозволить двигунам обертатися в обидві сторони, що дозволить реалізувати переміщення машини назад, змінюючи направлення оберту двигуна, відповідального за задню вісь, та реалізувати повертання направо та наліво, змінюючи напрям оберту переднього двигуна.

У якості драйверу двигуна спочатку розглядався варіант плати L298, але був швидко відкинутий, оскільки він базується на звичайних біполярних транзисторах, що виділяють багато тепла, та не є енергоефективними. Отже, було вирішено обрати драйвер Mx1508 (рисунок 2.2), який ідеально підходить для використання в моделях радіокерованих машинок, дронів, іграшок, та інших моделей з рухомими деталями, які живляться від батарейок або акумуляторів. Напруга живлення драйверу від 2 В до 10 В, драйвер може

керувати двома двигунами постійного струму або 4 – провідними двофазними кроковими двигунами, а також дозволяє регулювати швидкість обертання та змінювати напрямок обертання (реверс). Mx1508 може забезпечити постійний струм 1.5A та піковий струм до 2.5A, а також має блок теплового захисту з автоматичним відновленням [5].

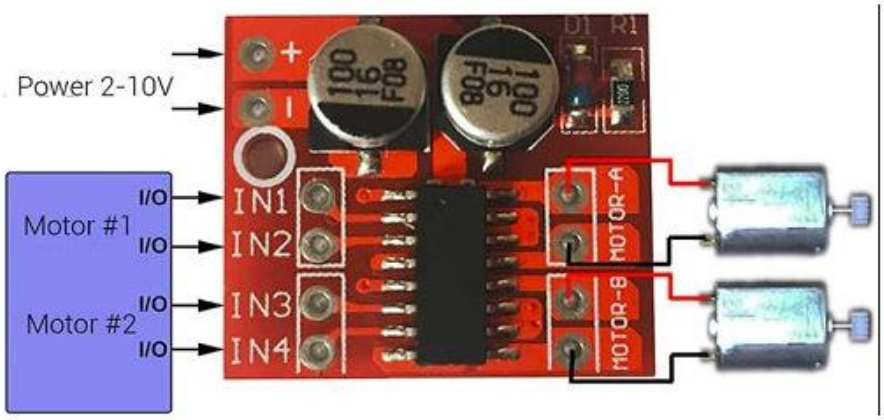

Рисунок 2.2 – Драйвер Mx1508 та приклад його підключення до двох двигунів.

2.2.3 Система живлення – акумулятор та модулі для конвертування напруги.

У якості головного елемента живлення використовується акумулятор типу 18650, що є одним з найрозповсюдженіших типів акумуляторів, які широко використовуються у різноманітних портативних електронних пристроях, таких як ноутбуки (у складі батарейних блоків), павербанки, потужні ліхтарі, деякі електроінструменти, та навіть у батарейних модулях деяких електромобілів та систем накопичення енергії. Ці акумулятори високо цінуються за їх високу питому енергетичну щільність, тобто здатність накопичувати значний заряд відносно свого розміру та ваги. Вони мають номінальну напругу, що становить 3.7 В, та пропонують широкий діапазон місткостей, який, залежно від хімічного складу та виробника, може варіюватися від приблизно 1500 мАг до 3500 мАг і навіть вище. Для моделі

було обрано акумулятор Samsung ICR18650 – 24E, місткість якого складає 2400mAh [6].

Модуль для конвертації напруг необхідний у даній роботі, тому що напруга акумуляторної батареї варіюється в діапазоні від 3.2 В до 4.2 В при повному заряді, чого недостатньо для роботи мікроконтролера, бо його робоча напруга – 5 В. Отже, було вирішено використовувати модуль HTM2356, який призначається для створення саморобних павербанків, оскільки він може конвертувати напругу з акумулятора у стабільні 5 В, а також виконує функції захисту від перевантаження (рисунок 2.3).

Рисунок 2.3 – Модуль для перетворення напруги з акумулятора у 5 В

Також потрібно розв’язати задачу заряджання акумулятора. Для цього потрібен окремий модуль, який буде підтримувати сучасний роз’єм для заряджання, функції захисту від перезаряду та перерозряду, перевантаження по струму та короткого замикання, а також бути компактним та дешевим. Отже, було обрано модуль TP4056 HW – 373 (рисунок 2.4), що призначається для заряджання одного акумулятора, має сучасний та зручний роз’єм USB Type – C, що дозволить заряджати автомобіль за допомогою стандартного адаптера для смартфона, має вхідну напругу 5 В та вихідну (для заряджання акумулятора) 4.2 В, та максимальний струм заряду 1 А. Також він має у своєму складі мікросхему DW01 та декілька MOSFET – транзисторів для реалізацій функцій захисту акумулятора та світлодіоди для перевірки стану зарядки (наприклад, червоний – заряджається, синій – заряд завершено) [7].

Рисунок 2.4 – Модуль TP4056 HW – 373

2.2.4 Дисплей

Було вирішено додати дисплей, який буде відображати корисну інформацію про поточний стан моделі. Головні критерії вибору – розмір дисплея, та чи підтримує дисплей кольори. Отже, спочатку розглядався варіант з одноколірним дисплеєм на 0.96 дюйма, але було вирішено, що він занадто маленький, щоб бачити його здалека на моделі, отже було вирішено використати кольоровий дисплей на 1.69 дюйми.

Цей дисплей має роздільну здатність 240 на 280 пікселів, може відображати до 262000 кольорів та є гарним варіантом для розроблюваної моделі, оскільки дозволяє відображати інформацію різними кольорами, а розмір самого дисплею дозволяє відображати велику кількість інформації та бачити його зміст здалеку. Для підключення дисплей використовує інтерфейс SPI, фактичний послідовний синхронний повнодуплексний стандарт передачі даних, який був розроблений компанією Motorola у 1980 – х роках для забезпечення простого сполучення мікроконтролерів та периферійних пристроїв, таких як цей дисплей [8]. Оскільки обраний мікроконтролер також підтримує цей інтерфейс, дисплей та мікроконтролер є сумісними. Зображення дисплею можна побачити на рисунку 2.5.

Рисунок 2.5 – Багатокольоровий IPS дисплей на 1.69 дюйми

2.2.5 Схема апаратної частини

Щоб забезпечити сумісну роботу усіх компонентів розроблюваної моделі, була розроблена схема, що представлена на рисунку 2.6. На ній зображені всі апаратні компоненти та спосіб їх взаємодії. Більш детальний опис буде розміщено у наступному розділі.

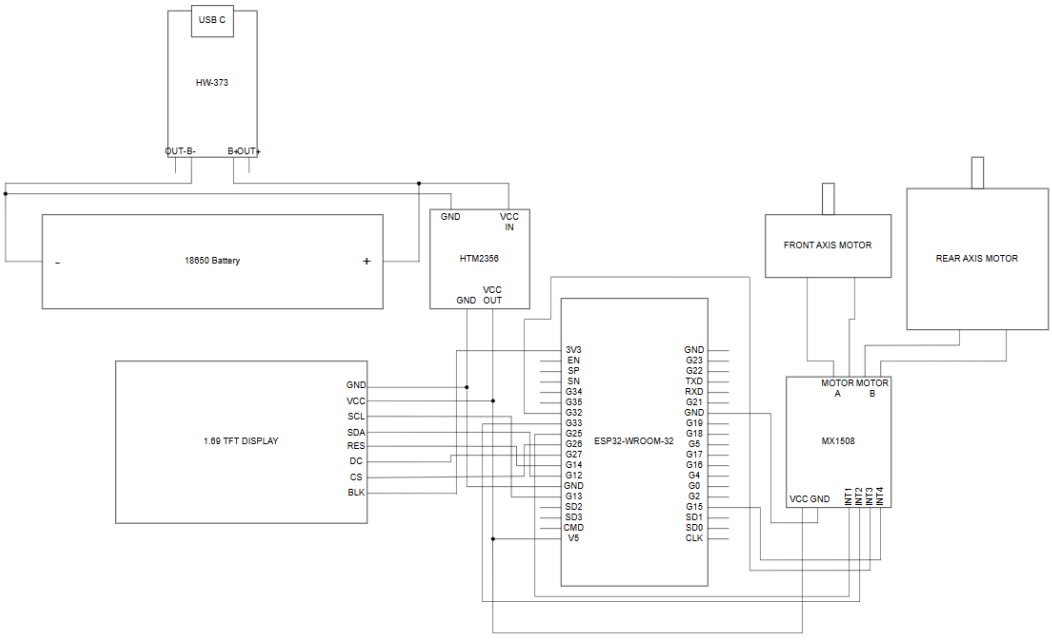

Рисунок 2.6 – Схема апаратної частини

2.3 Розробка програмної частини

Створення програмної частини було поділено на два великих блоки, які будуть розроблюватися паралельно у різних середовищах розробки та на різних мовах програмування. Такий поділ зумовлено тим, що потрібно розробити дві окремі програми, одна з яких буде відповідати за керування ззовні з мобільного пристрою, а інша буде приймати сигнали та оброблювати їх належним чином на мікроконтролері.

2.3.1 Створення прошивки для мікроконтролера

Перший блок – створення прошивки для мікроконтролера. Вона повинна виконувати такі функції:

- виконувати та підтримувати з'єднання з додатком на мобільному пристрої;
- коректно оброблювати вхідні сигнали;
- виконувати керування моторами шляхом надсилання потрібних сигналів на драйвер моторів;
- виконувати вивід на дисплей;
- виводити інформацію у Serial для подальшого відлагодження та полегшення розробки;

Для розробки прошивки для мікроконтролера було обрано середовище Arduino IDE, бо ESP32 є ардуіно – сумісним. Це середовище дуже просте в освоєнні, його інтерфейс зрозумілий, а структура коду зі стандартними для ардуіно функціями setup() і loop() допомагає швидко розпочати роботу. Також є величезна кількість готових бібліотек для ESP32 що дозволяє легко підключати та використовувати різні датчики та модулі, вбудовані функції Wi-Fi та Bluetooth та з легкістю керувати двигунами без необхідності писати весь код з нуля [9]. Інтерфейс Arduino IDE можна побачити на рисунку 2.7.

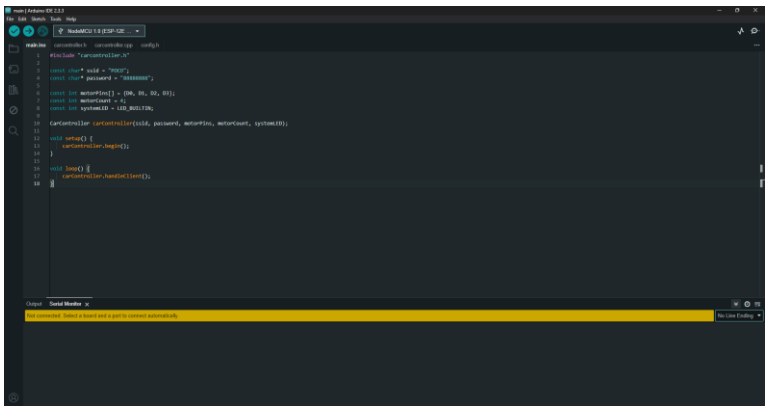

Рисунок 2.7 – Інтерфейс Arduino IDE

2.3.2 Розробка мобільного застосунку

Другий блок – розробка мобільного застосунку. Цей застосунок має виконувати функції:

- отримання необхідних для функціонування застосунку дозволів у операційній системі;
- підключення до мікроконтролера;
- здійснювати надсилання сигналів;
- мати зрозумілий для користувача інтерфейс;

Для розробки мобільного застосунку було обрано операційну систему Android, для якої буде розроблений мобільний додаток. Обране середовище розробки – Android Studio, тому що це офіційне інтегроване середовище розробки від Google, яке надає повний набір інструментів, спеціально адаптованих для створення застосунків саме під цю операційну систему, забезпечує найкращу інтеграцію з Android SDK, підтримує рекомендовані мови програмування (Kotlin та Java), має потужний редактор коду, інструменти для налагодження, емулятор, дизайнер інтерфейсу, та оптимізоване для ефективної розробки та тестування Android – додатків, включаючи доступ до інструментів аналізу продуктивності, що є досить важливим для створення стабільного та швидкодіючого застосунку для керування моделлю [10]. Інтерфейс Android Studio зображено на рисунку 2.8.

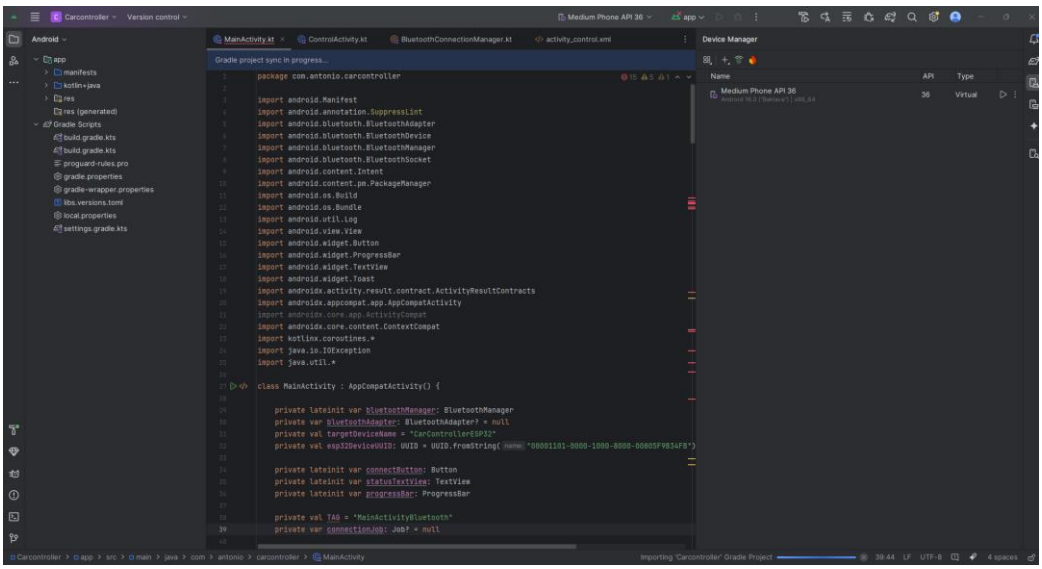

Рисунок 2.8 – Інтерфейс Android Studio

В якості мови програмування було вирішено обрати Kotlin, оскільки це офіційна, рекомендована Google мова програмування для розробки під Android, яка є більш лаконічною та виразною порівняно з Java, що дозволяє писати менше коду для досягнення того самого результату, має вбудовану безпеку від NullPointerException (проблеми з нульовими посиланнями), роблячи додатки більш стабільними та надійними, пропонує сучасні конструкції мови та функції (як – от корутини для асинхронного програмування, розширення, data classes) і повністю сумісна з Java, що дозволяє використовувати всі наявні бібліотеки та інтегрувати її у вже присутні Java – проєкти, роблячи процес розробки швидшим.

3 РЕАЛІЗАЦІЯ ПРОЕКТУ ТА ТЕСТУВАННЯ МОДЕЛІ

3.1 Реалізація та складання корпусу автомобіля

Розробка корпусу моделі, у який надалі будуть встановлені усі інші апаратні компоненти є однією з трьох частин цієї роботи. На цьому етапі необхідно визначити оптимальне просторове розташування всіх конструктивних елементів та як вони будуть взаємодіяти. Ключовим завданням корпусу є забезпечення інтеграції електронних компонентів з фізичними складовими, а саме з двома основними колісними осями: задньою та передньою. Задня вісь відповідає за реалізацію руху автомобіля, тоді як передня вісь призначена для здійснення маневрів шляхом зміни напрямку руху. Проєктування зазначених осей є важливим, оскільки задня вісь повинна генерувати достатнє зусилля для переміщення моделі з усіма її компонентами, а передня вісь – забезпечувати керованість через зміну кута повороту коліс.

3.1.1 Реалізація задньої віхи

Задня вісь у даній роботі відповідає за рухи вперед та назад, а також повинна мати місце та спосіб для закріплення дисплея. Отже, задля реалізації цієї віхи, потрібно продумати наступні моменти:

- механізм конвертування високої кутової швидкості вала двигуна у відповідне збільшення крутного моменту;
- спосіб закріплення дисплея у зоні задньої віхи;
- реалізація з’єднання задньої вісі з іншою частиною корпусу;

Задля конвертації високої кутової швидкості вала двигуна у відповідне збільшення крутного моменту було вирішено реалізувати редуктор. Редуктор – це механічний пристрій, який призначений для зміни кутової швидкості та

крутного моменту обертового руху. Основна функція редуктора полягає у зменшенні кутової швидкості (кількості обертів за одиницю часу) від ведучого вала (наприклад, від двигуна) та одночасному збільшенні крутного моменту на веденому валу (який приводить у рух механізм). Може бути реалізований за допомогою зубчатої (зокрема черв'ячної), гідравлічної, або ремінної передачі [11]. Редуктор у реалізованій моделі має у своєму складі як зубчату, так і ремінну передачу, тобто є подвійним, та складається з валу двигуна, проміжного валу та валу, на якому закріплені колеса. Перший ступінь редуктора реалізований за допомогою клиноремінної передачі, що складається з ведучого шківа, встановленого на валу електродвигуна, та веденого шківа, закріпленого на проміжному валу. Другий ступінь редуктора являє собою циліндричну зубчасту передачу прямозубого типу. Вона складається з ведучої шестерні з 12 зубцями, розташованої на проміжному валу, та веденої шестірні з 20 зубцями, жорстко з'єднаної з віссю коліс. Цей тип передачі обраний для забезпечення надійного зачеплення та передачі значного крутного моменту на привідні колеса. Така комбінована конструкція редуктора дозволяє ефективно поєднати переваги обох типів передач: ремінна передача на вході забезпечує поглинання частини вібрацій від двигуна та простоту конструкції, тоді як зубчаста передача на виході гарантує точне передавальне відношення та здатність передавати більший крутний момент на колеса. Зовнішній вигляд реалізованого редуктора можна побачити на рисунках 3.1 та 3.2.

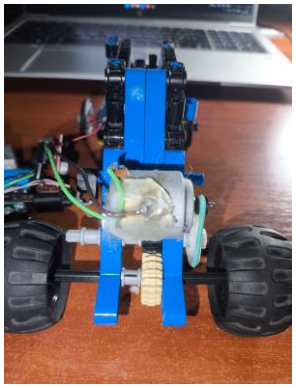

Рисунок 3.1 – Реалізація редуктора, вид сзаду

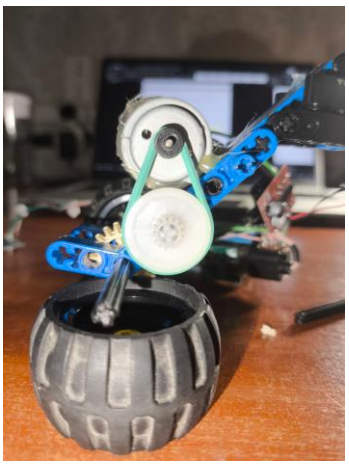

Рисунок 3.2 – Реалізація редуктора, вид збоку

Щодо головного параметру редуктора – передатного числа, його у даному випадку можна розрахувати за наступною формулою:

$$U = \left(\frac{D_{вед}}{D_{вдч}} \right) * \left(\frac{N_{вед}}{N_{вдч}} \right), \quad (3.1)$$

де $D_{вед}$ – діаметр веденого шківу, мм;

$D_{вдч}$ – діаметр ведучого шківу, мм;

$N_{вед}$ – кількість зубців на веденій шестерні, шт;

$N_{вдч}$ – кількість зубців на ведучої шестерні, шт;

Підставивши значення у формулу, отримуємо значення передатного числа: 20/6 * 20/12 = 5.56, що значить, що для одного повного оберту вала з колесами (веденого вала другого ступеня), вал двигуна (ведучий вал першого ступеня) повинен зробити приблизно 5.56 обертів. Також це свідчить про те, що крутний момент на валу з колесами теоретично (без урахування втрат на тертя в редукторі) буде приблизно у 5.56 разів більшим, ніж крутний момент, що розвивається безпосередньо на валу двигуна. Таке збільшення крутного моменту є необхідним для забезпечення здатності моделі автомобіля долати сили опору руху та ефективно переміщуватися.

Задля сумісності деталей конструктора та апаратного обладнання було

вирішено до кожного апаратного компонента прикріпити деталь конструктора за допомогою термоклею. Цей підхід дозволить більш гнучко модернізувати конструкцію у майбутньому, оскільки концептуально кожен елемент буде мати “інтерфейс” для з'єднання з конструктором. Отже, до дисплея були прикріплені дві деталі, що дозволило закріпити його ззаду на конструкції ведучої віхи (рисунок 3.3).

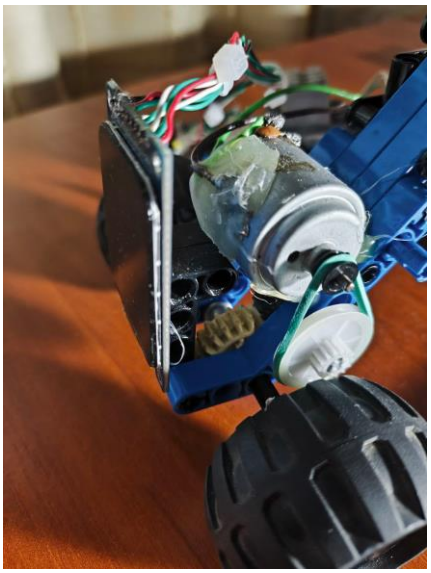

Рисунок 3.3 – Закріплення дисплея на ведучій вісі

Щодо реалізації з’єднання задньої віхи з іншими частинами корпусу – це питання буде розглянуто після розглядання реалізації передньої, поворотної віхи.

3.1.2 Реалізація передньої віхи

Передня вісь у даній моделі відповідає за повороти направо та наліво. Задля реалізації цієї віхи, потрібно продумати наступні моменти:

- реалізація механізму повороту коліс;
- механізм повертання коліс у початковий стан;
- реалізація з’єднання передньої вісі з іншою частиною корпусу;

При створенні механізму повертання коліс головною ідеєю було взяти

за основу механізм рейкового повертання у реальних моделях авто та змінити його під потреби розроблюваної моделі (рисунок 3.4).

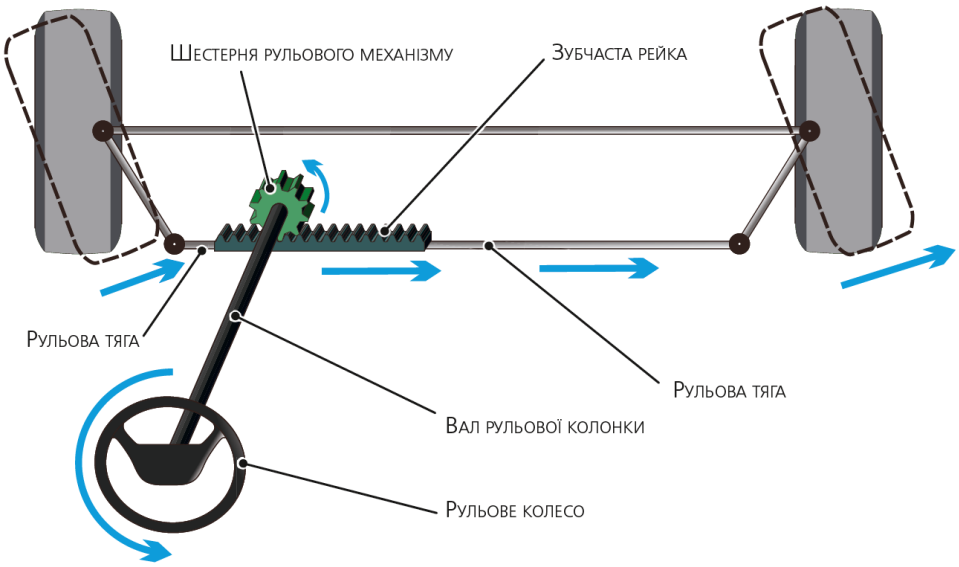

Рисунок 3.4 – Механізм рейкового повертання

Отже, зміни у розроблюваній моделі полягають в тому, щоб замінити рульове колесо мотором, а замість рейки використовувати шестерню (тобто зубчату передачу). Запланована реалізація повертального механізму схематично зображена на рисунках 3.5 та 3.6.

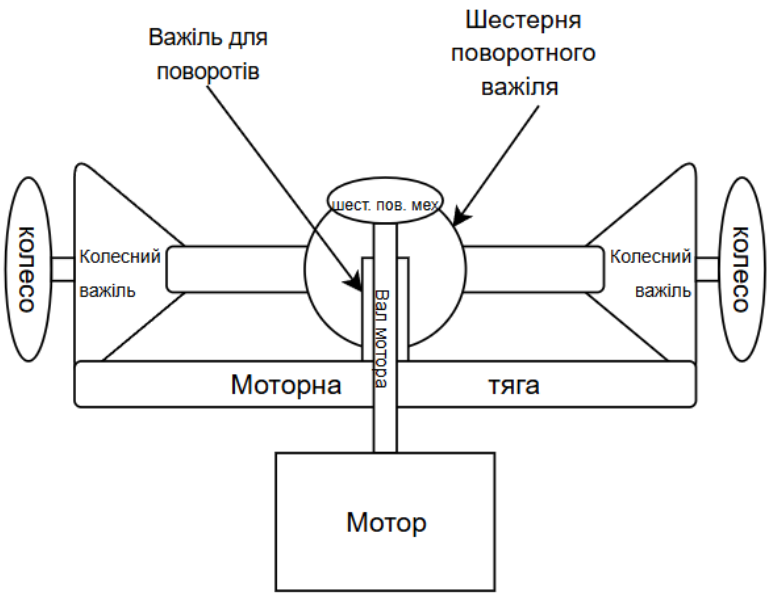

Рисунок 3.5 – Схематичне зображення розроблюваного механізму (статичне)

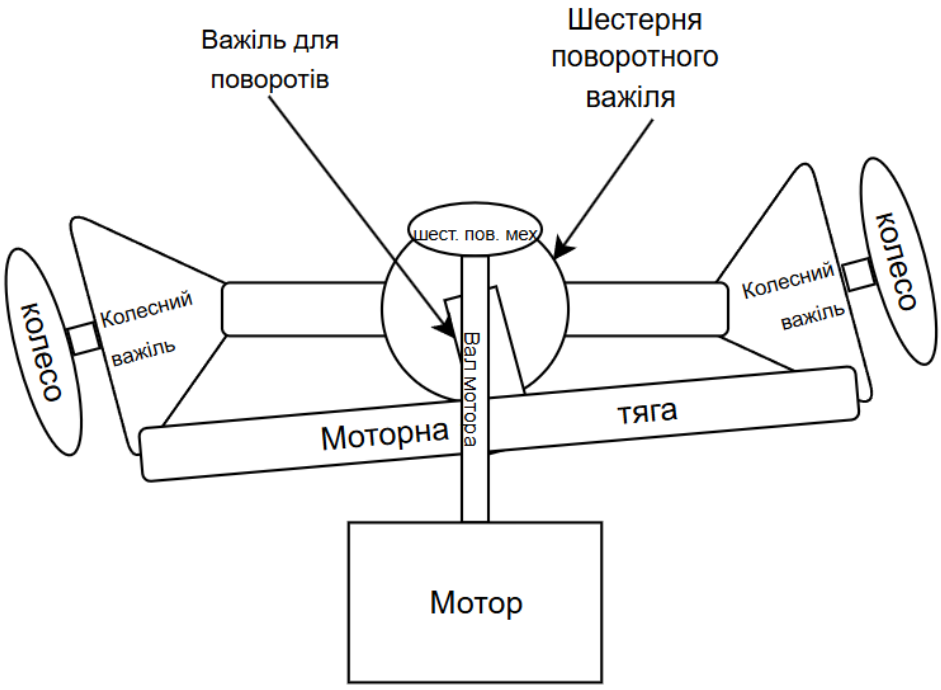

Рисунок 3.6 – Схематичне зображення розроблюваного механізму (поворот)

Механізм роботи даного поворотного механізму полягає в тому, що оберти від мотора передаються на шестерню поворотного важеля через шестерню поворотного механізму. Також, оскільки шестерня поворотного механізму є меншою за шестерню поворотного важеля, тут, як й при реалізації ведучої вісі, є редуктор, який дозволяє зменшувати кутову швидкість від ведучого вала та пропорціонально збільшувати крутний момент на веденому валу, але тут він простіший та складається лише з однієї пари шестерень. Шестерня поворотного механізму, своєю чергою, повертає важіль для поворотів, який зміщує рейку моторної тяги. Ця рейка зміщує обидва колісних важеля, що змушує колеса змінювати свій кут повороту. Саме цей механізм дозволяє моделі реалізовувати повертання направо та наліво під час руху.

Наступний крок при розробці поворотного механізму випливає зі специфіки цього механізму, а саме повертання коліс у нейтральне положення, оскільки без реалізації цієї функції модель буде їхати у сторону

останнього повороту, навіть якщо кнопку для поворотів відпустити. Для реалізації цієї функції попередню схему було доповнено резинкою для стабілізації. Вона прикріплюється до нерухомої частини поворотного механізму за допомогою додаткових деталей у центрі, а також до самих колісних важелів. Доповнена схема зображена на рисунку 3.7.

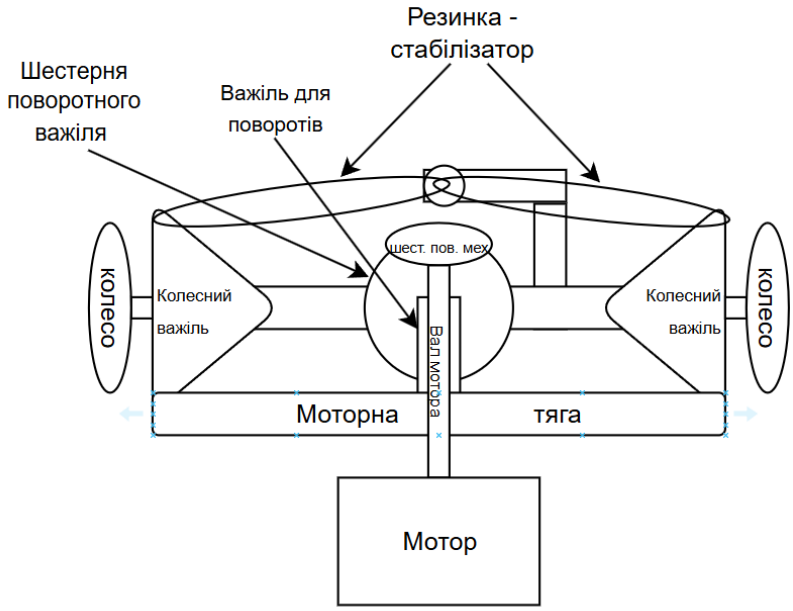

Рисунок 3.7 – Схема, доповнена системою стабілізації

Отже, фінальний результат реалізації передньої поворотної віхи можна побачити на рисунку 3.8.

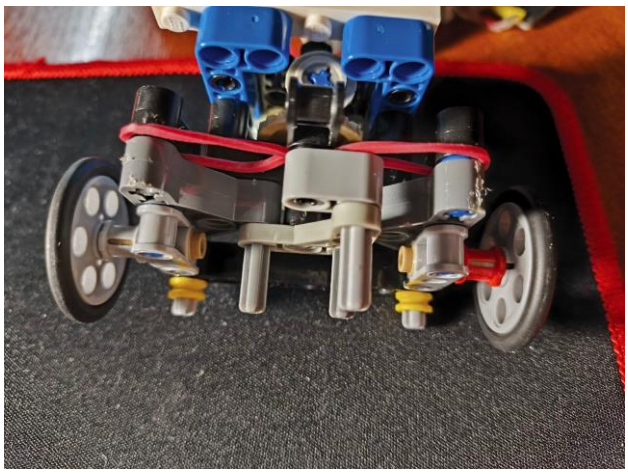

Рисунок 3.8 – Реалізація поворотної віхи

3.1.3 Фінальний результат реалізації корпусу

Головні задачі корпусу – це забезпечувати можливість моделі їздити вперед та назад та виконувати повороти, а також забезпечувати розміщення усіх апаратних компонентів. Оскільки задня та передня вісі вже реалізовано, залишається їх з’єднати та забезпечити місце для апаратних компонентів, найбільшими з яких є плата з мікроконтролером, акумулятор та дисплей. Дисплей вже розміщено у задній частині моделі, тому залишаються плата та акумулятор. Було вирішено поєднати задню та передню вісі за допомогою додаткових деталей, саме на яких й будуть розміщуватися ці апаратні компоненти, а також драйвер для моторів. Оскільки апаратна частина та корпус розроблювалися одночасно (з ціллю підігнати всі деталі), готовий варіант зібраної моделі буде зображено у наступному підрозділі.

3.2 Реалізація апаратної частини

Задля реалізації апаратної частини, була реалізована схема, що зображена на рисунку 3.9.

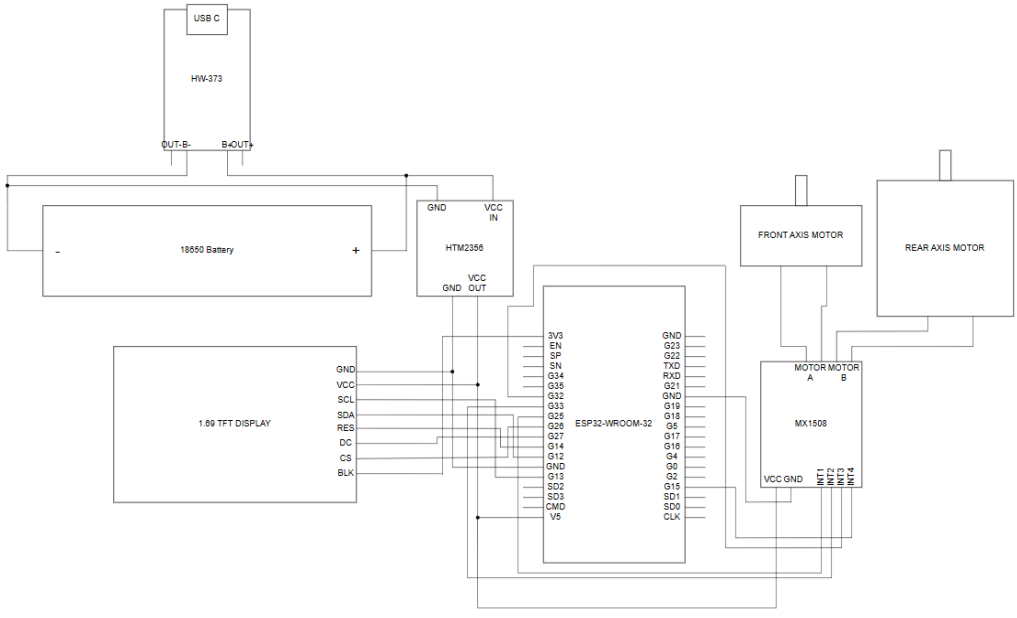

Рисунок 3.9 – Схема апаратної частини

У цій схемі головним центральним елементом, що координує роботу всіх систем моделі, є мікроконтролер, та саме на ньому була сконцентрована увага при проєктуванні апаратної частини та згодом при збірці. Плата HW-373 виконує функцію заряджання акумулятора. Отже, підключена до нього. Плата HTM2356 конвертує напругу з акумулятора (яка залежно від рівня заряду може варіюватися приблизно від 3.2 В до 4.2 В, із номінальним значенням 3.7 В) у стабільні 5 В, які потрібні для живлення мікроконтролера, дисплею та драйверу двигунів. Мікроконтролер, задля керування дисплеєм підключений до нього за допомогою пінів 3V3 задля керування підсвіткою дисплея, до VCC (напруги 5 В з виходу HTM2356) та до GND (земля) задля забезпечення живлення вбудованого у дисплей чіпа ST7789, який виконує роль графічного контролера та моста між екраном і мікроконтролером. Для реалізації передачі даних на дисплей використовуються піни мікроконтролера G13, G12, G14, G27, G26, які програмно емулюють шину SPI (апаратний SPI також доступний у мікроконтролері, але задля зручності підключення було вирішено використовувати програмну версію), що підключена до пінів SCL (синхросигнал), SDA (дані), RES (апаратне скидання дисплея), DC (вибір передавання даних або команд), CS (для вибору дисплея, якщо їх кілька) дисплею відповідно. Мікроконтролер надсилає сигнали драйверу двигунів за допомогою пінів G25 та G33 для контролювання передньої вісі коліс та G32 та G15 для контролювання задньої віхи. Задля керування моторами на драйвері є спеціальні піни для підключення MOTOR-A, до яких був підключений мотор передньої віхи та MOTOR-B, до яких був підключений мотор задньої віхи. Живлення драйвера двигуна здійснюється через плату HTM2356.

Отже, поєднавши корпус, реалізований у попередньому підрозділі та апаратну частину, було отримано результат, що зображено на рисунку 3.10. На цьому рисунку можна побачити, як були розташовані акумулятор, мікроконтролер та драйвер двигуна відносно корпуса моделі та як було реалізовано кабель менеджмент.

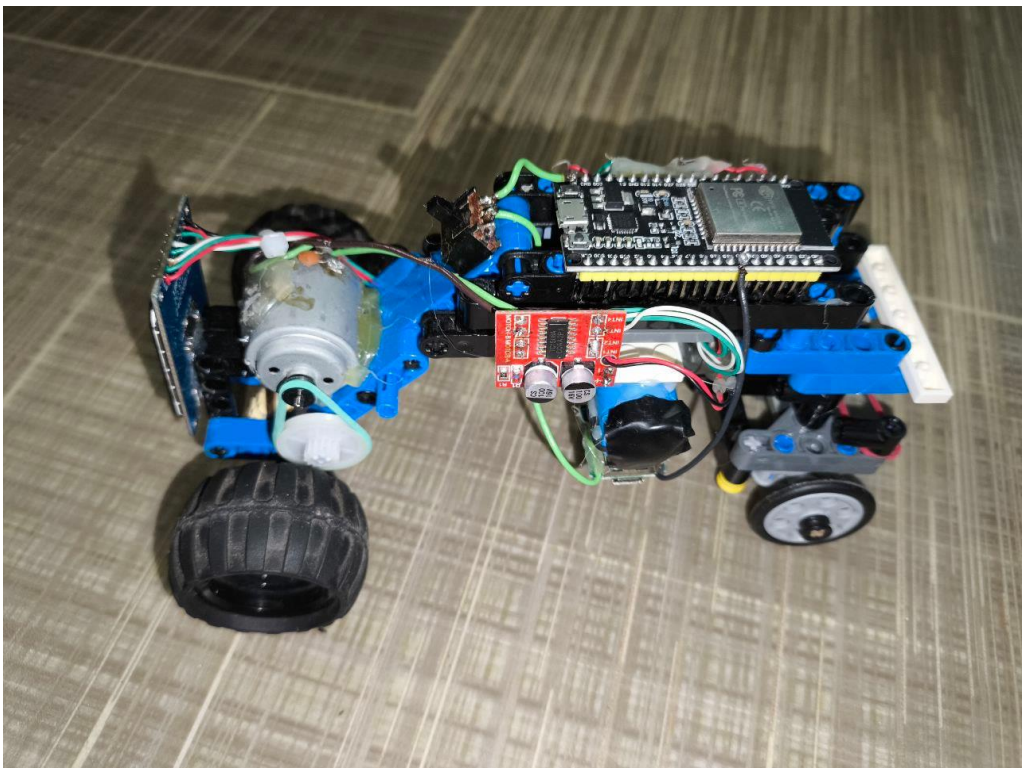

Рисунок 3.10 – Фінальна реалізація корпусу та апаратної частини, вид збоку

3.3 Реалізація програмної частини

Реалізація програмної частини поділяється на дві основних частини, які розроблюються у різних середовищах розробки та різними мовами програмування. Прошивка для мікроконтролера була написана мовою C++ у середовищі Arduino IDE. Мобільний додаток було написано мовою програмування Kotlin у середовищі Android Studio.

3.3.1 Реалізація прошивки для мікроконтролера

Задля зручності розробки увесь код було вирішено поділити на 5 файлів. Ці файли – BluetoothHandler.cpp та BluetoothHandler.h задля реалізації усіх функцій, що пов’язані з Bluetooth зв’язком, DisplayHandler.cpp та DisplayHandler.h задля реалізації роботи з дисплеєм, та головний файл ESP32Server.ino, у якому реалізовано основну логіку роботи мікроконтролера, в тому числі й виконання функцій з вищезгаданих файлів.

Розділення коду C++ на заголовкові файли (.h) та файли реалізації (.cpp) є важливою практикою для модульності, читабельності та подальшого супроводу проєктів. Заголовкові файли визначають публічний інтерфейс компонента, приховуючи деталі реалізації у файлах .cpp. Це сприяє повторному використанню коду, зменшує залежності та оптимізує компіляцію, оскільки зміни в одному модулі не вимагають повної перекомпіляції всього проєкту.

Файл BluetoothHandler.h, наведений у лістингу 3.1, є заголовковим файлом для файлу BluetoothHandler.cpp. Він містить у собі #define BLUETOOTH_HANDLER_H (у зв'язці з #ifndef BLUETOOTH_HANDLER_H та #endif) задля запобігання проблемам множинного включення цього заголовкового файлу, include <Arduino.h> для включення типу даних String у роботу, callback – функцію для створення типу даних, який є покажчиком на функцію, що дозволяє реалізувати механізм зворотних викликів (callbacks) для обробки подій Bluetooth (наприклад, підключення чи відключення), викликаючи функції, визначені користувачем. та які оголошують інтерфейс модуля обробки Bluetooth, визначаючи, які функції він надає, які параметри приймає та що повертає, і які надалі будуть реалізовані (матимуть свій код) у файлі BluetoothHandler.cpp.

Лістинг 3.1 – Вміст файлу BluetoothHandler.h

```
#ifndef BLUETOOTH_HANDLER_H
#define BLUETOOTH_HANDLER_H
#include <Arduino.h>
typedef void (*BluetoothEventCallback)();
void setupBluetooth(const String& deviceName);
bool loopBluetooth(String &receivedCommand);
void registerBluetoothCallbacks(BluetoothEventCallback
onConnectCallback, BluetoothEventCallback onDisconnectCallback);
#endif
```

Файл BluetoothHandler.cpp, наведений у Додатку Б, лістингу Б.1.2, є файлом реалізації для заголовкового файлу BluetoothHandler.h. Він містить у собі включення BluetoothHandler.h для узгодження інтерфейсу та реалізації,

включення BluetoothSerial.h для надання функціоналу Bluetooth Serial Port Profile, створення глобального об'єкта BluetoothSerial SerialBT для керування Bluetooth-з'єднанням, оголошення статичних покажчиків на callback-функції connectCallback та disconnectCallback для зберігання адрес користувацьких функцій обробки подій. Також він містить внутрішню callback-функцію btInternalCallback, яка реєструється в бібліотеці BluetoothSerial для обробки системних подій Bluetooth та виклику користувацьких callback-функцій, функцію ініціалізації Bluetooth setupBluetooth для налаштування Bluetooth-сервера та реєстрації btInternalCallback, функцію реєстрації користувацьких callback-функцій registerBluetoothCallbacks для передачі покажчиків на функції обробки підключення та відключення з основної програми, та функцію обробки даних Bluetooth в циклі loopBluetooth, яка зчитує вхідні дані, накопичує їх до символу нового рядка, формує повну команду та сигналізує про її успішне отримання.

Файл DisplayHandler.h, наведений у лістингу 3.2, є заголовковим файлом для файлу DisplayHandler.cpp. Він містить у собі #define DISPLAY_HANDLER_H (у зв'язці з #ifndef DISPLAY_HANDLER_H та #endif) задля запобігання проблемам множинного включення, #include <Arduino.h> для включення базових функцій та типів платформи Arduino. Далі йдуть директиви #define для визначення констант, що задають піни підключення дисплея, його розміри, та кольори елементів інтерфейсу, інтервал мерехтіння та константи для дій малювання стрілок. Також у файлі оголошені зовнішні змінні (extern volatile) для обміну даними про поточний напрямок, необхідність оновлення дисплея, стан підключення клієнта та необхідність оновлення статусу з'єднання між різними файлами роботи. Також файл містить прототипи функцій setupDisplay для ініціалізації дисплея, startDisplayTask для запуску задачі оновлення дисплея та requestDisplayUpdate для запиту на оновлення відображення з вказанням напрямку у якості параметру функції, реалізація яких буде знаходитися у файлі DisplayHandler.cpp.

Лістинг 3.2 – Вміст файлу DisplayHandler.h

```
#ifndef DISPLAY_HANDLER_H
#define DISPLAY_HANDLER_H
#include <Arduino.h>
// Display pin definitions
#define TFT_SCLK 13
#define TFT_MOSI 12
#define TFT_RST  14
#define TFT_DC   27
#define TFT_CS   26
#define TFT_WIDTH  240
#define TFT_HEIGHT 280
// Display params
#define ARROW_COLOR ST77XX_BLUE
#define BACKGROUND_COLOR ST77XX_BLACK
#define BLINK_INTERVAL_MS 500
// Constants for arrow drawing actions
#define ACTION_DRAW 1
#define ACTION_ERASE 0
// Extern vars to share with other files
extern volatile char currentDirectionToDisplay;
extern volatile bool displayNeedsUpdate;
extern volatile bool isClientConnected;
extern volatile bool connectionStatusNeedsUpdate;
// Function Prototypes
void setupDisplay();
void startDisplayTask();
void requestDisplayUpdate(char direction);
#endif
```

Зміст файлу DisplayHandler.cpp можна побачити у Додатку Б, лістинг Б.1.4. Він є файлом реалізації для заголовкового файлу DisplayHandler.h, та містить функції для реалізації роботи дисплея. Він містить у собі включення бібліотек Adafruit_GFX.h, Adafruit_ST7789.h та SPI.h для роботи з графікою, дисплеєм ST7789 та SPI-інтерфейсом відповідно, а також заголовків FreeRTOS (freertos/FreeRTOS.h, freertos/task.h) для багатозадачності та driver/adc.h для роботи з АЦП.

Функція setupDisplay() ініціалізує дисплей, встановлює його розміри, режим SPI та заповнює екран фоновим кольором. Функція startDisplayTask() створює та запускає завдання FreeRTOS displayTaskCode, закріплюючи його за ядром 1, для асинхронного оновлення дисплея. Функції drawOrEraseArrow(), drawOrEraseArrowHead(), та

updateConnectionStatusDisplay() працюють схожим чином – drawOrEraseArrow() та drawOrEraseArrowHead() мають два вхідних параметри – напрям руху та дія, updateConnectionStatusDisplay() має лише параметр дії. Отже, всі три функції спочатку встановлюють потрібне їм обертання дисплея та колір, яким потрібно провести вивід на дисплей – або основний колір для відображення, або чорний колір для стирання. Потім функції визначають та підраховують параметри того, що потрібно відобразити – позицію та розмір для стрілки або ж текст та розмір для його відображення.

Функція displayTaskCode(), яка є найголовнішою функцією для роботи дисплея, реалізована для виконання в асинхронному режимі в рамках окремого завдання операційної системи реального часу FreeRTOS. Її робота організована у вигляді нескінченного циклу, що забезпечує безперервний моніторинг та реагування на зміни стану системи. На початку своєї роботи завдання ініціалізує внутрішні статичні змінні, що зберігають попередній стан. Також на старті завдання відбувається первинне відображення статусу Bluetooth-з'єднання, виводячи напис "READY TO CONNECT", якщо клієнт не підключений, або очищуючи цей напис, якщо підключення вже активне. У кожній ітерації основного циклу функція зчитує поточні значення глобальних змінних, та на основі цих даних була реалізована логіка керування станами. Спочатку, перевіряється необхідність оновлення текстового індикатора статусу з'єднання: якщо стан підключення змінився порівняно з попередньою ітерацією, або надійшов явний запит на оновлення, функція відповідно виводить або стирає напис "READY TO CONNECT". При цьому, якщо клієнт відключається, будь-яка раніше намальована стрілка напрямку стирається з екрана. Потім, якщо клієнт підключений і надійшов запит на зміну напрямку руху (або напрямок змінився з моменту останнього малювання), функція displayTaskCode переходить до оновлення графічного індикатора – стрілки. Вона спочатку стирає попередню стрілку, а потім малює нову, що відповідає поточному напрямку, використовуючи для цього

drawOrEraseArrow(). Одночасно з перемальовуванням стрілки скидаються параметри її миготіння. Для активної стрілки напрямку реалізований механізм миготіння її наконечника. З використанням таймера, заснованого на функції millis(), та попередньо визначеного інтервалу BLINK_INTERVAL_MS, функція періодично інвертує стан видимості наконечника стрілки та викликає функцію drawOrEraseArrowHead для його оновлення. Це створює візуальний ефект, що привертає увагу до активного напрямку руху. Наприкінці кожної ітерації циклу завдання displayTaskCode викликає функцію vTaskDelay, передаючи їй невелику затримку (50 мілісекунд). Ця затримка є важливою для коректної роботи багатозадачної системи, бо вона дозволяє планувальнику FreeRTOS передати управління іншим завданням, запобігаючи монополізації процесорного часу одним завданням та можливим збоям системи через спрацювання сторожового таймера.

Файл ESP32Server.ino (Додаток Б, лістинг Б.1.5) є головним файлом у розроблюваній прошивці для мікроконтролера. Він містить у собі включення заголовкових файлів BluetoothHandler.h та DisplayHandler.h для доступу до функціоналу обробки Bluetooth-з'єднання та керування дисплеєм відповідно, визначення константи для імені Bluetooth-пристрою, макровизначення для пінів керування двома моторами, визначення рядкових констант команд для розпізнавання інструкцій, що надходять по Bluetooth, а також глобальні логічні змінні для відстеження поточного напрямку руху моторів. Також у файлі оголошені прототипи функцій, які були реалізовані та не є функціями setup() та loop(), типовими для Arduino проєктів.

Функція setup(), що виконується один раз при старті мікроконтролера, ініціалізує серійний порт для відладки, налаштовує піни моторів як виходи, викликає функцію stopAllMotors для встановлення початкового стану моторів, ініціалізує дисплей за допомогою setupDisplay(), налаштовує Bluetooth з визначеним ім'ям через setupBluetooth, реєструє callback-функції handleBluetoothConnect та handleBluetoothDisconnect для обробки подій

Bluetooth, та запускає асинхронне завдання для оновлення дисплея за допомогою startDisplayTask.

Функція loop(), що виконується у нескінченному циклі, постійно викликає loopBluetooth для перевірки та отримання команд по Bluetooth; у разі отримання команди вона обробляє її за допомогою серії умовних операторів if-else, встановлюючи відповідні стани на пінах моторів та оновлюючи глобальні змінні стану руху. Після обробки команди вона визначає пріоритетний напрямок для відображення на дисплеї (повороти мають вищий пріоритет над рухом вперед/назад) та викликає requestDisplayUpdate для оновлення графіки на дисплеї.

Функція stopAllMotors() встановлює низький рівень на всіх пінах керування моторами, скидає всі змінні стану руху та виводить повідомлення про зупинку.

Функції handleBluetoothConnect() та handleBluetoothDisconnect() викликаються відповідними подіями Bluetooth: перша встановлює глобальні прапорці isClientConnected та connectionStatusNeedsUpdate в true при підключенні, а друга встановлює isClientConnected в false, connectionStatusNeedsUpdate в true, викликає stopAllMotors для безпеки та requestDisplayUpdate для очищення стрілки напрямку на дисплеї при відключенні клієнта.

3.3.2 Реалізація мобільного застосунку

Розробка мобільних застосунків для повноцінних операційних систем, таких як Android, є більш специфічною за написання прошивки для мікроконтролера. Специфіка полягає у роботі з життєвим циклом компонентів, керуванні ресурсами, обробці дозволів користувача для доступу до апаратних функцій, адаптації інтерфейсу під різні розміри та орієнтації екранів, а також взаємодії з системою через Intents для запуску інших компонентів або застосунків.

У файлі AndroidManifest.xml (Лістинг 3.3) було визначено основні параметри застосунку, найважливішими з яких для задачі є запит дозволів на роботу з Bluetooth: BLUETOOTH та BLUETOOTH_ADMIN для старих версій Android (до API 30), а також BLUETOOTH_SCAN та BLUETOOTH_CONNECT для Android 12 (API 31) і вище, що дозволяє застосунку знаходити ESP32 та встановлювати з ним з'єднання. Також було оголошено дві основні активності: MainActivity як точку входу для ініціалізації Bluetooth-з'єднання та ControlActivity для безпосереднього керування автомобілем після успішного підключення. Для обох активностей було встановлено горизонтальну орієнтацію екрана, щоб забезпечити більш зручний інтерфейс керування. Крім того, в маніфесті вказано, що застосунок вимагає наявності апаратного Bluetooth модуля на пристрої.

Лістинг 3.3 – Файл AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.antonio.carcontroller">
    <uses-permission android:name="android.permission.BLUETOOTH"
android:maxSdkVersion="30" />
    <uses-permission
android:name="android.permission.BLUETOOTH_ADMIN"
android:maxSdkVersion="30" />
    <uses-permission
android:name="android.permission.BLUETOOTH_SCAN" />
    <uses-permission
android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-feature android:name="android.hardware.bluetooth"
android:required="true"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.CarController"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
```

```
            android:exported="true"
            android:screenOrientation="landscape">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ControlActivity"
            android:exported="false"
            android:screenOrientation="landscape"/>
    </application>
</manifest>
```

Файл activity_main.xml визначає візуальний інтерфейс для MainActivity, першого екрана застосунку. Його основне призначення – надати користувачеві можливість під'єднатися до моделі. У цьому макеті було розміщено ключові компоненти: кнопку buttonConnect з текстом "Connect to CarControllerESP32", яка запускає процес пошуку та з'єднання з Bluetooth-пристроєм, текстове поле textViewStatus для відображення поточного статусу процесу підключення та індикатор прогресу ProgressBar, який стає видимим під час активної спроби з'єднання. Елементи розташовані за допомогою ConstraintLayout таким чином, щоб кнопка була центральною, статус відображався під нею, а індикатор прогресу – над кнопкою, але видимий лише за потреби. Весь макет орієнтований на горизонтальне відображення екрана. Зовнішній вигляд цього екрану можна побачити на рисунку 3.11.

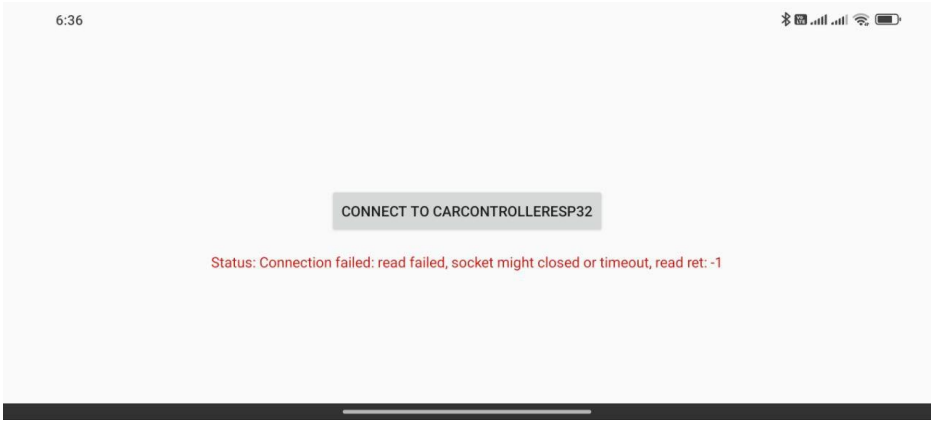

Рисунок 3.11 – Екран MainActivity для реалізації підключення

Файл activity_control.xml розроблено для визначення користувацького інтерфейсу ControlActivity, який активується після успішного Bluetooth-з'єднання з ESP32. Було розташовано чотири елементи типу ImageButton: buttonForward, buttonBackward, buttonLeft та buttonRight. Ці кнопки призначені для надсилання команд руху вперед, назад, вліво та вправо відповідно. Для візуального оформлення кнопок було використано векторні зображення стрілок (ic_arrow_up, ic_arrow_down, ic_arrow_left, ic_arrow_right) як джерело зображення (android:src) та спеціальний круглий фон (@drawable/button_background_circle), який також забезпечує візуальний відгук при натисканні. Кнопки руху вперед та назад розміщено у лівій частині екрана одна над одною, а кнопки повороту вліво та вправо – у правій частині, для зручного керування в горизонтальній орієнтації. У нижній частині екрана залишено текстове поле textViewControlStatus для відображення інформації про стан з'єднання або іншої службової інформації, хоча основний фокус цього екрана – на кнопках керування. Для позиціювання елементів також використано ConstraintLayout. Зовнішній вигляд цього екрану можна побачити на рисунку 3.12.

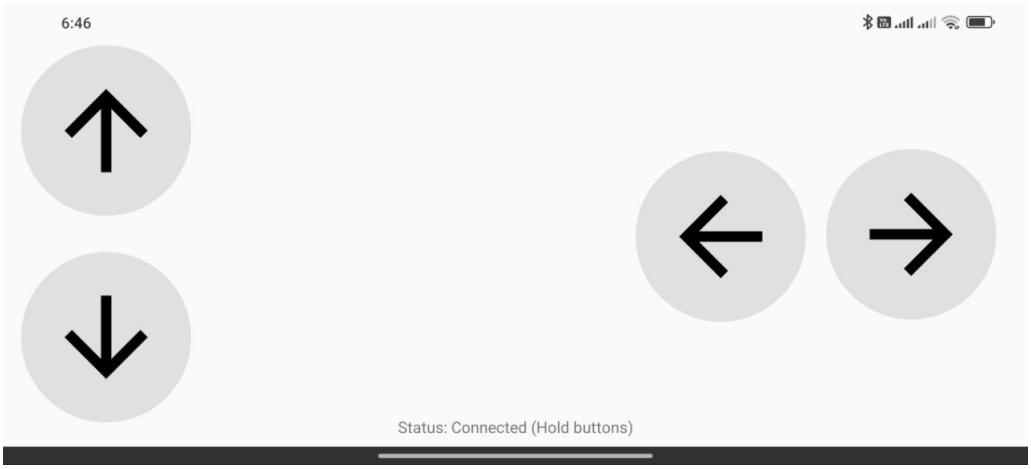

Рисунок 3.12 – Екран ControlActivity для реалізації керування

Файл BluetoothConnectionManager.kt (Додаток Б, лістинг Б.2.4) було створено як singleton object для централізованого керування станом Bluetooth-з'єднання та забезпечення доступу до потоку виводу (OutputStream)

з різних активностей (MainActivity та ControlActivity). Основна мета цього об'єкта – утримувати активний BluetoothSocket та пов'язаний з ним OutputStream після успішного встановлення з'єднання, дозволяючи ControlActivity надсилати команди на ESP32 без необхідності повторно встановлювати з'єднання або передавати складні об'єкти сокета між активностями. У ньому реалізовано функцію setSocket(), яка викликається з MainActivity при успішному з'єднанні для збереження сокета та отримання потоку виводу; функцію isConnected(), яка перевіряє, чи активне з'єднання та чи доступний потік виводу; функцію sendMessage(), яка приймає рядкову команду, додає до неї символ нового рядка як роздільник, перетворює в байтовий масив та надсилає через OutputStream на ESP32; та функцію closeConnection(), яка відповідає за коректне закриття потоку виводу та сокета, звільняючи ресурси Bluetooth.

Файл MainActivity.kt (Додаток Б, лістинг Б.2.5) реалізує логіку першого екрана застосунку, основною задачею якого є ініціалізація та встановлення Bluetooth-з'єднання з мікроконтролером ESP32. При запуску активності відбувається налаштування інтерфейсу користувача та отримання доступу до системних Bluetooth-сервісів. Ключова взаємодія починається з натискання кнопки "Connect", що запускає ланцюжок перевірок та дій: запит необхідних Bluetooth-дозволів, перевірка стану Bluetooth-адаптера (та запит на його увімкнення, якщо він вимкнений), і, нарешті, асинхронний пошук та підключення до спареного пристрою ESP32 з іменем "CarControllerESP32". У випадку успішного з'єднання сокет передається менеджеру з'єднань BluetoothConnectionManager, і відбувається автоматичний перехід до ControlActivity. Протягом усього процесу користувач інформується про поточний стан через текстове поле та індикатор прогресу.

Функція onCreate() – під час створення активності ініціалізуються елементи користувацького інтерфейсу (кнопка buttonConnect, текстове поле statusTextView, індикатор progressBar) та отримується доступ до BluetoothManager і BluetoothAdapter. Встановлюється обробник натискання

на кнопку buttonConnect, який ініціює процес перевірки дозволів та підключення. Перевіряється базова підтримка Bluetooth на пристрої.

Функція checkPermissionsAndConnect() – це приватна функція, що відповідає за перевірку наявності необхідних Bluetooth-дозволів. Вона визначає, які саме дозволи потрібні залежно від версії Android. Якщо дозволи відсутні, запускається системний діалог для їх отримання через requestMultiplePermissions.launch(). Якщо дозволи вже надано, викликається checkBluetoothStateAndConnect().

Функція checkBluetoothStateAndConnect() – після перевірки дозволів ця функція перевіряє, чи увімкнений Bluetooth-адаптер на пристрої. Якщо Bluetooth вимкнено, користувачеві пропонується його увімкнути через системний діалог, ініційований enableBluetoothLauncher.launch(). Якщо Bluetooth увімкнений, викликається startConnectionProcess().

Функція startConnectionProcess() виконує основну роботу з встановлення з'єднання і запускається в окремій корутині для уникнення блокування основного потоку. Вона отримує список спарених Bluetooth-пристроїв, шукає серед них цільовий пристрій "CarControllerESP32". Якщо пристрій знайдено, створюється RFCOMM BluetoothSocket з використанням стандартного SPP UUID і відбувається спроба підключення (socket.connect()). У разі успіху сокет зберігається в BluetoothConnectionManager, і здійснюється перехід до ControlActivity. Функція також обробляє можливі винятки під час спроби підключення, інформуючи користувача та оновлюючи інтерфейс.

Допоміжна приватна функція updateStatus() для оновлення текстового поля статусу (statusTextView) та видимості індикатора прогресу (progressBar) на головному потоці (runOnUiThread). Вона також керує активністю кнопки підключення залежно від поточного стану процесу.

Файл ControlActivity.kt (Додаток Б, лістинг Б.2.6) є кодом для другого екрана застосунку, який стає активним після успішного встановлення Bluetooth-з'єднання з ESP32. Його основне завдання – надати користувачеві інтерфейс для безпосереднього керування рухом автомобіля.

Функція onCreate() викликається при створенні активності. Тут відбувається зв'язування змінних класу з елементами інтерфейсу, визначеними в activity_control.xml. Перевіряється стан з'єднання через BluetoothConnectionManager; якщо воно неактивне, активність негайно закривається (finish()) для запобігання некоректної роботи. Якщо з'єднання активне, для кожної кнопки керування встановлюється OnTouchListener за допомогою допоміжної функції setupTouchListener.

Приватна допоміжна функція setupTouchListener() призначає OnTouchListener переданій кнопці(button). Вона визначає дві команди: commandStart (для надсилання при натисканні кнопки – MotionEvent.ACTION_DOWN) та commandStop (для надсилання при відпусканні кнопки – MotionEvent.ACTION_UP або при скасуванні дотику – MotionEvent.ACTION_CANCEL). При виникненні відповідної події дотику викликається функція sendCommand з відповідною командою. Це дозволяє реалізувати логіку утримування кнопки для руху.

Функція sendCommand() відповідає за надсилання рядкової команди на ESP32. Перед надсиланням вона перевіряє, чи активне з'єднання через BluetoothConnectionManager.isConnected(). Якщо з'єднання втрачено (і команда не є однією з команд зупинки, які варто спробувати надіслати навіть при підозрі на розрив), активність закривається. Надсилання команди відбувається асинхронно в корутині, щоб не блокувати основний потік. Використовується метод BluetoothConnectionManager.sendMessage(). У випадку невдалого надсилання (якщо це не команда STOP_ALL з onDestroy), виводиться повідомлення про помилку, і активність також буде закрита.

Метод onDestroy() викликається системою перед знищенням активності. У цій функції, якщо Bluetooth-з'єднання все ще активне, спочатку надсилається команда CMD_STOP_ALL на ESP32, щоб гарантувати зупинку автомобіля. Після невеликої паузи (для того, щоб команда встигла надіслатися) викликається BluetoothConnectionManager.closeConnection() для коректного закриття Bluetooth-сокета та звільнення ресурсів.

3.4 Тестування та результати

3.4.1 Тестування дистанційного керування

Для тестування підключення потрібен мобільний пристрій з встановленим додатком та ввімкнена модель. На рисунку 3.13 на дисплеї моделі можна побачити відображення надпису Ready to connect, що позначає готовність до підключення, а на екрані смартфона відкритий додаток.

Рисунок 3.13 – Тестування підключення

На рисунку 3.14 можна побачити запит додатка в операційної системи на включення Bluetooth на пристрої. Під час першого запуску додатка він також вимагає доступ до Bluetooth в цілому.

Рис 3.14 – Запит на використання Bluetooth

Якщо підключення вдалося, дисплей на моделі очищується (для того, щоб надалі відображати стрілки направлення руху). Тим часом у додатку головний екран змінюється на екран керування. Зміни можна побачити на рисунку 3.15. З цього моменту можна керувати моделлю натискаючи на кнопки керування.

Рисунок 3.15 – Результат підключення до моделі

Для тестування рухів та відображення модель було встановлено на імпровізований стенд. У результаті на рисунках 3.16 та 3.17 можна побачити роботу задньої та передньої осей та відповідне зображення напрямку руху на дисплеї. При натисканні кнопок керування у додатку, на модель надсилається відповідний сигнал. Мікроконтролер оброблює сигнал належним чином та вмикає потрібний мотор, а також проводить відображення напрямку руху на дисплеї (в тому числі виконується миготіння конуса стрілки). Щодо повертань варто зазначити, що вони мають більший пріоритет відображення аніж рух вперед та назад. Це зроблено для того, щоб при одночасному натисканні кнопок вперед/назад та поворотів, відображалися самі повороти, відображаючи коректний напрямок руху. Після повороту передня вісь повертається у нейтральне положення за допомогою резинок для стабілізації.

Якщо модель вимкнути, додаток повернеться до MainActivity, та можна буде побачити спливаюче повідомлення про те, що зв’язок розірвано.

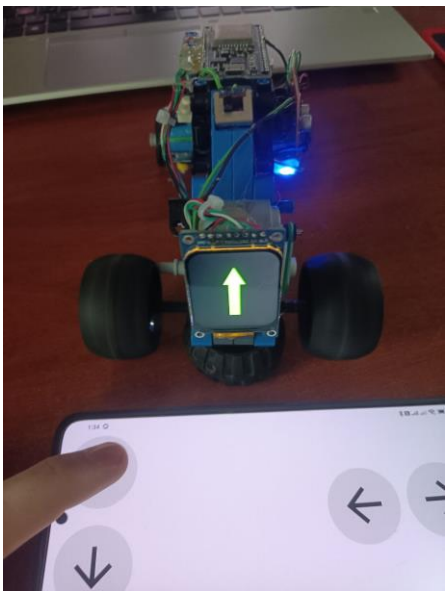

Рисунок 3.16 – Демонстрація руху вперед та відповідного відображення на дисплеї

Рисунок 3.17 – Демонстрація руху вправо та відповідного відображення на дисплеї

3.4.2 Тестування інших функцій

Для тестування максимальної швидкості було зроблено тестовий стенд, який складається зі штатива з камерою, двох позначок на відстані 1 м одна від одної та секундоміру (рисунок 3.18).

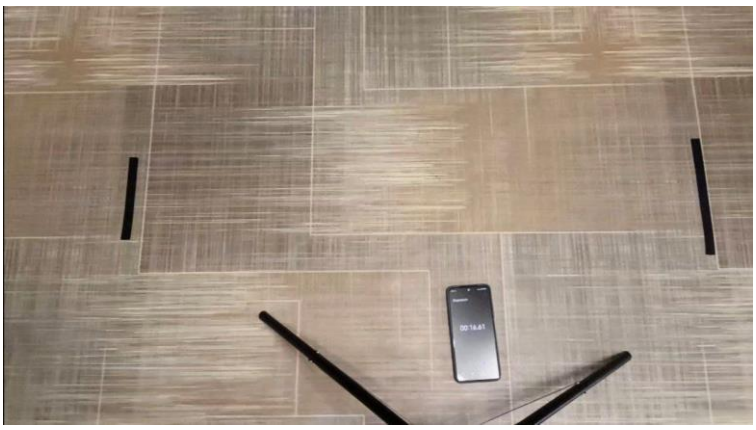

Рисунок 3.18 – Стенд для визначення максимальної швидкості

Ідея полягає в тому, щоб виміряти час, за який модель проїжджає 1 метр та знайти швидкість. Для вимірювання було вирішено взяти моменти дотику передніх коліс до стрічки. Отже, початкове значення часу – 18.62 с. (рисунок 3.19), а кінцеве – 19.42 с. (рисунок 3.20).

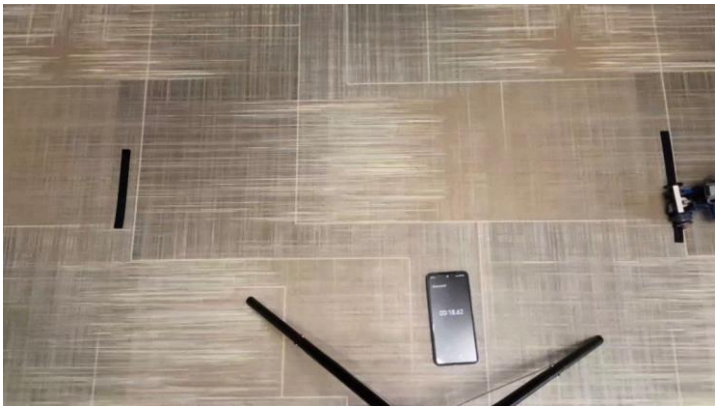

Рисунок 3.19 – Вимірювання часу дотику до першої стрічки

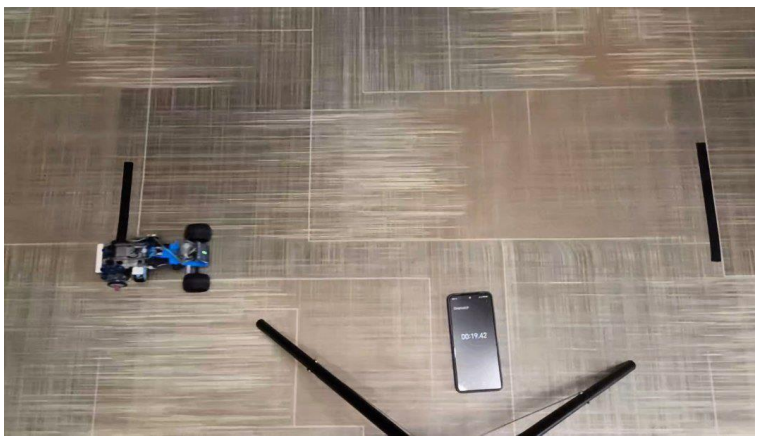

Рисунок 3.20 – Вимірювання часу дотику до другої стрічки

Різниця у часі становить 19.42 с. - 18.62 с. = 0.8 с. Швидкість моделі – 1/0.8 = 1.25 метра на секунду, тобто 4.5 км/год.

Далі – тестування надійності з'єднання. Під час тестування було виявлено що модель може підтримувати зв'язок на відстані до 50 м. З подальшим збільшенням відстані моделі починає втрачати зв'язок та згодом зовсім відключається від мобільного пристрою. Оскільки в моделі немає камер, дистанційне керування на відстані понад 10 м стає складним через те що потрібно візуально бачити модель, щоб правильно нею керувати.

Для вимірювання максимального часу роботи при повному заряді акумулятора було вирішено зафіксувати модель в одному положенні та ввімкнути задню вісь коліс. В результаті вимірювання було виявлено що максимальний час роботи від акумулятора становить 30 – 45 хвилин залежно від активності використання.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

Для початку керування моделлю потрібно:

- встановити мобільний додаток, якщо його ще не встановлено;
- надати додатку всі необхідні дозволи;
- при кожному запуску додатка, якщо функцію Bluetooth вимкнено, додаток буде вимагати у користувача підтвердити ввімкнення Bluetooth, отже потрібно погодитися;
- далі модель потрібно ввімкнути за допомогою вимикача, що знаходиться у задній частині (рисунок 4.1);
- після ініціалізації моделі, коли на її екрані з’явиться надпис Ready to connect, що значить, що модель запустилася та можна підключатися;
- натиснути кнопку Connect to CarcontrollerESP32 у додатку та дочекатися з’єднання;
- після успішного з’єднання на екрані з’являться кнопки для керування моделлю, та все готово до роботи;

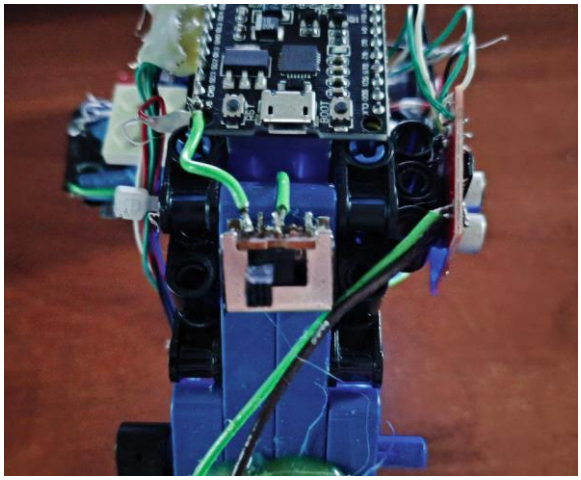

Рисунок 4.1 – Вимикач у задній частині моделі

ВИСНОВКИ

Дана кваліфікаційна робота була спрямована на розробку Bluetooth-керованої моделі автомобіля на базі мікроконтролера серії ESP. Основна мета - створення функціонального прототипу, керованого з мобільного додатку, була успішно досягнута.

Проект охоплював кілька ключових етапів, включаючи аналіз існуючих рішень, проектування фізичної структури, схеми з’єднання апаратних компонентів та архітектури програмного забезпечення. Актуальність теми, особливо в контексті розробки наземних дронів та дистанційно керованих систем, забезпечила сильну мотивацію для роботи та розвитку практичних навичок.

Важливими рішеннями при проектуванні були вибір мікроконтролера ESP32 завдяки його інтегрованому Bluetooth і можливостям обробки даних, прийняття гібридної конструкції корпусу з використанням конструктора Lego і перероблених компонентів для адаптивності та економічної ефективності, а також включення основних електронних модулів, таких як драйвер двигуна MX1508 і системи управління живленням. Дисплей був успішно інтегрований для забезпечення візуального зворотного зв'язку про робочий стан моделі та поточний напрямок руху.

На етапі реалізації було виготовлено кузов автомобіля, в якому були застосовані механічні рішення, такі як редуктор для підвищення крутного моменту на задній осі та самоцентрувальний рульовий механізм для передньої осі. Електронні компоненти були зібрані та підключені відповідно до розробленої апаратної схеми. Розробка програмного забезпечення включала створення прошивки для ESP32 за допомогою Arduino IDE для управління Bluetooth-зв'язком, управління двигуном і функціями дисплея, а також розробку спеціального додатку для управління з Android на мові Kotlin за допомогою Android Studio.

Тестування підтвердило працездатність системи. Було продемонстровано успішне Bluetooth-з'єднання між мобільним додатком та ESP32, що дозволило дистанційно керувати основними рухами автомобіля (вперед, назад, вліво, вправо). Вбудований дисплей правильно візуалізував стан з'єднання та активний напрямок руху. Функціональні тести дали змогу виміряти швидкість моделі, практичний діапазон з'єднання Bluetooth та час роботи акумулятора в різних умовах експлуатації. Також було підтверджено, що функція самоцентрування рульового механізму працює так, як було задумано.

На завершення, розроблена модель автомобіля, керованого через Bluetooth, слугує функціональним прототипом, який ефективно демонструє фундаментальні принципи дистанційного керування та інтегрованого дизайну системи. Ця робота надала цінний практичний досвід розробки вбудованих систем, бездротового зв'язку та створення мобільних додатків - навички, які можуть бути застосовані в майбутньому в робототехніці та автоматизації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Team Losi Racing. RC-Hobby [Електронний ресурс]. – Режим доступу: https://rc-hobby.com.ua/infocenter/obzory-i-stati/team-losi-racing_-komanda-gonshchikov_chempionov/ - 14.05.2025р

2. The 10 Best Arduino Lego Projects. ALL3DP [Електронний ресурс]. – Режим доступу: https://all3dp.com/2/arduino-lego-projects-kits/ - 14.05.2025р.

3. Is Arduino outdated? Quora [Електронний ресурс]. – Режим доступу: https://www.quora.com/Is-Arduino-outdated - 14.05.2025р.

4. ESP32 vs ESP8266 – Pros and Cons. MakerAdvisor [Електронний ресурс]. – Режим доступу: https://makeradvisor.com/esp32-vs-esp8266/ - 14.05.2025р.

5. Interfacing MX1508 DC Motor Driver with Arduino. Circuit Digest [Електронний ресурс]. – Режим доступу: https://circuitdigest.com/microcontroller-projects/interfacing-mx1508-dcmotor-driver-with-arduino - 15.05.2025р.

6. Samsung ICR18650-24E Cell Specifications. Second life storage [Електронний ресурс]. – Режим доступу: https://secondlifestorage.com/index.php?threads/samsung-icr18650-24e-cell-specifications.1767/ - 15.05.2025р.

7. Charger module with TP4056 controller. lygte-inf [Електронний ресурс]. – Режим доступу: https://lygte-info.dk/review/Review%20Charger%20TP4056%20UK.html - 15.05.2025р.

8. Serial Peripheral Interface. Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Serial_Peripheral_Interface - 16.05.2025р.

9. Recipes to Begin, Expand, and Enhance Your Projects. Arduino Cookbook [Текст] / M. Margolis. - 3rd ed. - O'Reilly Media, 2020. - 795 с.

10. Android Studio. Wikipedia [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Android_Studio - 15.05.2025р.

11. Редуктор. Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D1%83%D0%BA%D1%82%D0%BE%D1%80 – 25.05.2025р.