

Хмельницький національний університет

Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Кіберфізична система пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій

Назва теми

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Шифр, назва

Спеціальність 123 «Комп'ютерна інженерія»

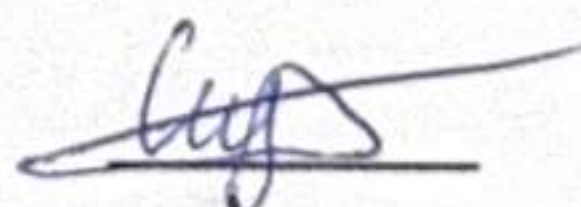
Шифр, назва

Освітня програма «Комп'ютерна інженерія та програмування»

Назва

Шифр КвРКІ 022097.22.01.107 ПЗ

Виконав здобувач IV курсу, група KI2-22-
1



Підпис

Олександр
СУХОВІРСЬКИЙ
Ініціали, прізвище

Керівник

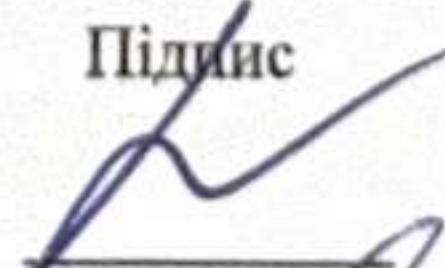


Підпис

Олександр
ГУРАЛЬНИК
Ініціали, прізвище

Нормоконтролер

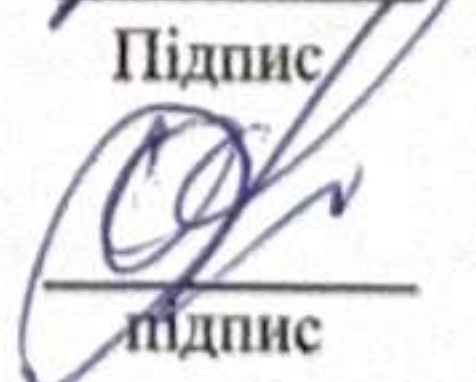
Науковий ступінь, учене звання



Підпис

Тетяна КИСІЛЬ
Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС
«01» червня 2026 р.



Підпис

Ольга ПАВЛОВА
Ініціали, прізвище

дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ПЕРШИЙ (БАКАЛАВРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Завідувачка кафедри КІІС



Ольга ПАВЛОВА

“ 10 ” 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Суховірському Олександр Сергійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій

Керівник проекту (роботи) Гуральник Олександр Борисович.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) _____

Дослідження предметної області та постановка задачі

Проектування кіберфізичної системи пріоритетного пропуску спецтранспорту

Програмна реалізація, моделювання та тестування кіберфізичної системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) _____

Принципова схема кіберфізичної системи

Функційна схема кіберфізичної системи

Структурна схема кіберфізичної системи

Емуляція програмного забезпечення в середовищі Wokwi

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « 10 » 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напрямку дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – вибір компонентів для проектування системи пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій	01.04.2026	виконано
5	Робота над розділом 3 – проектування системи пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2026	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач

Підпис

Олександр СУХОВІРСЬКИЙ

Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи

Підпис

Олександр ГУРАЛЬНИК

Ім'я, ПРІЗВИЩЕ

[illegible]

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно–апаратної система моніторингу параметрів комп’ютерного обладнання».

Автор роботи: Олександр СУХОВІРСЬКИЙ.

Керівник роботи: Олександр ГУРАЛЬНИК.

Пояснювальна записка: 84 с., 10 рис., 12 табл., 5 дод., 46 джерел.

Графічна частина: 4 креслення.

АЛГОРИТМ КЕРУВАННЯ, ESP32, ІНТЕЛЕКТУАЛЬНА ТРАНСПОРТНА СИСТЕМА, КІБЕРФІЗИЧНА СИСТЕМА, MQTT, ПРІОРИТЕТНИЙ ПРОПУСК, RFID, СВІТЛОФОРНЕ КЕРУВАННЯ.

Кваліфікаційна робота бакалавра присвячена розробленню та дослідженню кіберфізичної системи пріоритетного пропуску спеціалізованого транспорту на регульованих перехрестях. Актуальність теми зумовлена необхідністю скорочення часу прибуття екстрених служб до місця події, зменшення затримок руху та підвищення рівня автоматизації сучасних інтелектуальних транспортних систем. Забезпечення безперешкодного проїзду автомобілів швидкої допомоги, пожежної служби та поліції є важливим завданням для підвищення ефективності функціонування міської транспортної інфраструктури.

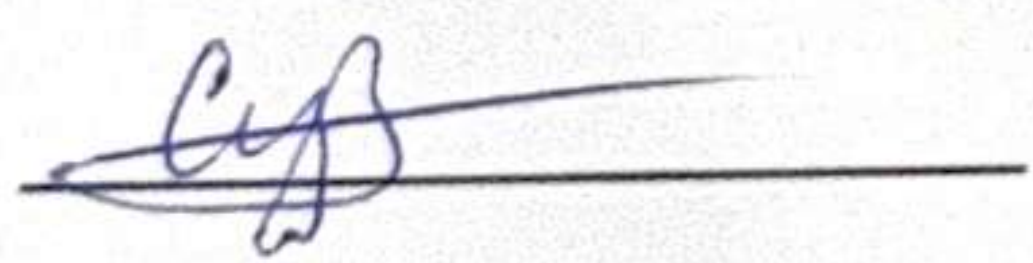
Метою роботи є проєктування, програмна реалізація та тестування кіберфізичної системи, яка забезпечує автоматичне формування пріоритетного режиму роботи світлофорного об’єкта на основі інформації про наближення спеціалізованого транспортного засобу. Для досягнення поставленої мети було виконано аналіз сучасних підходів до побудови інтелектуальних транспортних систем, обґрунтовано вибір мікроконтролерної платформи ESP32, розроблено архітектуру системи, алгоритм адаптивного керування світлофорними фазами, структуру інформаційної взаємодії на основі MQTT–протоколу та веб–інтерфейс моніторингу.

У роботі реалізовано імітаційну модель системи, що складається з трьох взаємодіючих модулів на базі ESP32: модуля диспетчеризації, який генерує

екстрені виклики та керує чергою завдань; модуля «Автомобіль», який моделює рух спеціалізованого транспортного засобу та передає дані про відстань до перехрестя; та модуля «Перехрестя», який виконує перевірку RFID-ідентифікаторів, аналізує активні запити та керує світлофорною індикацією. Для організації обміну даними використано MQTT-брокер, а для візуалізації стану системи та ручного керування розроблено веб-інтерфейс оператора.

Програмну реалізацію виконано мовою C++ із використанням Arduino Framework, бібліотек WiFi.h та PubSubClient. Моделювання роботи системи проведено у середовищі Wokwi, що дозволило створити віртуальну схему на базі ESP32 та протестувати взаємодію всіх компонентів без використання фізичного обладнання. Веб-інтерфейс реалізовано засобами HTML, CSS та JavaScript із використанням MQTT over WebSocket.

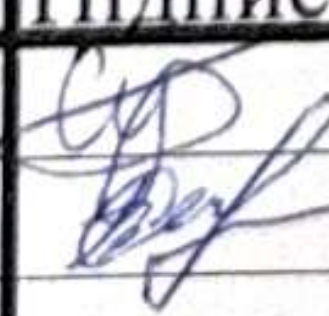
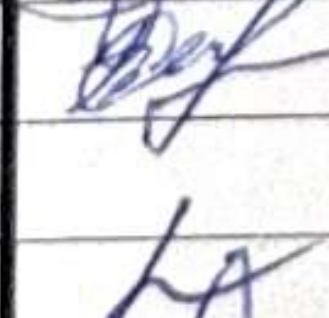
У результаті роботи створено працездатну імітаційну модель кіберфізичної системи пріоритетного пропуску спецтранспорту, яка забезпечує автоматичне формування «зеленого коридору», підтримує обробку множинних викликів, роботу внутрішніх черг та ручне керування через веб-інтерфейс. Проведене тестування підтвердило коректність закладених алгоритмів і можливість використання розробленої системи як основи для подальшого створення фізичного програмно-апаратного прототипу та впровадження в інтелектуальні транспортні системи міської інфраструктури.


Підпис здобувача

30.05.2026
Дата

ЗМІСТ

Вступ	4
1 Дослідження предметної області та постановка задачі	5
1.1 Аналіз предметної області: Проблема затримки спецтранспорту	5
1.2 Статистика та вплив затримок спецтранспорту на ефективність реагування	6
1.3 Інтелектуальні транспортні системи та концепція Smart City	8
1.4 Порівняння існуючих рішень пріоритетного пропуску спецтранспорту	9
1.5 Аналіз технічних засобів реалізації	12
1.6 Огляд сучасних апаратно-програмних платформ та технологій для реалізації систем пріоритетного керування	14
1.7 Визначення вимог та технічного завдання	16
1.7.1 Функціональні вимоги	16
1.7.2 Нефункціональні вимоги	17
1.7.3 Обмеження	18
1.8 Постановка задачі	18
2 Проектування кіберфізичної системи пріоритетного пропуску спецтранспорту	20
2.1 Розроблення загальної архітектури кіберфізичної системи	20
2.2 Проектування апаратної конфігурації системи на основі ESP32	25
2.3 Проектування підсистеми модуля диспетчеризації	28
2.4 Проектування підсистеми модуля автомобіля	30
2.5 Розроблення алгоритму адаптивного керування світлофорними фазами	32
2.6 Проектування підсистеми модуля «Перехрестя»	36
2.7 Проектування інформаційної взаємодії між вузлами системи	38
2.8 Проектування веб-інтерфейсу моніторингу стану перехрестя	42

					КВРКІ.022097.22.01.107 ПЗ			
Зм.	Арк.	№докум.	Підпис	Дата	Кіберфізична сиестмеа пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій	Літера	Аркуш	Аркушів
Виконав		Олександр СУХОВІРСЬКИЙ		01.06		у		
Перевір.		Олександр ГУРАЛЬНИК		01.06			2	72
Н.контр.		Тетяна КИСІЛЬ		01.06		ХНУ КІ2-22-1		
Затвер.		Ольга ПАВЛОВА		01.06				

2.9 Висновки до другого розділу..... 43

3 Програмна реалізація, моделювання та тестування кіберфізичної системи 45

3.1 Реалізація віртуальної схеми системи на основі ESP32 у середовищі Wokwi 45

3.2 Реалізація програмного забезпечення модуля диспетчеризації..... 48

3.3 Реалізація програмного забезпечення модуля «Автомобіль» 54

3.4 Реалізація програмного забезпечення модуля «Перехрестя» 59

3.5 Реалізація інформаційної взаємодії та MQTT–обміну 64

3.6 Реалізація веб–інтерфейсу моніторингу стану перехрестя 66

3.7 Тестування роботи системи та аналіз результатів моделювання 69

3.8 Висновки до третього розділу 75

Висновки..... 78

Перелік джерел посилань 80

Додаток А Структурна схема кіберфізичної системи 85

Додаток Б Функційна схема кіберфізичної системи 86

Додаток В Емуляція програмного забезпечення в середовищі Wokwi..... 87

Додаток Г Принципова схема кіберфізичної системи 88

Додаток Д Код програмного забезпечення 89

ВСТУП

Сучасні міські транспортні системи характеризуються високою інтенсивністю руху, що призводить до заторів, збільшення часу реагування екстрених служб та зниження загального рівня безпеки дорожнього руху. Особливо серйозною є проблема затримки спеціального транспорту: карет швидкої допомоги, пожежних автомобілів та поліцейських служб – під час проходження регульованих перехресть. Навіть кілька додаткових хвилин затримки можуть мати вирішальне значення для збереження життя людей та мінімізації наслідків надзвичайних ситуацій.

Розвиток кіберфізичних систем, Інтернету речей (IoT) та бездротових комунікацій відкриває нові можливості для автоматизації керування дорожнім рухом. Інтеграція сенсорних пристроїв, вбудованих обчислювальних платформ та мережевих технологій дозволяє створювати інтелектуальні транспортні рішення, здатні адаптуватися до реального стану дорожньої ситуації. Одним із перспективних напрямів є впровадження систем пріоритетного пропуску спецтранспорту, які автоматично ідентифікують транспортний засіб та змінюють сигнали світлофорів для забезпечення безперешкодного проїзду.

Актуальність даної роботи зумовлена необхідністю підвищення ефективності роботи екстрених служб, зменшення ризиків дорожньо–транспортних пригод та впровадження інтелектуальних транспортних систем у міське середовище. Розробка кіберфізичної системи пріоритетного пропуску є важливим кроком у напрямку цифровізації дорожньої інфраструктури та підвищення рівня громадської безпеки.

Метою кваліфікаційної роботи є проєктування, програмна реалізація та моделювання кіберфізичної системи пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 із використанням MQTT–взаємодії та імітаційних засобів ідентифікації.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області: проблема затримки спецтранспорту

Однією з найбільших проблем сучасних міських транспортних систем є затримка руху спеціального транспорту на регульованих перехрестях. Карети швидкої медичної допомоги, пожежно-рятувальні підрозділи та поліцейські служби виконують завдання, що безпосередньо пов’язані із збереженням життя людей та ліквідацією надзвичайних ситуацій. Час реагування екстрених служб є головним показником ефективності системи безпеки міста, і навіть незначні затримки можуть мати серйозні наслідки [1].

У великих містах інтенсивність транспортних потоків постійно зростає, що призводить до перевантаження дорожньої інфраструктури. Затори на міських дорогах залишаються однією з найважливіших проблем сучасної урбаністики. Попри численні наукові дослідження протягом останніх двох десятиліть, оптимізація роботи світлофорних об’єктів на рівні всієї транспортної мережі досі залишається відкритою науковою проблемою [2]. Регульовані перехрестя, які працюють за фіксованими або напіваадаптивними циклами світлофорів, не завжди враховують появу спецтранспорту. Ефективність транспортних потоків значною мірою залежить від роботи світлофорної сигналізації, тоді як розширення існуючої інфраструктури є неможливим через просторові, економічні та екологічні обмеження [3]. У результаті екстрені служби змушені чекати зміни сигналу або здійснювати ризиковані маневри, що підвищує ймовірність дорожньо-транспортних пригод.

Традиційні методи організації дорожнього руху орієнтовані на середні статистичні показники транспортних потоків і не забезпечують динамічного пріоритету для окремих категорій транспорту [4]. У більшості випадків відсутня автоматизована система ідентифікації спеціальних транспортних засобів у реальному часі, що унеможлиблює оперативне керування світлофорними

об’єктами [5]. Автомобільні виробники розробляють бортові сенсори та системи їхнього застосування в різних сферах, включаючи безпеку, керування дорожнім рухом та інформаційно–розважальні сервіси [6]. Людський фактор, затримки в передачі інформації та обмежені можливості існуючих систем керування створюють додаткові ризики для безпечного руху [7].

Проблема ускладнюється необхідністю балансування між пріоритетом спецтранспорту та загальною пропускнуою здатністю перехрестя [8]. Неконтрольоване втручання в цикл світлофора може призвести до хаотичного руху, утворення нових заторів та зниження ефективності транспортної мережі. Тому сучасні підходи до вирішення цієї проблеми передбачають використання інтелектуальних систем керування, здатних аналізувати дорожню ситуацію в режимі реального часу та забезпечувати керований пріоритет без порушення загальної логіки руху [9].

У містах також виникають проблеми з доступом спецтранспорту до місця подій у періоди несприятливих погодних умов [10]. Аналіз сезонних транспортних ситуацій показує, що сильні опади та снігові явища можуть спричиняти значне порушення транспортних потоків, зростання кількості дорожньо–транспортних пригод і ускладнення пересування екстрених служб. Несприятливі погодні фактори, такі як дощ, сніг і туман, суттєво впливають на видимість та зчіпні властивості дорожнього покриття, що змінює поведінку водіїв і знижує середню швидкість руху. Як наслідок, збільшується час реагування екстрених служб та загальний ризик аварійних ситуацій [11].

1.2 Статистика та вплив затримок спецтранспорту на ефективність реагування

Екстрені служби, такі як швидка медична допомога, пожежні підрозділи та поліція, мають важливі часові вимоги для доїзду до місця події. Несвоєчасний проїзд через регульовані перехрестя значно збільшує середній час реагування,

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

що безпосередньо впливає на результати медичних випадків та безпеку громадян. Дослідження показують, що дорожні затори можуть додавати в середньому до 10 хвилин додаткового часу реагування для служб екстреної допомоги під час виїздів у міських умовах, що значно перевищує рекомендовані стандарти якості обслуговування екстрених викликів [12–14].

Одним із важливих показників є вплив світлофорів на час очікування. Аналіз даних показав, що середня затримка для швидких на червоне світло у деяких міських сценаріях зафіксовано на понад 15–20 хвилин у залежності від інтенсивності руху та загального стану дорожньої мережі, що ще більше підкреслює необхідність пріоритетних систем керування [15, 16].

Дані багатьох досліджень також демонструють, що впровадження спеціальних пріоритетних систем для екстрених служб здатне суттєво скоротити час реагування. Наприклад, звіти від впровадження адаптивних пріоритетних світлофорів показують зменшення часу реакції служб на 15–50 %, а також до 70 % зниження частоти аварій за участю службового транспорту [17, 18].

У контексті безпеки дорожнього руху статистичні дані свідчать про значну кількість аварій за участю екстрених автомобілів: у 2023 році загальна кількість смертей у ДТП за участю аварійних служб становила 198 осіб [19], що також підкреслює необхідність удосконалення дорожньої інфраструктури для безпечного переміщення таких транспортних засобів [20, 21].

Результати численних досліджень і практичних впроваджень підтверджують, що основною причиною затримок є статичні цикли регулювання світлофорів, які не враховують наявності та наближення спецтранспорту, що призводить не лише до втрат часу, але й до потенційно фатальних наслідків у критичних ситуаціях [22].

1.3 Інтелектуальні транспортні системи та концепція Smart City

Інтелектуальні транспортні системи (Intelligent Transport Systems, ITS) розглядаються як комплекс інтегрованих апаратно–програмних рішень, що поєднують телекомунікації, сенсорні мережі та автоматизовані механізми керування з метою підвищення безпеки та ефективності транспортної інфраструктури [23]. Вони забезпечують збір, обробку та аналіз даних у режимі реального часу, що створює передумови для оптимізації транспортних потоків і зменшення заторів у міських умовах [24].

Згідно з міжнародними дослідженнями, впровадження ITS пов’язується зі зниженням аварійності, скороченням часу поїздок та підвищенням пропускної здатності дорожньої мережі [25]. Особливе значення мають адаптивні системи керування світлофорами, здатні динамічно реагувати на зміни транспортної ситуації. Такі системи використовують сенсорні дані, алгоритми прогнозування та мережеву взаємодію між інфраструктурними вузлами [26].

Інтелектуальні транспортні системи розглядаються як складова ширшої концепції Smart City, яка передбачає інтеграцію цифрових технологій у міське середовище з метою підвищення якості життя населення та ефективності використання ресурсів [27]. У межах цієї концепції транспортна інфраструктура трактується як кіберфізична система, де фізичні об’єкти пов’язані з цифровими платформами керування через мережі передачі даних [28].

Дослідження підкреслюють, що концепція Smart City орієнтована не лише на автоматизацію, а й на створення адаптивних систем, здатних до самоаналізу та оптимізації функціонування в режимі реального часу [29]. Транспортні підсистеми в таких середовищах використовують бездротові комунікації, хмарні сервіси та розподілені обчислення для керування світлофорами, моніторингу руху та підтримки пріоритетних режимів для екстрених служб [30].

Автоматизація регульованих перехресть розглядається як один із пріоритетних напрямів розвитку інтелектуального міського транспорту.

Адаптивні світлофорні системи дозволяють забезпечувати пріоритет спецтранспорту без суттєвого порушення загальної структури руху, що позитивно впливає на безпеку та час реагування екстрених служб [31]. Інтеграція таких рішень у міську інфраструктуру розглядається як важливий елемент сучасної транспортної політики розумних міст [32].

1.4 Порівняння існуючих рішень пріоритетного пропуску спецтранспорту

Системи пріоритетного пропуску екстрених транспортних засобів активно досліджуються в межах інтелектуальних транспортних систем та Smart City–платформ. У сучасних містах використовуються різні технологічні підходи до виявлення спецтранспорту та адаптивного керування світлофорами: GPS–моніторинг, RFID–ідентифікація, відеоаналіз, IoT–сенсори та мережі підключених транспортних засобів.

Більшість існуючих рішень орієнтовані або на централізовані системи керування, або на складну інфраструктуру, що потребує значних фінансових витрат. Крім того, багато систем не враховують реальний стан транспортних потоків або не забезпечують достатню швидкість реагування.

Для аналізу було розглянуто декілька типових підходів, що застосовуються в сучасних дослідженнях і практичних системах.

Зокрема, увага зосереджена на технологіях, що варіюються від супутникового моніторингу до прямих комунікаційних протоколів між транспортним засобом та дорожньою інфраструктурою. Це дозволяє порівняти їхню ефективність за такими параметрами, як час затримки сигналу, вартість розгортання та стійкість до зовнішніх чинників.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1 – Порівняння рішень						
Рішення	Технологія виявлення	Тип системи	Реальний час	Вартість впровадження	Масштабованість	Недоліки
GPS–based priority control [41]	GPS + центр керування	Централізована	1-5	Висока	Висока	Затримки передачі даних
Video detection systems [42]	Камери + AI	Локальна	1–5с	Дуже висока	Низька	Чутливість до погоди та освітлення
RFID smart intersections [43]	RFID	Децентралізована	100–500мс	Середня	Середня	Потрібна інфраструктура міток
IoT smart traffic lights [44]	IoT сенсори	Гібридна	10–50мс	Середня	Висока	Складність інтеграції
V2X communication systems [45]	V2X / C–V2X	Мережна	50–200мс	Дуже висока	Висока	Потребує повної міської інфраструктури
					КвРКІ.022097.22.01.107 ПЗ	
Зм.	Арк.	№ докум.	Підпис	Дата		
						Арк. 10

Рішення на основі GPS–пріоритету широко використовуються в системах громадського транспорту та екстрених служб. Одним із прикладів є GPS–based Transit Signal Priority (TSP), що досліджувався у роботі [41], де транспортні засоби передають своє місцезнаходження до централізованого серверу керування. Система аналізує положення транспорту та коригує цикл світлофора. Такий підхід дозволяє реалізувати глобальне керування, проте затримки передачі даних через мобільні мережі та залежність від серверної інфраструктури знижують ефективність у критичних ситуаціях.

Відеоаналітичні системи виявлення спецтранспорту базуються на алгоритмах комп’ютерного зору та штучного інтелекту. У роботі [42] описано систему відеоспостереження з автоматичним розпізнаванням аварійних ситуацій та адаптивним керуванням світлофорами. Перевагою такого підходу є відсутність необхідності встановлення додаткових пристроїв на транспортних засобах. Проте стабільність роботи суттєво залежить від погодних умов, освітлення та якості відеопотоку, що обмежує застосування в реальному міському середовищі.

RFID–рішення передбачають встановлення ідентифікаційних міток на спецтранспорті та зчитувачів на перехрестях. У дослідженні [43] продемонстровано прототип інтелектуального світлофора, що змінює сигнали при наближенні транспортного засобу з RFID–міткою. Такий підхід забезпечує швидку ідентифікацію та малу затримку реакції, однак потребує розгортання інфраструктури міток і зчитувачів.

ІоТ–орієнтовані світлофорні системи поєднують сенсорні мережі, бездротовий зв’язок та локальні контролери. Огляд ІоТ–платформ для транспортного керування наведено в роботі [44], де підкреслюється ефективність розподілених рішень із локальною обробкою даних. Такі системи дозволяють реагувати на зміну трафіку практично миттєво, проте складність інтеграції з існуючою інфраструктурою залишається суттєвим викликом.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

Системи V2X–комунікації представляють найбільш технологічно розвинений підхід. У дослідженні [45] порівнюються стандарти ITS–G5 та C–V2X, які дозволяють транспортним засобам безпосередньо обмінюватися даними з інфраструктурою. Це забезпечує мінімальну затримку та високу надійність зв’язку. Недоліком є необхідність повного покриття міста спеціалізованою комунікаційною мережею, що суттєво збільшує вартість впровадження.

Порівняльний аналіз в таблиці 1.1 показує, що більшість існуючих рішень або потребують дорогої централізованої інфраструктури, або залежать від складних систем відеоаналізу та міських мереж зв’язку.

1.5 Аналіз технічних засобів реалізації

Для реалізації кіберфізичної системи пріоритетного пропуску спецтранспорту необхідно забезпечити ідентифікацію транспортних засобів, збір даних із транспортних сенсорів, обмін інформацією в режимі реального часу та адаптивне керування світлофорними вузлами. Вибір технологій здійснювався з урахуванням вимог до швидкодії, надійності, масштабованості та можливості створення лабораторного прототипу [32, 33].

У процесі проєктування кіберфізичних систем пріоритетного пропуску спецтранспорту головну роль відіграє вибір апаратної платформи керування. Сучасні рішення базуються на мікроконтролерах сімейства ESP32, які відрізняються за продуктивністю, енергоспоживанням та рівнем підтримки криптографічних механізмів. Модулі ESP32–WROOM–32 широко використовуються як універсальна платформа для вбудованих систем завдяки двоядерній архітектурі та інтегрованим бездротовим інтерфейсам. Мікроконтролери ESP32–S3 орієнтовані на системи підвищеної безпеки та підтримують апаратне прискорення шифрування, що є важливим для захисту каналів передачі службових сигналів. Більш економним варіантом є ESP32–C3

на архітектурі RISC–V, який доцільно використовувати у допоміжних вузлах або транспортних маяках з обмеженим енергоспоживанням.

Вибір каналу бездротового зв’язку визначає швидкість реагування системи та радіус її дії. Технологія ESP–NOW забезпечує прямий обмін даними між пристроями без використання маршрутизаторів і характеризується малою латентністю, що придатно для систем реального часу в межах одного перехрестя. Протоколи LoRa орієнтовані на далекодійний зв’язок із низьким енергоспоживанням і можуть використовуватися для міжвузлової координації на рівні міської мережі. Bluetooth Low Energy застосовується у системах короткого радіусу дії з мінімальним енергоспоживанням, однак його дальність обмежує використання в умовах швидкого руху транспорту.

Одним з головних елементів є метод ідентифікації спецтранспорту. Найпростішим підходом є використання активних RFID або NFC–міток, які автоматично зчитуються при вході в зону дії перехрестя. Альтернативним рішенням є цифрова автентифікація на основі криптографічних токенів, коли транспортний вузол передає зашифровані пакети даних із використанням апаратної підтримки AES, що унеможливлює несанкціоноване втручання. Перспективним напрямком є геопросторова активація (геофенсинг), при якій пріоритет вмикається на основі координат транспортного засобу та попередньо визначених зон перехрестя.

Інтеграція з виконавчими механізмами світлофорного обладнання може реалізовуватися різними способами залежно від типу інфраструктури. Для лабораторних прототипів доцільно використовувати релейні модулі, що фізично керують сигнальними лініями. У промислових системах застосовуються промислові протоколи обміну даними, зокрема RS–485 або Modbus, які дозволяють взаємодіяти з існуючими контролерами без втручання у силові ланцюги. Альтернативним методом є емуляція датчиків транспортних петель, коли система імітує наявність транспортного потоку, змушуючи світлофор змінити фазу без прямого керування лампами.

Архітектура кіберфізичної системи передбачає взаємодію декількох функціональних блоків. Транспортний вузол містить мікроконтролер, модуль позиціонування та бездротовий передавач, який транслює ідентифікаційні дані та статус транспортного засобу. Вузол перехрестя приймає сигнал, виконує автентифікацію та формує команду керування світлофорним контролером. Зворотний канал зв'язку дозволяє передавати підтвердження активації пріоритету, що підвищує надійність системи та інформує водія про стан перехрестя.

Аналіз існуючих технічних рішень показує, що ефективна система пріоритетного пропуску повинна поєднувати низьку латентність, локальну автономність та можливість масштабування без необхідності складної централізованої інфраструктури.

1.6 Огляд сучасних апаратно–програмних платформ та технологій для реалізації систем пріоритетного керування

Реалізація кіберфізичних систем пріоритетного пропуску спецтранспорту передбачає використання мікроконтролерних платформ, бездротових технологій зв'язку, засобів ідентифікації та програмних інструментів моделювання. Сучасні дослідження та прототипи в цій галузі базуються на поєднанні вбудованих IoT–рішень із технологіями реального часу.

Однією з найбільш поширених апаратних платформ для побудови вбудованих систем є мікроконтролери сімейства ESP32. Вони поєднують достатню обчислювальну продуктивність із вбудованими бездротовими інтерфейсами Wi-Fi та Bluetooth, а також підтримкою криптографічних механізмів [34]. Завдяки цьому ESP32 широко застосовується в IoT–системах, включаючи транспортні та інфраструктурні рішення, де важливими є енергоефективність, автономність та стабільність функціонування [35].

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

Для ідентифікації об'єктів у транспортних системах активно використовується технологія RFID. Вона забезпечує безконтактне автоматичне розпізнавання транспортних засобів у зоні дії зчитувача та характеризується малою затримкою обробки сигналу [36]. У транспортних і логістичних системах RFID застосовується як надійний механізм контролю доступу та відстеження руху об'єктів [37], що робить її придатною для сценаріїв пріоритетного пропуску.

У частині бездротового зв'язку для систем локальної взаємодії розглядаються протоколи прямого обміну даними між пристроями, зокрема ESP–NOW, який забезпечує низьку латентність і не потребує розгортання мережевої інфраструктури [38]. Для задач із розширеним радіусом дії використовується технологія LoRa, що забезпечує значну дальність передачі даних при низькому енергоспоживанні [39]. Поєднання різних каналів зв'язку дозволяє адаптувати систему як для локального рівня перехрестя, так і для інтеграції в міську інфраструктуру.

Обробка даних у таких системах, як правило, здійснюється локально на вбудованому контролері, що відповідає принципам кіберфізичних систем і дозволяє зменшити затримки керування [27]. Такий підхід забезпечує автономність роботи вузла перехрестя та знижує залежність від зовнішніх серверів.

Для розробки та тестування вбудованих систем активно застосовуються середовища моделювання апаратної частини. Одним із таких інструментів є платформа Wokwi, яка дозволяє симулювати роботу мікроконтролерів та електронних схем без фізичного обладнання [40]. Це спрощує прототипування та зменшує ризик апаратних помилок на ранніх етапах розробки.

Візуалізація стану інфраструктурних об'єктів у режимі реального часу є невід'ємною складовою інтелектуальних транспортних систем. Використання веб–інтерфейсів для моніторингу стану перехресть відповідає сучасним

підходам інтеграції транспортних підсистем у межах концепції Smart City [26, 31].

1.7 Визначення вимог та технічного завдання

На основі аналізу існуючих рішень та обґрунтування вибору технічних засобів сформовано систему вимог до кіберфізичної системи пріоритетного пропуску спецтранспорту. Визначення вимог дозволяє формалізувати функціональні можливості системи, її обмеження та критерії ефективності.

1.7.1 Функціональні вимоги

Система повинна забезпечувати:

Автоматичну ідентифікацію спеціального транспорту при його наближенні до регульованого перехрестя.

Перевірку автентичності сигналу з метою запобігання несанкціонованому доступу до механізму пріоритету.

Збір даних із сенсорів транспортних потоків (інтенсивність руху, стан фаз світлофора).

Локальну обробку отриманої інформації на вбудованому контролері.

Формування керуючої дії для зміни фази світлофорного об’єкта з метою створення «зеленого коридору».

Бездротову взаємодію між транспортним вузлом та вузлом перехрестя.

Передачу службової інформації у режимі, наближеному до реального часу.

Візуалізацію поточного стану перехрестя та режимів роботи системи через веб–інтерфейс.

1.7.2 Нефункціональні вимоги

До нефункціональних вимог належать:

Швидкодія. Система повинна забезпечувати своєчасне передавання повідомлень між модулями та оперативне формування керуючих дій для зміни світлофорних фаз. Передавання даних між вузлами здійснюється за допомогою MQTT-протоколу через бездротову мережу Wi-Fi, що забезпечує функціонування системи в режимі, наближеному до реального часу.

Надійність. Програмні модулі повинні забезпечувати стабільну роботу в умовах тривалого моделювання, підтримувати автоматичне повторне підключення до Wi-Fi мережі та MQTT-брокера після тимчасового розриву зв'язку, а також коректно обробляти помилки та некоректні повідомлення.

Безпека. Система повинна виконувати перевірку допустимості спеціалізованого транспортного засобу шляхом порівняння переданого RFID-ідентифікатора з внутрішнім списком дозволених значень, що запобігає активації пріоритетного режиму для невідомих транспортних засобів.

Масштабованість. Архітектура системи повинна забезпечувати можливість розширення шляхом додавання нових транспортних засобів, перехресть або модулів моніторингу без необхідності суттєвої зміни основної логіки роботи.

Модульність. Програмна реалізація повинна бути структурована у вигляді окремих функціональних компонентів, що відповідають за диспетчеризацію, моделювання руху транспортного засобу, керування перехрестям та візуалізацію стану системи.

Портативність. Система повинна використовувати стандартні програмні засоби та бібліотеки Arduino Framework (WiFi.h, PubSubClient), що забезпечує можливість перенесення програмного забезпечення на інші плати сімейства ESP32.

1.7.3 Обмеження

Під час розроблення імітаційної моделі враховуються такі обмеження:

- використання мікроконтролерної платформи ESP32 як базового керуючого вузла;
- реалізація програмної імітації RFID–ідентифікації без застосування фізичних RFID–зчитувачів;
- використання бездротового зв’язку через Wi-Fi та MQTT–протокол;
- моделювання апаратної частини у середовищі Wokwi;
- відсутність інтеграції з реальними міськими світлофорними контролерами;
- використання світлодіодної індикації замість промислового світлофорного обладнання;
- застосування публічного MQTT–брокера для організації інформаційної взаємодії;
- реалізація системи у вигляді лабораторної імітаційної моделі.

Сформовані вимоги визначають рамки розроблення та є основою для формалізованої постановки задачі створення кіберфізичної системи пріоритетного пропуску спеціалізованого транспорту.

1.8 Постановка задачі

Проведений аналіз існуючих систем пріоритетного пропуску спецтранспорту показав, що сучасні рішення або потребують складної централізованої інфраструктури, або базуються на дорогих системах відеоаналізу та мережевих сервісах. Такі підходи характеризуються високою вартістю впровадження, залежністю від стабільності мережі та обмеженою адаптивністю в умовах реального міського середовища. У більшості випадків

відсутня децентралізована архітектура, здатна локально приймати рішення в режимі реального часу.

У зв’язку з цим формулюється задача розробки кіберфізичної системи пріоритетного пропуску спецтранспорту, що забезпечує локальне прийняття рішень, мінімальну затримку реагування та можливість масштабування без складної централізованої інфраструктури.

Метою роботи є створення прототипу адаптивної системи керування регульованим перехрестям, здатної автоматично виявляти спецтранспорт, аналізувати поточний стан транспортних потоків, формувати «зелений коридор» у режимі реального часу та забезпечувати моніторинг і візуалізацію стану системи. Для досягнення поставленої мети необхідно вирішити такі задачі:

- 1. Розробити архітектуру децентралізованої кіберфізичної системи.
- 2. Реалізувати механізм ідентифікації спецтранспорту;
- 3. Розробити алгоритм формування пріоритетної фази світлофора;
- 4. Забезпечити локальну обробку даних у режимі реального часу;
- 5. Реалізувати програмну частину системи на платформі ESP32;
- 6. Змодельовати роботу системи у середовищі Wokwi;
- 7. Провести тестування сценаріїв пріоритетного пропуску та оцінити часові характеристики системи.

Результатом роботи має стати працездатна імітаційна модель кіберфізичної системи, що демонструє принципи адаптивного керування транспортним перехрестям і може бути використана як основа для подальшого створення фізичного програмно–апаратного прототипу.

2

ПРОЄКТУВАННЯ

КІБЕРФІЗИЧНОЇ

СИСТЕМИ

ПРІОРИТЕТНОГО ПРОПУСКУ СПЕЦТРАНСПОРТУ

2.1 Розроблення загальної архітектури кіберфізичної системи

Розроблення загальної архітектури кіберфізичної системи пріоритетного пропуску спецтранспорту виконується на основі вимог, сформульованих у першому розділі роботи. Основними вимогами до системи є забезпечення роботи в режимі реального часу, мінімізація затримок передавання даних, надійність функціонування, можливість масштабування та підтримка взаємодії між вузлами транспортної інфраструктури.

Під час проєктування було розглянуто централізований, децентралізований та гібридний підходи до побудови архітектури системи. Централізовані системи забезпечують глобальне керування транспортними потоками, проте характеризуються високою залежністю від серверної інфраструктури та стабільності мережевого з'єднання. Повністю децентралізовані системи забезпечують автономність вузлів, однак ускладнюють організацію моніторингу та координації роботи системи.

У цій роботі використано гібридну розподілену архітектуру, у якій рішення щодо перемикання світлофорних фаз приймаються локально модулем «Перехрестя» на базі ESP32, а модуль диспетчеризації, MQTT-брокер та веб-інтерфейс оператора забезпечують передавання повідомлень, диспетчеризацію та моніторинг стану системи. Такий підхід дозволяє поєднати переваги локального керування в режимі реального часу із можливістю централізованого спостереження за роботою транспортної інфраструктури.

У межах проєктованої системи доцільно виділити два основні рівні: рівень кіберфізичних вузлів та мережевий і прикладний рівень.

До рівня кіберфізичних вузлів належать модуль «Автомобіль» на базі ESP32, модуль «Перехрестя» ESP32, модуль диспетчеризації ESP32, а також локальна індикація світлофорів. До мережевого та прикладного рівня належать

MQTT-брокер і веб-інтерфейс оператора. Загальну структурну схему кіберфізичної системи пріоритетного пропуску спецтранспорту наведено на рисунку 2.1.

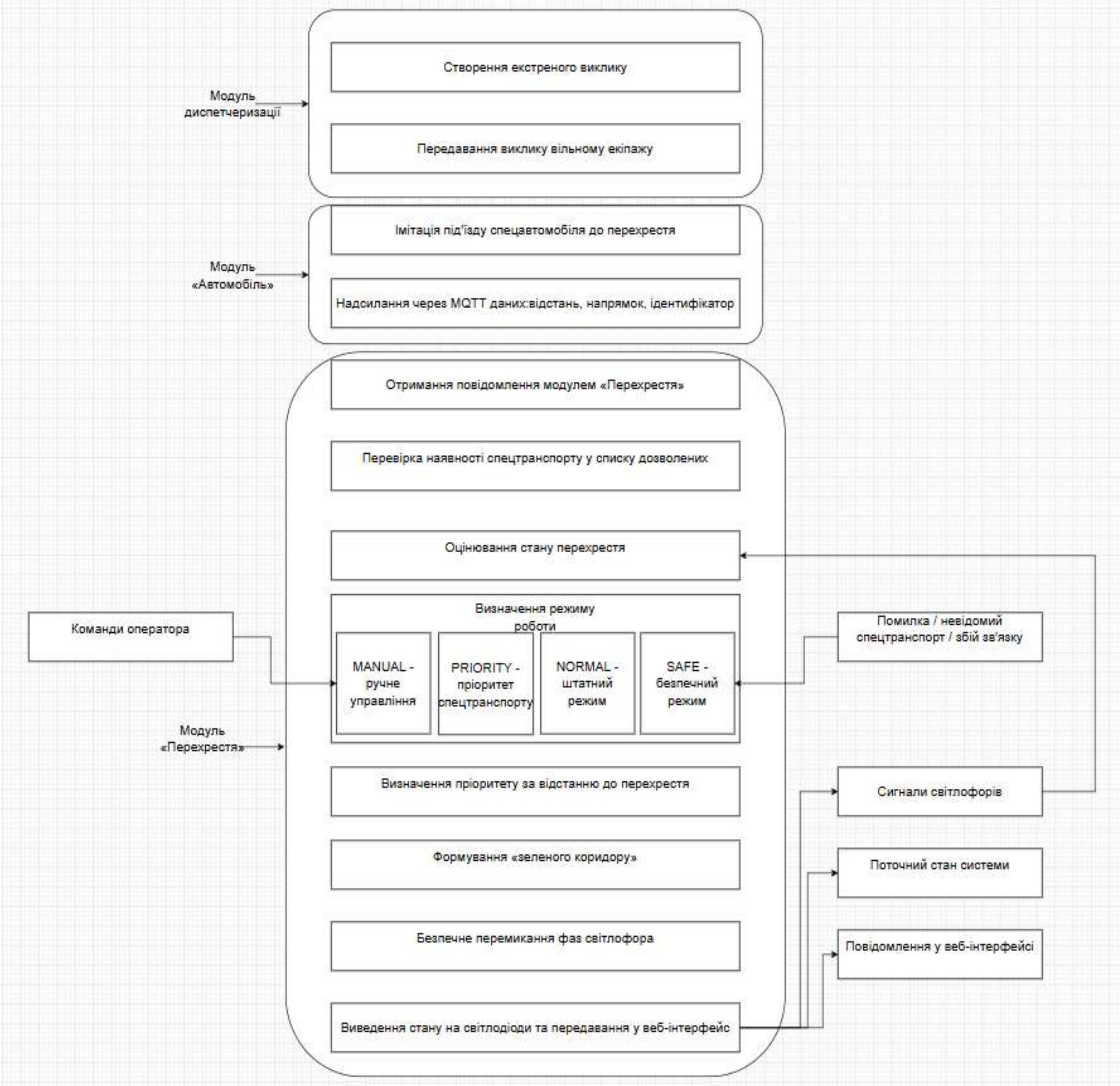


Рисунок 2.1 – Структурна схема кіберфізичної системи пріоритетного пропуску спецтранспорту

Модуль диспетчеризації ESP32 виконує функції формування та передавання службових повідомлень про виникнення екстрених викликів. У межах імітаційної моделі в середовищі Wokwi цей модуль використовується для

генерації даних про активацію спецтранспорту та передачі їх через MQTT–брокер іншим вузлам системи.

Модуль «Автомобіль» ESP32 виконує функції імітації спеціального транспортного засобу та формує MQTT–повідомлення про наближення до перехрестя. У повідомленнях передаються ідентифікатор транспортного засобу, тип екстреної служби, напрям руху та відстань до перехрестя. У межах цієї роботи RFID–ідентифікація та визначення координат спецтранспорту не реалізуються фізичними модулями, а моделюються програмно шляхом передавання відповідних параметрів у MQTT–повідомленні.

Модуль «Перехрестя» ESP32 є основним вузлом локального керування. Його функціями є приймання повідомлень від модуля «Автомобіль», аналіз стану системи, визначення необхідності активації пріоритетного режиму та формування керувальних сигналів для світлофорної індикації. Прийняття рішення виконується локально без необхідності звернення до інших вузлів системи, що дозволяє мінімізувати затримки реакції.

Локальна індикація світлофорів використовується для моделювання роботи регульованого перехрестя. Вона забезпечує відображення поточного стану транспортних фаз та демонстрацію переходу системи до режиму пріоритетного пропуску спецтранспорту.

MQTT–брокер виконує функції проміжного вузла обміну повідомленнями між компонентами системи. Використання MQTT–брокерної моделі обміну повідомленнями дозволяє реалізувати асинхронний обмін даними, спростити масштабування системи та організувати взаємодію між вузлами без прямого мережевого з’єднання між ними.

Веб–інтерфейс оператора призначений для візуалізації стану системи в режимі реального часу. Через веб–інтерфейс відображаються активні транспортні вузли, стан світлофорів, повідомлення про активацію пріоритетного режиму та журнали подій системи.

Для відображення послідовності виконання основних функцій системи розроблено функційну схему, наведену на рисунку 2.2.

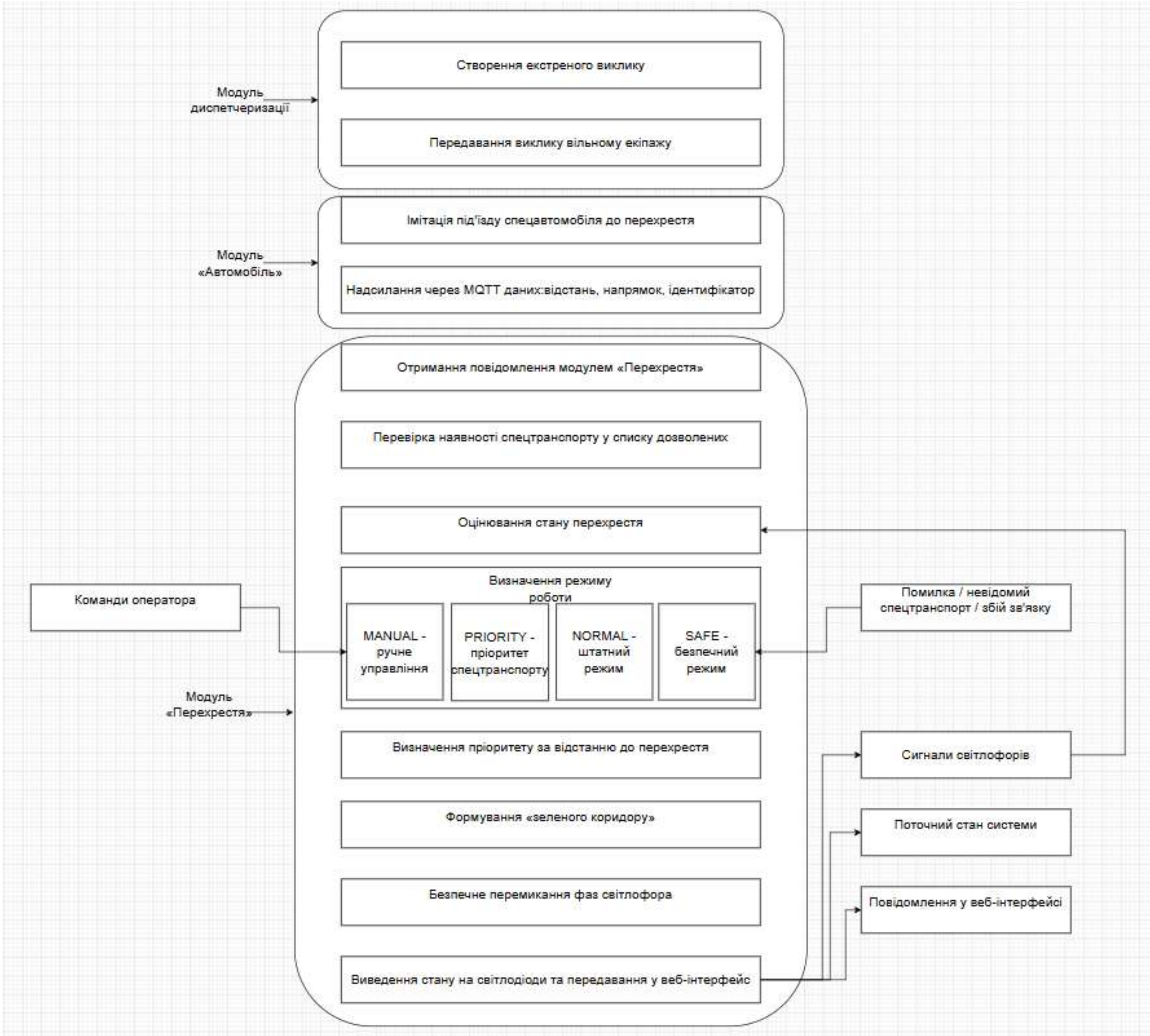


Рисунок 2.2 – Функційна схема кіберфізичної системи пріоритетного пропуску спецтранспорту

Взаємодія між компонентами системи реалізується шляхом передавання MQTT-повідомлень через мережеву інфраструктуру. Основні інформаційні потоки системи наведено в таблиці 2.1.

Таблиця 2.1 – Основні інформаційні потоки системи

Інформаційний потік	Джерело	Приймач	Зміст повідомлення
Дані виклику	Модуль диспетчеризації ESP32	Модуль «Автомобіль» ESP32	Ідентифікатор виклику, тип служби, ідентифікатор транспортного засобу
Дані про наближення спецтранспорту	Модуль «Автомобіль» ESP32	Модуль «Перехрестя» ESP32	Ідентифікатор транспорту, напрям руху, відстань до перехрестя
Керувальні сигнали світлофорної індикації	Модуль «Перехрестя» ESP32	Локальна індикація світлофорів	Сигнали керування червоним, жовтим і зеленим світлодіодами
Дані моніторингу	Модуль «Перехрестя» ESP32	Веб–інтерфейс оператора	Стан світлофорів, режим роботи системи, активні транспортні засоби
Команди оператора	Веб–інтерфейс оператора	Модуль «Перехрестя» ESP32	Ручний режим, вибір фази світлофора, рівень навантаження
Службові MQTT–повідомлення	MQTT–брокер	Усі вузли системи	Передавання повідомлень між вузлами системи

Запропонована архітектура забезпечує можливість локального прийняття рішень у режимі реального часу, підтримує масштабування шляхом додавання нових вузлів та відповідає принципам побудови сучасних кіберфізичних систем

і інтелектуальної транспортної інфраструктури. Крім того, використання MQTT–брокерної моделі та імітаційної моделі у середовищі Wokwi дозволяє реалізувати тестування системи без необхідності використання реальної транспортної інфраструктури.

2.2 Проєктування апаратної конфігурації системи на основі ESP32

Проєктування апаратної конфігурації кіберфізичної системи пріоритетного пропуску спецтранспорту є важливим етапом розроблення, оскільки саме апаратна платформа визначає можливість роботи системи в режимі реального часу, стабільність бездротового обміну даними та ефективність локального керування транспортною інфраструктурою.

Під час вибору апаратної платформи було розглянуто декілька варіантів реалізації системи, зокрема мікроконтролери сімейства Arduino, одноплатні комп’ютери Raspberry Pi та мікроконтролери ESP32.

Платформи Arduino характеризуються простотою використання та низькою вартістю, однак мають обмежені обчислювальні ресурси та не оснащені вбудованими засобами бездротового зв’язку. Для організації мережевої взаємодії необхідне використання додаткових модулів Wi-Fi або Bluetooth, що ускладнює структуру системи та збільшує кількість апаратних компонентів.

Одноплатні комп’ютери Raspberry Pi забезпечують вищу продуктивність і підтримку повноцінних операційних систем, однак їх використання є менш доцільним для задач локального керування в режимі реального часу через складність програмної реалізації, вище енергоспоживання та більшу вартість.

У цій роботі як основну апаратну платформу обрано мікроконтролер ESP32. Ця платформа поєднує достатню обчислювальну потужність, підтримку бездротових технологій Wi-Fi та Bluetooth, а також можливість одночасного виконання декількох задач завдяки двоядерній архітектурі. Використання ESP32

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

дозволяє реалізувати взаємодію між вузлами системи без застосування додаткових мережевих модулів.

Апаратна конфігурація системи формується відповідно до архітектури, наведеної у підрозділі 2.1, та включає модуль диспетчеризації ESP32, модуль «Автомобіль» ESP32, модуль «Перехрестя» ESP32, локальну індикацію світлофорів, MQTT-брокер і веб-інтерфейс оператора. Основні апаратні компоненти системи наведено в таблиці 2.2.

Таблиця 2.2 – Компоненти імітаційної моделі системи

Компонент	Призначення	Спосіб використання в моделі
ESP32 модуля диспетчеризації	Формування повідомлень про виклик спецтранспорту	Передавання MQTT–повідомлень про активацію виклику
ESP32 модуля «Автомобіль»	Імітація спеціального транспортного засобу	Формування MQTT–повідомлень про наближення до перехрестя
ESP32 модуля «Перехрестя»	Локальне керування світлофорними фазами	Аналіз MQTT–повідомлень та формування керувальних сигналів
Світлодіоди світлофорної індикації	Відображення режимів роботи світлофора	Імітація роботи транспортного перехрестя
MQTT–брокер	Організація обміну повідомленнями	Передавання MQTT–повідомлень між вузлами
Веб–інтерфейс оператора	Моніторинг стану системи	Відображення стану вузлів і режимів роботи

Модуль диспетчеризації ESP32 виконує функцію генерації повідомлень про виникнення екстреної ситуації. У межах імітаційної моделі у середовищі Wokwi цей модуль передає MQTT-повідомлення з інформацією про активацію виклику, тип екстреної служби та напрямок руху спецтранспорту.

Модуль «Автомобіль» ESP32 використовується для імітації роботи спеціального транспортного засобу. Його основною функцією є формування та передавання MQTT-повідомлень про наближення до перехрестя. У повідомленнях передаються ідентифікатор транспортного засобу, тип служби, напрям руху та відстань до перехрестя.

У межах цієї роботи RFID-ідентифікація та визначення координат спецтранспорту не реалізуються фізичними модулями, а моделюються програмно шляхом передавання ідентифікатора, напрямку руху та відстані до перехрестя в MQTT-повідомленні.

Модуль «Перехрестя» ESP32 є центральним елементом локального керування системою. Даний вузол приймає MQTT-повідомлення від модуля «Автомобіль», аналізує стан транспортної ситуації та приймає рішення щодо активації пріоритетного режиму.

Апаратна конфігурація модуля «Перехрестя» включає світлодіодну індикацію, яка моделює роботу транспортного світлофора. Для кожного напрямку руху використовується три світлодіоди:

- червоний сигнал;
- жовтий сигнал;
- зелений сигнал.

У межах моделі реалізовано чотири напрямки руху, тому загальна кількість світлодіодів становить дванадцять. Світлодіоди групуються відповідно до напрямків руху та підключаються до цифрових виходів ESP32. Зміна стану світлодіодів відображає перемикання фаз світлофора та активацію режиму пріоритетного пропуску спецтранспорту.

MQTT–брокер використовується для організації асинхронного обміну повідомленнями між вузлами системи. Передавання даних реалізується через MQTT–топіки, що дозволяє розділити інформаційні потоки між окремими компонентами системи. Такий підхід спрощує масштабування архітектури та забезпечує незалежність вузлів один від одного.

Веб–інтерфейс оператора використовується для візуалізації стану системи в режимі реального часу. Через веб–інтерфейс відображаються повідомлення про активацію спецтранспорту, стан світлофорів та режими роботи системи.

Для реалізації та тестування системи використовується імітаційна модель у середовищі Wokwi. Дане середовище дозволяє моделювати MQTT–взаємодію, яка реалізується через підключення ESP32 до зовнішнього MQTT–брокера з використанням Wi-Fi–можливостей середовища Wokwi.

Оскільки в межах роботи система реалізується як імітаційна модель, питання енергоживлення не є предметом практичного тестування. Водночас під час переходу до фізичного прототипу модулі ESP32 можуть живитися від стабілізованого джерела або акумулятора з урахуванням вимог до автономності та стабільності роботи.

Запропонована апаратна конфігурація відповідає вимогам до побудови кіберфізичної системи пріоритетного пропуску спецтранспорту, забезпечує підтримку MQTT–взаємодії між вузлами та дозволяє реалізувати локальне керування світлофорними фазами в режимі реального часу.

2.3 Проєктування підсистеми модуля диспетчеризації

Підсистема модуля диспетчеризації призначена для формування екстрених подій та ініціації роботи кіберфізичної системи пріоритетного пропуску спецтранспорту. Вона забезпечує створення інформаційних повідомлень про виникнення надзвичайної ситуації та передачу їх до інших вузлів системи через мережеву інфраструктуру.

Відповідно до загальної архітектури системи, модуль диспетчеризації не виконує функцій керування світлофорними фазами або аналізу дорожньої ситуації. Його роль обмежується генерацією подій і розподілом викликів між транспортними засобами, що відповідає принципу розподіленої кіберфізичної системи з локальним прийняттям рішень на рівні перехрестя.

У межах реалізації даної роботи модуль диспетчеризації функціонує як програмно–апаратний компонент на базі ESP32, який взаємодіє з іншими вузлами через MQTT–протокол. Використання брокерної моделі обміну повідомленнями забезпечує асинхронну передачу даних та дозволяє масштабувати систему без зміни її структури.

Основною функцією підсистеми є генерація екстрених викликів. Кожен виклик формується з використанням таких параметрів:

- унікальний ідентифікатор виклику;
- тип екстреної служби (швидка допомога, пожежна служба, поліція);
- ідентифікатор транспортного засобу, що призначається на виклик.

Для передачі інформації про виклик у системі використовується структурований текстовий формат повідомлення, у якому параметри розділяються спеціальним символом. У межах реалізації застосовується наступний формат MQTT–повідомлення:

CALL;emergencyId;vehicleId;serviceType;rfidTag

- CALL – ідентифікатор типу повідомлення;
- emergencyId – унікальний ідентифікатор виклику;
- vehicleId – ідентифікатор транспортного засобу, призначеного на виклик;
- serviceType – тип екстреної служби;
- rfidTag – ідентифікаційний тег транспортного засобу.

Такий формат повідомлення обрано з урахуванням простоти реалізації на мікроконтролерах та ефективності обробки даних. Розділення параметрів символом дозволяє виконувати швидкий розбір повідомлення без використання

складних бібліотек обробки JSON, що є доцільним для вбудованих систем із обмеженими ресурсами.

Для забезпечення безперервності роботи системи в підсистемі передбачено механізм керування доступністю транспортних засобів. У разі відсутності вільних машин сформовані виклики не втрачаються, а зберігаються у внутрішній черзі.

У межах імітаційної моделі черга викликів реалізується як програмна структура даних у пам'яті модуля диспетчеризації ESP32. Для цього використовується масив записів, кожен елемент якого містить інформацію про ідентифікатор виклику, тип екстреної служби та час створення події. Після появи доступного транспортного засобу відповідного типу модуль диспетчеризації аналізує вміст черги та призначає на виконання найстаріший виклик.

Після завершення обслуговування попередніх викликів транспортний засіб повертається до списку доступних і може бути повторно використаний для виконання нових завдань.

У межах імітаційної моделі, реалізованої в середовищі Wokwi, підсистема диспетчеризації працює в режимі генерації випадкових екстрених подій через задані інтервали часу. Це дозволяє дослідити поведінку системи в умовах динамічної зміни транспортної ситуації та перевірити коректність взаємодії між її компонентами.

2.4 Проєктування підсистеми модуля автомобіля

Підсистема модуля автомобіля призначена для моделювання поведінки спеціалізованого транспортного засобу в умовах дорожнього руху та передачі інформації про його стан до інших вузлів кіберфізичної системи. Вона забезпечує формування параметрів руху транспортного засобу та їх передачу до модуля «Перехрестя» для подальшого аналізу.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

Модуль автомобіля функціонує на базі ESP32 та взаємодіє з іншими компонентами системи через MQTT–протокол. Він отримує інформацію про призначення на виклик від модуля диспетчеризації та переходить у стан активного виконання завдання.

Основною функцією підсистеми є моделювання руху транспортного засобу до перехрестя. Для цього формуються параметри ідентифікатора транспортного засобу, типу екстреної служби, RFID-ідентифікатора, напрямку руху та відстані до перехрестя. Напрямок руху у межах даної роботи не визначається на основі реальних навігаційних даних, а моделюється програмно як випадкова величина. Такий підхід дозволяє відтворити різні сценарії під’їзду транспортного засобу до перехрестя без використання складних апаратних засобів.

Відстань до перехрестя змінюється у процесі роботи модуля та імітує рух транспортного засобу. Початкове значення задається програмно, після чого поступово зменшується до нуля, що відповідає моменту досягнення перехрестя. Значення відстані періодично передається до модуля «Перехрестя» у вигляді MQTT–повідомлень.

Передавання даних про стан транспортного засобу здійснюється періодично у процесі його руху. Зокрема, значення відстані до перехрестя передається з певним часовим інтервалом, що визначає частоту оновлення інформації у системі.

Період передавання повідомлень може змінюватися залежно від умов реалізації системи та вимог до швидкодії. Зменшення інтервалу дозволяє підвищити точність визначення положення транспортного засобу, проте призводить до збільшення навантаження на мережу та обчислювальні ресурси. Збільшення інтервалу, навпаки, зменшує навантаження, але може негативно впливати на оперативність реагування системи.

У межах імітаційної моделі, реалізованої в даній роботі, для забезпечення стабільної роботи та можливості тестування системи прийнято фіксований

період передавання повідомлень, що становить 1 секунду. Такий інтервал дозволяє забезпечити достатню точність відстеження руху транспортного засобу та коректну роботу алгоритму адаптивного керування світлофорними фазами.

В імітаційній моделі RFID-ідентифікація не реалізується апаратно, а задається як унікальний рядковий параметр, що дозволяє моделювати процес перевірки доступу транспортного засобу до системи.

Підсистема також передбачає керування станом транспортного засобу. Після завершення виконання виклику, що відповідає досягненню нульового значення відстані, транспортний засіб переходить у неактивний стан та може бути повторно використаний для обробки нових викликів після певної затримки.

2.5 Розроблення алгоритму адаптивного керування світлофорними фазами

Алгоритм адаптивного керування світлофорними фазами є найважливішим компонентом кіберфізичної системи пріоритетного пропуску спецтранспорту, оскільки саме він забезпечує оперативну зміну режимів роботи світлофорного об’єкта залежно від поточної дорожньої ситуації. На відміну від традиційних систем керування світлофорами, що функціонують за фіксованими часовими циклами, у даній роботі реалізовано адаптивний підхід, за якого рішення щодо перемикання фаз приймається на основі даних про наближення спеціалізованого транспортного засобу.

Алгоритм функціонує на модулі «Перехрестя» та використовує інформацію, що надходить від модуля «Автомобіль» у вигляді MQTT-повідомлень. Основними параметрами, які враховуються під час прийняття рішення, є ідентифікатор транспортного засобу, напрямок його руху та відстань до перехрестя. Обробка цих даних дозволяє визначити необхідність зміни світлофорної фази та сформуванати відповідну реакцію системи.

Для формалізації логіки роботи алгоритму виділено основні режими функціонування системи: NORMAL, PRIORITY, MANUAL та SAFE. Режим

NORMAL відповідає штатній роботі світлофора за циклічним алгоритмом. Режим PRIORITY активується у разі наближення спеціалізованого транспортного засобу та забезпечує його безперешкодний проїзд. Режим MANUAL дозволяє оператору здійснювати ручне керування системою через веб-інтерфейс. Режим SAFE передбачений як безпечний стан системи та реалізується через переведення світлофорної індикації у стан «усе червоне».

Таблиця 2.3 – Режими роботи системи

Режим	Умова активації	Дія системи
NORMAL	спецтранспорт не виявлено	стандартний цикл світлофора
PRIORITY	виявлено дозволений спецтранспорт у зоні реагування	формування «зеленого коридору»
MANUAL	команда оператора	виконання ручної команди
SAFE	збій зв’язку, невідомий транспорт, конфлікт команд	усі напрямки переводяться в червоний

У нормальному режимі роботи світлофор функціонує за циклічним алгоритмом, що передбачає послідовне перемикання фаз для основних напрямків руху. Такий режим забезпечує базову організацію дорожнього руху за відсутності екстрених ситуацій. При надходженні інформації про наближення спеціалізованого транспортного засобу алгоритм переходить до аналізу отриманих даних.

Головним елементом алгоритму є порівняння відстані до транспортного засобу із заданим пороговим значенням. Це значення визначається з урахуванням рівня навантаженості перехрестя, що дозволяє системі адаптуватися до різних умов дорожнього руху.

Таблиця 2.4 – Порогові значення активації пріоритетного режиму

Рівень навантаження	Порогова відстань активації, м
Низький	25
Середній	45
Високий	70

У разі одночасного надходження даних про декілька транспортних засобів формується черга запитів. Алгоритм виконує їх аналіз та визначає транспортний засіб з найвищим пріоритетом. Критерієм вибору є мінімальна відстань до перехрестя, що дозволяє забезпечити першочерговий пропуск найближчого транспортного засобу та уникнути конфліктних ситуацій.

Після визначення пріоритетного транспортного засобу алгоритм активує режим PRIORITY. На цьому етапі виконується переведення світлофорного об’єкта у безпечний стан із встановленням сигналу «усе червоне», після чого вмикається зелений сигнал для напрямку руху вибраного транспортного засобу. Одночасно призупиняється виконання стандартного циклу світлофора.

Особливістю реалізованого алгоритму є те, що повернення до нормального режиму роботи не відбувається до моменту повного проходження транспортного засобу через перехрестя. Це визначається за значенням відстані, яке у процесі руху зменшується та досягає нульового рівня. Такий підхід забезпечує стабільність роботи системи та запобігає передчасному перемиканню світлофорних фаз.

Після завершення пріоритетного пропуску алгоритм здійснює повернення до режиму NORMAL. Для цього світлофор переводиться у стан «усе червоне», після чого відновлюється циклічне перемикання фаз. Інформація про оброблений транспортний засіб видаляється з черги, що дозволяє системі перейти до обробки наступних запитів.

2.6 Проектування підсистеми модуля «Перехрестя»

Підсистема модуля «Перехрестя» є центральним елементом кіберфізичної системи пріоритетного пропуску спецтранспорту, оскільки саме вона виконує функції прийому даних, їх обробки, вибору режиму роботи та керування світлофорною індикацією. На відміну від інших модулів системи, що відповідають за генерацію подій або моделювання руху, даний модуль реалізує локальне керування у режимі реального часу.

У межах даної роботи модуль «Перехрестя» реалізовано на базі мікроконтролера ESP32, що забезпечує підтримку бездротового зв’язку, обробку вхідних повідомлень та керування виконавчими пристроями. Взаємодія з іншими вузлами системи здійснюється через MQTT–протокол.

Функціонально підсистема модуля «Перехрестя» складається з кількох взаємопов’язаних блоків, кожен з яких виконує окрему роль у загальному процесі обробки даних та керування.

Першим елементом є MQTT–приймач, який забезпечує прийом повідомлень від інших вузлів системи. Через цей блок надходять дані від модуля «Автомобіль», а також команди від веб–інтерфейсу оператора. Отримані повідомлення передаються на подальшу обробку всередині модуля.

Наступним є блок перевірки ідентифікатора, який виконує валідацію транспортного засобу. Перевірка здійснюється на основі ідентифікаційного параметра, що моделює RFID–тег. У разі невідповідності транспортний засіб ігнорується, що дозволяє запобігти несанкціонованому впливу на систему.

Після перевірки дані надходять до черги активних запитів, яка реалізована як програмна структура у пам’яті мікроконтролера. У цій черзі зберігається інформація про всі транспортні засоби, що знаходяться в зоні обслуговування. Черга постійно оновлюється відповідно до зміни відстані до транспортних засобів та очищується від застарілих записів.

Далі обробка виконується у блоці вибору режиму, який визначає поточний стан системи. На основі наявності транспортних засобів, їх відстані до перехрестя та команд оператора виконується вибір між режимами NORMAL, PRIORITY або MANUAL. У випадку невизначеного стану або помилки система переводиться у безпечний режим SAFE, що відповідає стану «усе червоне».

Вибраний режим передається до блоку керування світлофорами, який безпосередньо формує сигнали для світлодіодної індикації. Цей блок реалізує перемикання світлофорних фаз, включаючи стандартний цикл роботи та пріоритетний пропуск транспортних засобів.

Окремо реалізовано блок передавання стану, який формує інформацію про поточний стан системи та передає її до веб–інтерфейсу. Передаються такі параметри, як режим роботи, активна фаза світлофора, рівень навантаження та інформація про активні транспортні засоби.

Також у складі підсистеми передбачено блок обробки команд оператора, який забезпечує прийом і виконання команд, що надходять із веб–інтерфейсу. Цей блок дозволяє переводити систему у ручний режим, змінювати фази світлофора та встановлювати рівень навантаження.

Таким чином, внутрішня структура модуля «Перехрестя» побудована за принципом послідовної обробки даних, де кожен функціональний блок виконує чітко визначену роль. Це забезпечує модульність системи, спрощує її масштабування та дозволяє реалізувати адаптивне керування світлофорними фазами в умовах реального часу.

Узагальнену внутрішню структуру модуля «Перехрестя» наведено на рисунку 2.3.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

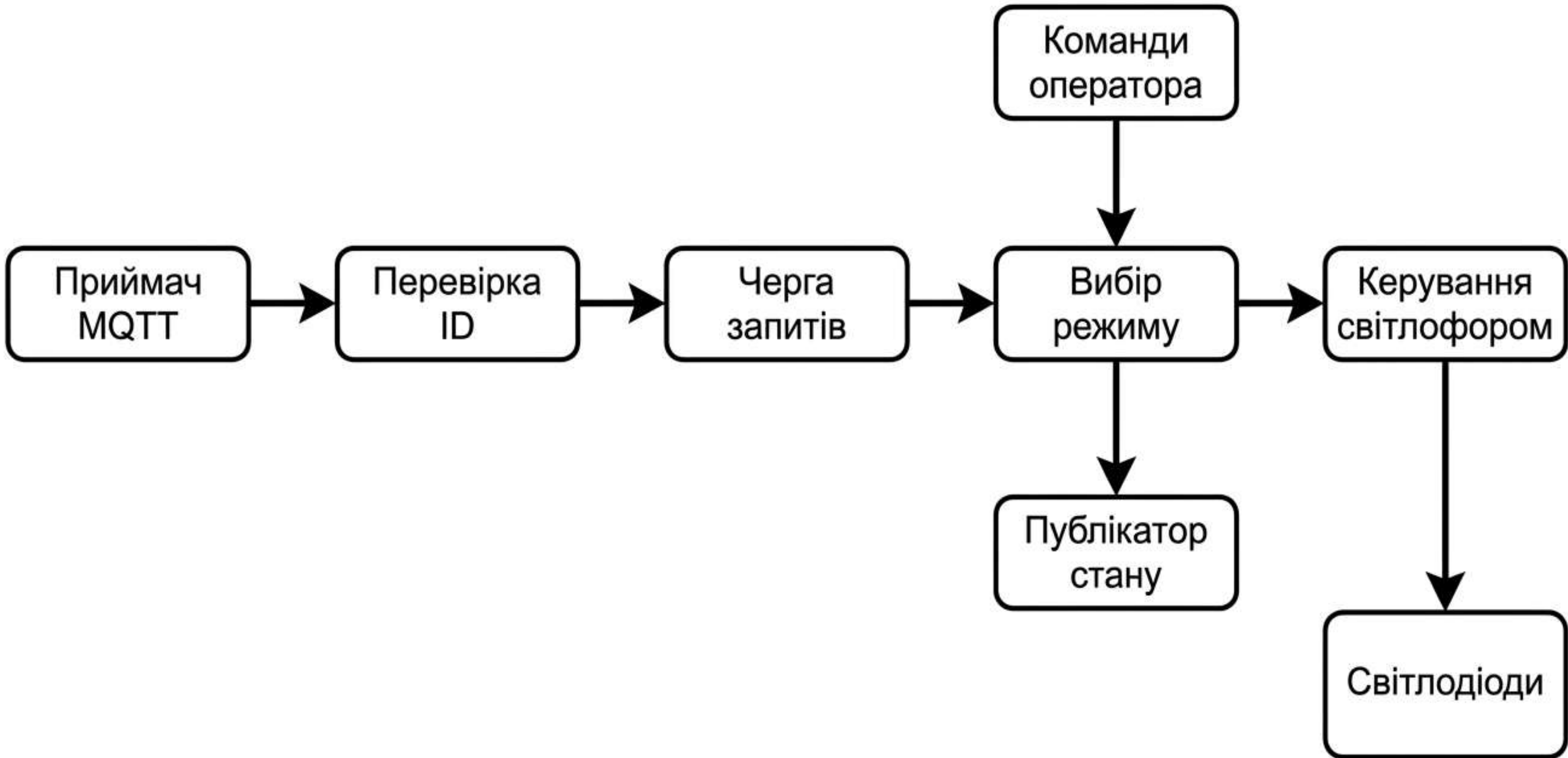


Рисунок 2.3 – Структура модуля «Перехрестя»

2.7 Проектування інформаційної взаємодії між вузлами системи

Інформаційна взаємодія між вузлами кіберфізичної системи є визначним елементом, що забезпечує узгоджену роботу всіх її компонентів. Вона визначає способи передачі даних, структуру повідомлень та принципи обміну інформацією між модулями диспетчеризації, автомобіля, перехрестя та веб-інтерфейсом.

У даній роботі інформаційна взаємодія реалізується з використанням протоколу MQTT, який широко застосовується в системах Інтернету речей. Використання цього протоколу обумовлено його легкістю, асинхронністю та можливістю роботи за моделлю «публікація–підписка», що дозволяє ефективно організувати обмін повідомленнями між розподіленими вузлами системи.

У межах обраної архітектури всі вузли системи взаємодіють через MQTT-брокер, який виступає посередником між відправниками та отримувачами повідомлень. Такий підхід забезпечує слабку зв’язаність компонентів системи, підвищує її гнучкість та спрощує масштабування.

Інформаційна взаємодія організована у вигляді набору тематичних каналів (топиків), кожен з яких відповідає за передачу певного типу даних. Основні топіки системи наведено в таблиці 2.5.

Таблиця 2.5 – MQTT–топіки системи

MQTT–топик	Відправник	Отримувач	Призначення
smartcity/dispatch/call	модуль диспетчеризації	модуль «Автомобіль»	передавання виклику
smartcity/vehicle/status	модуль «Автомобіль»	модуль «Перехрестя»	передавання даних про рух транспортного засобу
smartcity/vehicle/done	модуль «Автомобіль»	модуль диспетчеризації	повідомлення про завершення виклику
smartcity/intersection/status	модуль «Перехрестя»	веб–інтерфейс	передавання стану перехрестя
smartcity/intersection/vehicles	модуль «Перехрестя»	веб–інтерфейс	передавання списку активних транспортних засобів
smartcity/intersection/operator	веб–інтерфейс	модуль «Перехрестя»	передавання команд оператора

Модуль диспетчеризації формує повідомлення про виникнення екстрених ситуацій та передає їх до модуля «Автомобіль». У свою чергу, модуль «Автомобіль» надсилає повідомлення про свій стан до модуля «Перехрестя», включаючи інформацію про напрямок руху та відстань до перехрестя. Після завершення виклику транспортний засіб передає повідомлення про завершення, що дозволяє оновити його статус доступності.

Модуль «Перехрестя» виконує обробку отриманих даних та формує повідомлення про стан системи, які передаються до веб–інтерфейсу. Крім цього, веб–інтерфейс має можливість надсилати керуючі команди до модуля «Перехрестя», що забезпечує реалізацію ручного режиму роботи.

Для забезпечення коректної обробки даних у системі використовується структурований текстовий формат повідомлень, у якому параметри розділяються символом «;». Основні формати повідомлень наведено в таблиці 2.6.

Таблиця 2.6 – Формати MQTT–повідомлень

Тип повідомлення	Формат	Опис полів
Виклик (CALL)	CALL;emergencyId;vehicleId;serviceType;rfidTag	emergencyId – ідентифікатор виклику; vehicleId – транспортний засіб; serviceType – тип служби; rfidTag – ідентифікатор
Дані транспорту (VEHICLE)	VEHICLE;emergencyId;vehicleId;serviceType;rfidTag;direction;distance	direction – напрям руху; distance – відстань до перехрестя
Завершення (DONE)	DONE;vehicleId;serviceType;rfidTag	повідомлення про завершення виклику

Кінець таблиці 2.6

Стан перехрестя (STATUS)	STATUS;mode;phase;load;priority;vehicleId;direction	mode – режим роботи; phase – фаза світлофора; load – навантаження
Список транспорту (VEHICLES)	VEHICLES;...	перелік активних транспортних засобів та їх параметри
Команди оператора	AUTO ALL_RED NS_GREEN EW_GREEN LOAD_LOW LOAD_MEDIUM LOAD_HIGH	керування режимом та параметрами системи

Використання текстового формату повідомлень замість більш складних структур, таких як JSON, дозволяє зменшити обчислювальне навантаження на мікроконтролери та забезпечити швидку обробку даних у режимі реального часу.

Особливістю реалізації є асинхронний характер обміну інформацією. Вузли системи не потребують постійного прямого з’єднання між собою, а взаємодіють через брокер, отримуючи лише ті повідомлення, на які підписані. Це дозволяє зменшити навантаження на мережу, підвищити надійність системи та забезпечити її стабільну роботу навіть у разі тимчасових затримок передавання даних.

2.8 Проектування веб–інтерфейсу моніторингу стану перехрестя

Веб–інтерфейс моніторингу стану перехрестя призначений для візуалізації роботи кіберфізичної системи в режимі реального часу та забезпечення взаємодії оператора з системою. Його використання дозволяє отримувати актуальну

інформацію про стан світлофорного об’єкта, активні транспортні засоби та процес пріоритетного пропуску спецтранспорту.

У межах даної роботи веб–інтерфейс реалізовано у вигляді клієнтського додатку, що функціонує у браузері та взаємодіє з іншими компонентами системи через MQTT–брокер. Такий підхід дозволяє забезпечити прямий обмін даними між інтерфейсом та модулем «Перехрестя» без використання складної серверної частини.

Веб–інтерфейс виконує дві основні функції: відображення стану системи та надання можливості ручного керування. У режимі моніторингу відображається інформація про поточний режим роботи перехрестя, активну фазу світлофора, рівень навантаження та наявність пріоритетного транспортного засобу. Окрім цього, здійснюється виведення списку активних транспортних засобів із зазначенням їхніх характеристик, зокрема типу служби, напрямку руху та відстані до перехрестя.

Для підвищення наочності реалізовано візуальне відображення руху транспортних засобів відносно перехрестя. Положення автомобіля демонструється на умовній шкалі, що відповідає відстані до перехрестя. Така форма представлення дозволяє швидко оцінити поточну ситуацію та визначити, який транспортний засіб знаходиться найближче.

Функція ручного керування забезпечує можливість втручання оператора у роботу системи. Через веб–інтерфейс можуть передаватися команди до модуля «Перехрестя», зокрема для перемикання режиму роботи, встановлення сигналу «усе червоне» або примусового увімкнення зеленої фази для певного напрямку. Також передбачена можливість зміни рівня навантаження перехрестя, що впливає на параметри роботи алгоритму адаптивного керування.

Взаємодія веб–інтерфейсу з іншими вузлами системи здійснюється через MQTT–топіки. Інтерфейс підписується на канали, що містять інформацію про стан перехрестя та транспортних засобів, і обробляє отримані повідомлення для

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

їх подальшого відображення. Для передачі команд використовується окремий канал, у який публікуються відповідні керуючі повідомлення.

Особливістю реалізації є асинхронний характер оновлення даних. Інформація відображається у режимі реального часу без перезавантаження сторінки, що забезпечує зручність використання та оперативність реагування. Динамічне оновлення елементів інтерфейсу дозволяє відображати зміну стану системи без створення додаткового навантаження.

Таблиця 2.7 – Структура екрана веб–інтерфейсу моніторингу

Елемент веб–інтерфейсу	Призначення
Заголовок сторінки	Відображає назву системи та її функціональне призначення
Індикатор з’єднання	Показує стан підключення до MQTT–брокера
Блок стану перехрестя	Відображає режим роботи, активну фазу світлофора та рівень навантаження
Блок пріоритету	Показує наявність активного пріоритету, ідентифікатор транспортного засобу та напрямок руху
Панель ручного керування	Дозволяє оператору змінювати режим роботи та керувати світлофором
Панель вибору навантаження	Забезпечує встановлення рівня транспортного навантаження
Список активних транспортних засобів	Відображає всі транспортні засоби, що знаходяться в зоні обслуговування
Картка транспортного засобу	Містить інформацію про конкретний транспортний засіб (тип, напрямок, відстань)
Індикатор стану транспортного засобу	Відображає статус (очікування або пріоритет)

Кінець таблиці 2.7

Графічна шкала відстані	Візуалізує відстань транспортного засобу до перехрестя
Анімація руху транспортного засобу	Демонструє наближення автомобіля до перехрестя

2.9 Висновки до другого розділу

У другому розділі виконано комплексне проєктування кіберфізичної системи пріоритетного пропуску спецтранспорту з урахуванням теоретичних положень, розглянутих у першому розділі. Визначено структурну організацію системи, її основні компоненти та принципи взаємодії між ними.

У ході проєктування обґрунтовано використання гібридної розподіленої архітектури, у якій рішення щодо керування світлофорними фазами приймаються на локальному рівні модулем «Перехрестя». Такий підхід дозволяє зменшити затримки реагування системи та підвищити її надійність у порівнянні з централізованими моделями керування.

Розроблено структуру основних підсистем, зокрема модуля диспетчеризації, модуля «Автомобіль» та модуля «Перехрестя». Для кожної підсистеми визначено її функціональне призначення та роль у загальному процесі забезпечення пріоритетного пропуску спеціалізованого транспорту. Встановлено чіткий розподіл функцій між модулями, що дозволяє уникнути дублювання обробки даних та забезпечує узгоджену роботу всієї системи.

Сформовано внутрішню структуру модуля «Перехрестя», яка включає блоки прийому повідомлень, перевірки ідентифікаторів, зберігання активних запитів, вибору режиму роботи, керування світлофорними сигналами, обробки команд оператора та передавання стану системи. Така структура забезпечує модульність реалізації та можливість подальшого розширення функціональності.

Розроблено алгоритм адаптивного керування світлофорними фазами, який базується на аналізі параметрів руху транспортних засобів. У межах алгоритму формалізовано режими роботи системи: NORMAL, PRIORITY, MANUAL та SAFE. Це дозволяє забезпечити гнучке реагування на зміну дорожньої ситуації та підвищити безпеку функціонування системи.

Визначено порогові значення активації пріоритетного режиму залежно від рівня навантаження перехрестя, що забезпечує адаптацію системи до різних умов транспортного потоку.

Проектовано підсистему інформаційної взаємодії між вузлами системи із використанням MQTT-протоколу. Визначено набір MQTT-топиків, структуру повідомлень та логіку їх передавання, що забезпечує ефективний асинхронний обмін даними між розподіленими компонентами системи.

Окремо розроблено веб-інтерфейс моніторингу стану перехрестя, який забезпечує відображення параметрів роботи системи в режимі реального часу та надає можливість оператору здійснювати ручне керування. Це дозволяє підвищити зручність експлуатації та забезпечити додатковий рівень контролю за функціонуванням системи.

У результаті виконаного проектування сформовано цілісну модель кіберфізичної системи, яка забезпечує адаптивне керування світлофорними фазами та створює основу для подальшої реалізації та експериментального дослідження.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

3 ПРОГРАМНА РЕАЛІЗАЦІЯ, МОДЕЛЮВАННЯ ТА ТЕСТУВАННЯ КІБЕРФІЗИЧНОЇ СИСТЕМИ

3.1 Реалізація віртуальної схеми системи на основі ESP32 у середовищі Wokwi

Для перевірки працездатності розробленої кіберфізичної системи пріоритетного пропуску спецтранспорту виконано побудову її віртуальної моделі у середовищі Wokwi. Використання цього середовища дозволяє моделювати роботу мікроконтролерних систем на базі ESP32, виконувати налагодження програмного забезпечення та досліджувати взаємодію між окремими вузлами без необхідності використання реального апаратного обладнання.

Середовище Wokwi підтримує емуляцію мікроконтролерів ESP32, цифрових входів і виходів, світлодіодної індикації, кнопок та мережевої взаємодії через Wi-Fi і MQTT. Це робить його придатним для дослідження розподілених кіберфізичних систем, у яких декілька незалежних контролерів обмінюються повідомленнями через мережеву інфраструктуру.

У межах даної роботи у Wokwi реалізовано повну віртуальну модель системи, яка складається з трьох окремих мікроконтролерних вузлів на базі ESP32:

- модуля диспетчеризації;
- модуля «Автомобіль»;
- модуля «Перехрестя».

Кожен із зазначених вузлів реалізовано як окремий екземпляр ESP32 із власним програмним кодом, що виконує визначені функції та взаємодіє з іншими компонентами системи через MQTT-брокер.

Модуль диспетчеризації моделює роботу диспетчерського центру екстрених служб. Він генерує випадкові екстрені події, визначає тип необхідної служби, обирає вільний транспортний засіб та формує повідомлення про виклик.

Модуль «Автомобіль» моделює спеціалізований транспортний засіб. Після отримання виклику він автоматично генерує напрямок руху, змінює значення відстані до перехрестя та періодично передає повідомлення про свій стан.

Модуль «Перехрестя» виконує функції локального керування світлофорним об’єктом. Він приймає повідомлення від модуля «Автомобіль», перевіряє ідентифікатор транспортного засобу, аналізує чергу активних запитів, обирає пріоритетний транспортний засіб та керує світлодіодною індикацією.

Для візуального відображення роботи світлофорного об’єкта до модуля «Перехрестя» підключено світлодіоди, що імітують сигнали чотирьох напрямків руху: північного, південного, східного та західного. Для кожного напрямку використовуються окремі червоний, жовтий та зелений світлодіоди. Загальна кількість світлодіодів становить дванадцять.

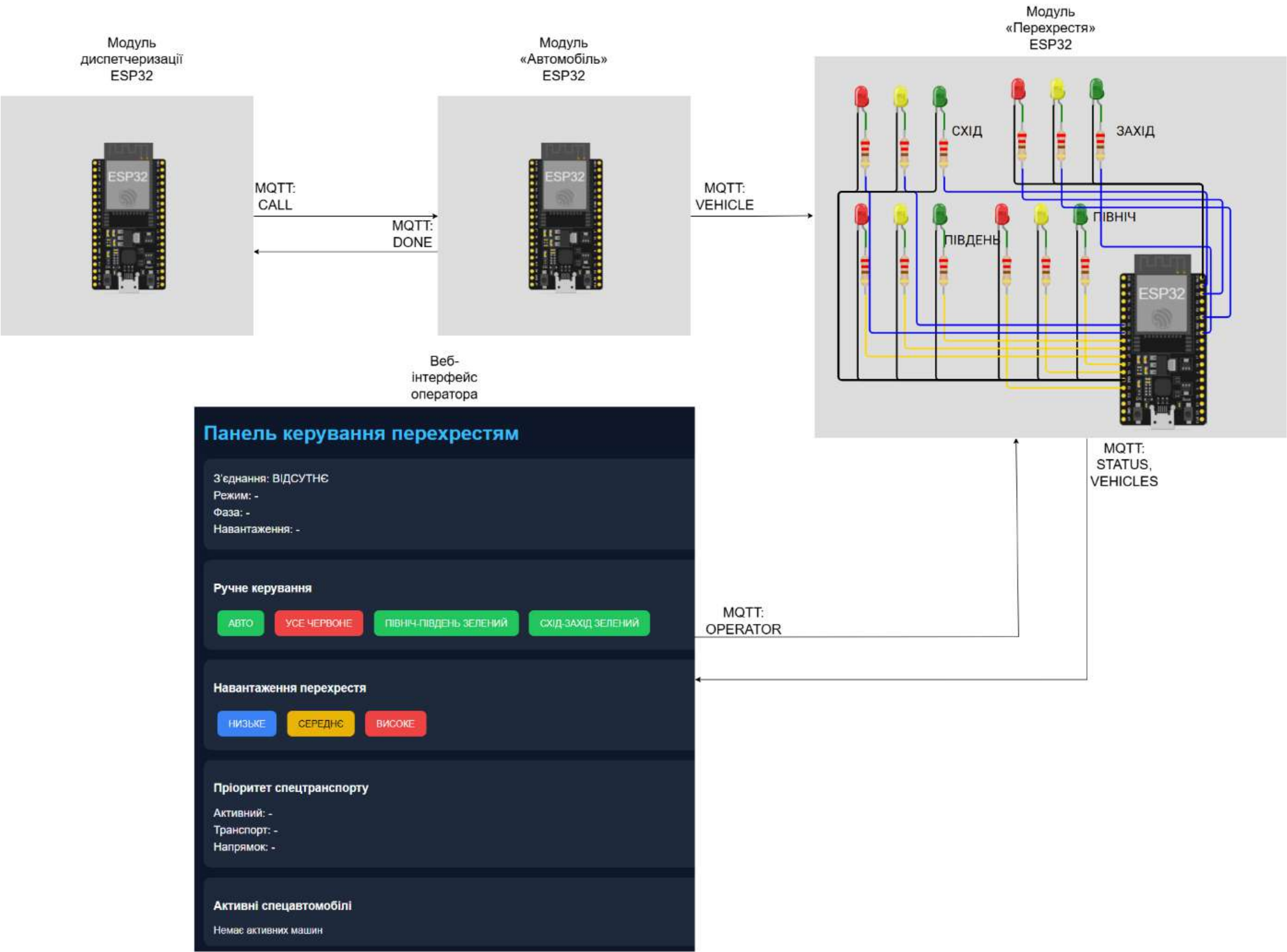


Рисунок 3.1 – Віртуальна схема кіберфізичної системи у середовищі Wokwi

Схему електричну принципову світлофорної індикації модуля «Перехрестя» наведено на рисунку 3.2.

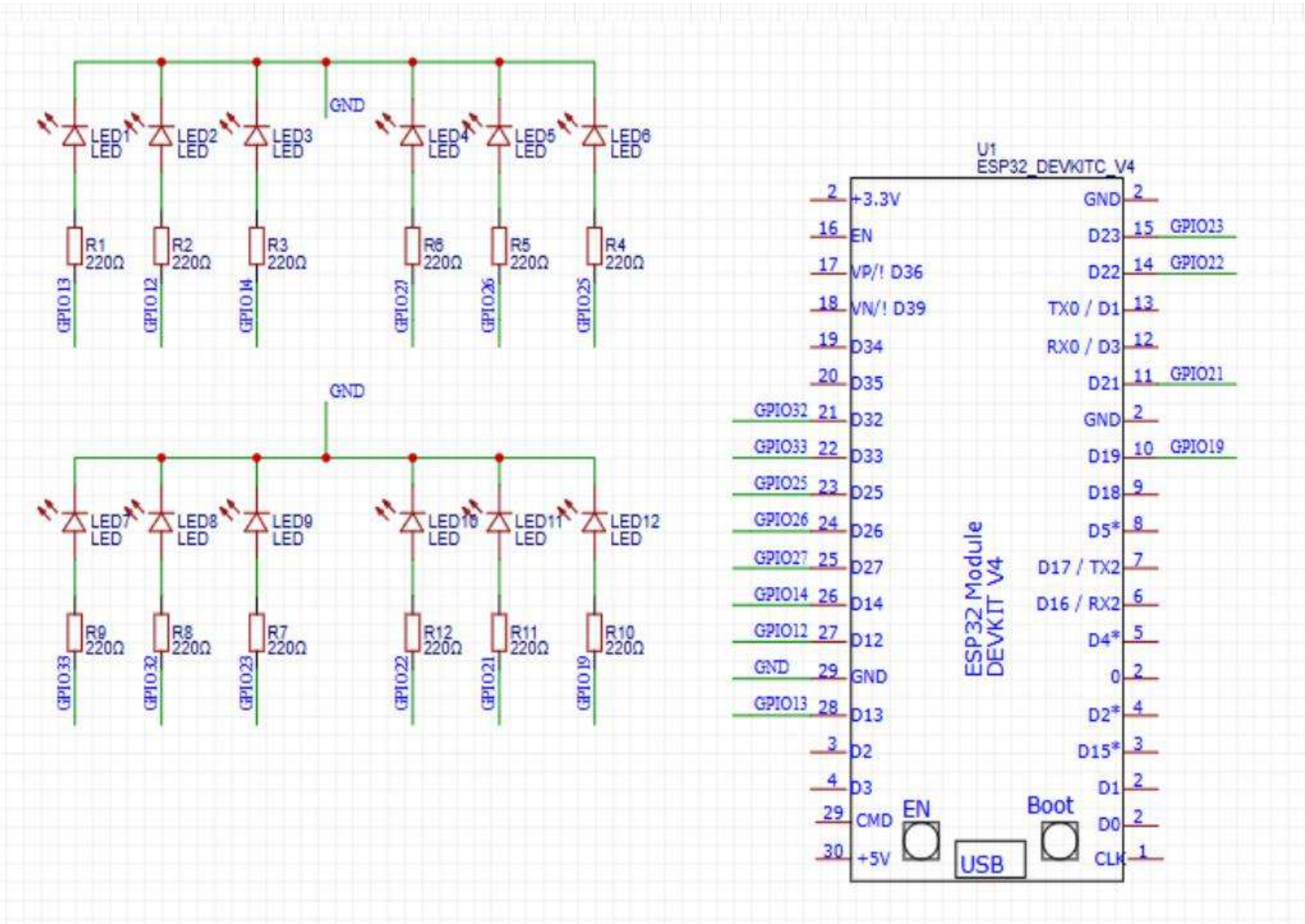


Рисунок 3.2 – Схема електрична принципова

Керування світлодіодами здійснюється через GPIO–виводи ESP32. Призначення виводів наведено в таблиці 3.1.

Таблиця 3.1 – Призначення GPIO–виводів модуля «Перехрестя»

Напрямок	Червоний	Жовтий	Зелений
Північний	GPIO 13	GPIO 12	GPIO 14
Південний	GPIO 27	GPIO 26	GPIO 25
Східний	GPIO 33	GPIO 32	GPIO 23
Західний	GPIO 22	GPIO 21	GPIO 19

Для зміни рівня транспортного навантаження у віртуальній схемі використовується кнопка, підключена до GPIO 18 модуля «Перехрестя».

Натискання кнопки циклічно перемикає режими LOW, MEDIUM та HIGH, що впливають на порогове значення відстані активації пріоритетного режиму.

Усі мікроконтролери підключаються до віртуальної бездротової мережі Wokwi–GUEST та використовують публічний MQTT–брокер для обміну повідомленнями. Такий підхід дозволяє реалізувати повністю розподілену модель системи, максимально наближену до реальних умов функціонування.

Перевагою використання Wokwi є можливість одночасного запуску кількох контролерів, перегляду журналів роботи кожного вузла, тестування різних сценаріїв функціонування та оперативного внесення змін до програмного забезпечення.

Таким чином, використання середовища Wokwi дало змогу реалізувати повноцінну віртуальну модель кіберфізичної системи пріоритетного пропуску спецтранспорту, що включає три взаємопов’язані модулі на базі ESP32, світлодіодну індикацію світлофорного об’єкта та мережеву взаємодію через MQTT–протокол. Побудована модель забезпечує можливість налагодження програмного забезпечення, дослідження взаємодії між вузлами системи та перевірки коректності роботи алгоритмів без використання реального апаратного обладнання. Отримана віртуальна схема стала основою для подальшої реалізації програмного забезпечення окремих модулів, опис яких наведено в наступних підрозділах третього розділу.

Наведена функція забезпечує автоматичне повторне підключення до брокера та підписку на топик повідомлень про завершення викликів.

3.2 Реалізація програмного забезпечення модуля диспетчеризації

Програмне забезпечення модуля диспетчеризації реалізує функції формування екстрених подій, вибору відповідного транспортного засобу та передавання інформації про виклик до модуля «Автомобіль». У структурі кіберфізичної системи цей модуль виконує роль диспетчерського центру, який

координує роботу спеціалізованого транспорту та забезпечує ініціацію процесу пріоритетного пропуску.

Програмна реалізація виконана мовою C++ з використанням Arduino Framework для мікроконтролера ESP32. Для забезпечення мережевої взаємодії застосовано стандартну бібліотеку WiFi.h, а для реалізації MQTT-обміну – бібліотеку PubSubClient. Після запуску модуль встановлює з’єднання з бездротовою мережею та підключається до MQTT-брокера, після чого переходить у режим очікування та генерації нових екстрених подій.

Основою програмної реалізації є структура Vehicle, яка описує характеристики транспортних засобів екстрених служб. Для кожного транспортного засобу зберігаються його ідентифікатор, тип служби, RFID-ідентифікатор, ознаки доступності та службові параметри, що визначають момент повернення транспортного засобу до списку вільних після завершення виклику. У моделі використовується один ESP32-модуль «Автомобіль», який програмно імітує роботу кількох логічних спецавтомобілів різних служб.

Для зберігання викликів, які не можуть бути оброблені негайно через відсутність вільного транспорту, використовується структура PendingEmergency. У межах імітаційної моделі черга викликів реалізована як програмна структура даних у пам’яті модуля диспетчеризації. Кожен елемент черги містить ознаку активності, ідентифікатор виклику, тип необхідної служби та час створення запису. Це дозволяє зберігати виклики та обслуговувати їх у порядку надходження після появи відповідного транспортного засобу.

Генерація екстрених подій виконується автоматично через випадкові проміжки часу. Для цього використовується вбудований генератор випадкових чисел ESP32, який визначає часову затримку між двома сусідніми подіями в межах від 5 до 12 секунд. Після настання запланованого моменту формується новий виклик, якому присвоюється унікальний ідентифікатор та випадковим чином визначається тип екстреної служби: швидка допомога, пожежна служба або поліція.

Після створення виклику програма виконує пошук доступного транспортного засобу відповідного типу. Якщо такий транспортний засіб знайдено, він позначається як зайнятий, а модуль формує MQTT-повідомлення та надсилає його до модуля «Автомобіль». Якщо ж усі транспортні засоби відповідного типу зайняті, виклик зберігається у внутрішній черзі очікування.

Для передавання інформації використовується MQTT-топик smartcity/dispatch/call. Повідомлення має текстовий формат:

CALL;emergencyId;vehicleId;serviceType;rfidTag

де emergencyId – унікальний ідентифікатор виклику, vehicleId – ідентифікатор призначеного транспортного засобу, serviceType – тип екстреної служби, rfidTag – ідентифікаційний параметр, що використовується для подальшої перевірки транспортного засобу модулем «Перехрестя».

Після завершення поїздки модуль «Автомобіль» передає повідомлення типу DONE через MQTT-топик smartcity/vehicle/done. Після отримання такого повідомлення транспортний засіб не стає доступним одразу, а переходить у стан очікування повернення. У програмній реалізації передбачено затримку тривалістю 10 секунд, яка моделює час повернення автомобіля до місця постійного базування.

Після завершення затримки транспортний засіб знову позначається як доступний. Програма автоматично перевіряє чергу необроблених викликів та, за наявності відповідного запиту, призначає повернений транспортний засіб на найстаріший виклик того самого типу служби. Такий механізм забезпечує безперервність функціонування системи та виключає втрату екстрених подій.

У процесі роботи модуль веде детальне журналювання через послідовний порт. У консоль виводяться повідомлення про створення екстрених ситуацій, вибір транспортних засобів, збереження викликів у черзі, повернення автомобілів та передавання MQTT-повідомлень. Це значно спрощує налагодження програмного забезпечення та аналіз роботи системи під час моделювання.

Для забезпечення обміну даними з іншими компонентами системи модуль диспетчеризації реалізує процедуру підключення до MQTT-брокера та підписки на топик повідомлень про завершення викликів. Відповідна функція наведена у лістингу 3.1.

```
void connectMQTT() {
    while (!mqttClient.connected()) {
        String clientId = "dispatch-module-" +
String(random(0xffff), HEX);

        if (mqttClient.connect(clientId.c_str())) {
            mqttClient.subscribe(VEHICLE_DONE_TOPIC);
        } else {
            delay(2000);
        }
    }
}
```

Лістинг 3.1 – Підключення до MQTT-брокера

Наведена функція забезпечує автоматичне повторне підключення до брокера у випадку втрати з’єднання та виконує підписку на топик smartcity/vehicle/done, через який надходять повідомлення про завершення роботи транспортних засобів.

Після вибору доступного транспортного засобу формується MQTT-повідомлення типу CALL, яке містить основні параметри виклику та передається до модуля «Автомобіль». Приклад формування такого повідомлення наведено у лістингу 3.2.

```
String payload = "CALL;";
payload += String(emergencyId);
payload += ";";
payload += vehicles[vehicleIndex].vehicleId;
```

```
payload += " ";
payload += vehicles[vehicleIndex].serviceType;
payload += " ";
payload += vehicles[vehicleIndex].rfidTag;
mqttClient.publish(CALL_TOPIC, payload.c_str());
```

Лістинг 3.2 – Формування MQTT-повідомлення про виклик

Сформоване повідомлення передається через топик smartcity/dispatch/call та використовується модулем «Автомобіль» для ініціалізації процесу моделювання руху спеціалізованого транспортного засобу.

У процесі функціонування система підтримує можливість одночасної обробки кількох екстрених подій, які можуть надходити у довільні моменти часу. У випадку виникнення кількох викликів різних типів диспетчерський модуль виконує незалежний підбір транспортних засобів для кожної служби, що дозволяє моделювати роботу реальної міської інфраструктури. Якщо ж у певний момент часу кількість викликів перевищує кількість доступного транспорту, система автоматично переходить до режиму накопичення необроблених заявок у внутрішній черзі. Після звільнення відповідного транспортного засобу виклики обробляються у порядку їх надходження, що забезпечує коректне функціонування алгоритму диспетчеризації.

Реалізований підхід дозволяє моделювати типові ситуації міського транспортного середовища, зокрема перевантаження системи екстрених служб, затримки повернення транспорту та одночасне обслуговування декількох маршрутів.

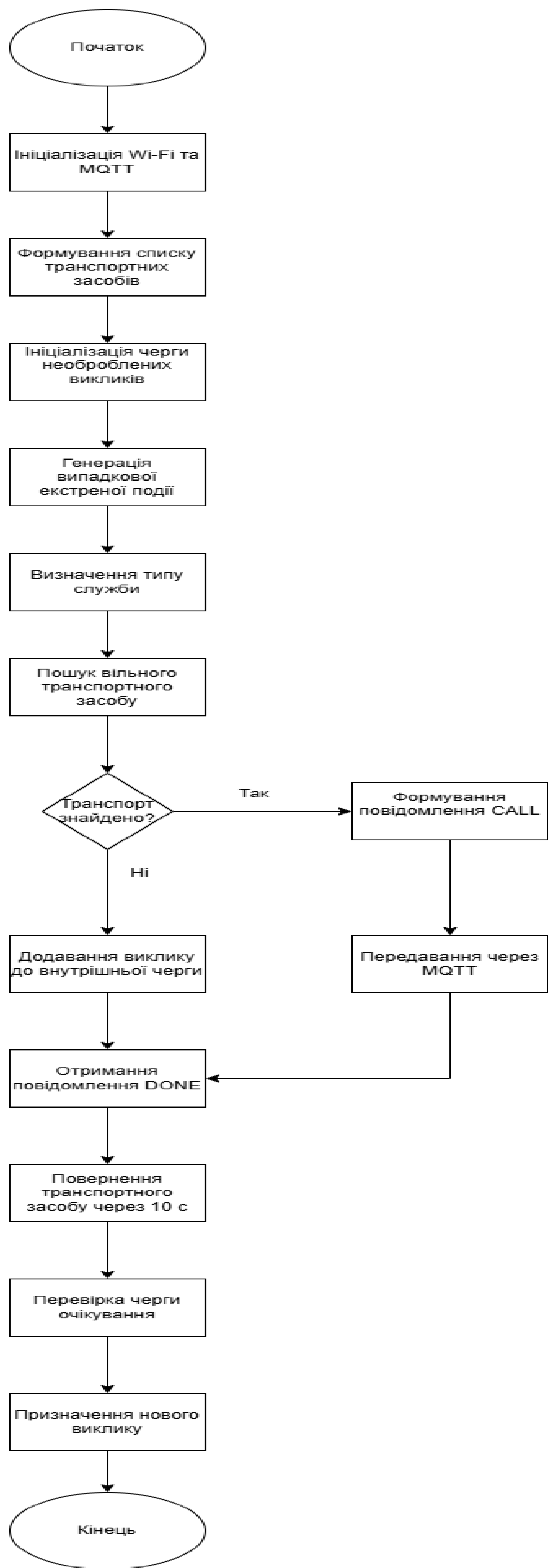


Рисунок 3.3 – Блок–схема роботи програмного забезпечення модуля диспетчеризації

3.3 Реалізація програмного забезпечення модуля «Автомобіль»

Програмне забезпечення модуля «Автомобіль» реалізує функції імітації роботи спеціалізованого транспортного засобу після отримання екстреного виклику від модуля диспетчеризації. У структурі кіберфізичної системи цей модуль виконує роль програмно–апаратного представлення автомобіля екстреної служби, який отримує завдання, моделює рух до перехрестя та передає інформацію про своє поточне положення.

Програмна реалізація виконана мовою C++ з використанням Arduino Framework для мікроконтролера ESP32. Для забезпечення бездротового підключення до локальної мережі використовується бібліотека WiFi.h, а для реалізації обміну повідомленнями через MQTT–протокол – бібліотека PubSubClient. Після запуску модуль встановлює з’єднання з мережею Wi-Fi, підключається до MQTT–брокера та підписується на топик smartcity/dispatch/call, через який надходять повідомлення про нові екстрені виклики.

Після отримання повідомлення типу CALL програма виконує його розбір та зчитує основні параметри: ідентифікатор виклику, ідентифікатор транспортного засобу, тип служби та RFID–ідентифікатор. Отримані дані зберігаються у внутрішніх змінних модуля та використовуються для подальшого моделювання руху.

Одразу після прийняття виклику модуль автоматично генерує початкові параметри руху. Напрямок підходу до перехрестя визначається випадковим чином із множини можливих значень: NORTH, SOUTH, EAST та WEST. Додатково випадково формується початкова відстань до перехрестя та кількість перехресть, після проходження яких виклик вважається завершеним. Це дозволяє моделювати різні сценарії руху транспортного засобу.

Під час активної поїздки модуль періодично зменшує значення відстані до перехрестя, імітуючи рух автомобіля. Передавання інформації про поточний стан виконується з періодом 1 секунда, що забезпечує регулярне оновлення

даних у системі та достатню деталізацію для тестування алгоритму керування світлофором.

Для передачі інформації використовується MQTT–топiк smartcity/vehicle/status. Повiдомлення має текстовий формат:

VEHICLE;emergencyId;vehicleId;serviceType;rfidTag;direction;distance

де emergencyId – iдентифікатор поточного виклику, vehicleId – iдентифікатор транспортного засобу, serviceType – тип екстреної служби, rfidTag – iдентифікаційний параметр, direction – напрямок руху, distance – поточна відстань до перехрестя.

Після досягнення нульового значення відстані вважається, що транспортний засіб пройшов перехрестя. Якщо кількість запланованих перехресть для даного виклику перевищує одиницю, процес моделювання повторюється для наступного перехрестя із новим значенням відстані. Після проходження останнього перехрестя виклик вважається завершеним.

Після завершення маршруту модуль формує MQTT–повідомлення типу DONE та передає його до модуля диспетчеризації через топiк smartcity/vehicle/done. Повiдомлення має формат:

DONE;vehicleId;serviceType;rfidTag

Отримання цього повідомлення диспетчерським модулем є підставою для переведення транспортного засобу у стан повернення та його подальшого включення до списку доступних автомобілів.

У процесі роботи модуль веде детальне журналювання через послідовний порт. У консоль виводяться повідомлення про отримання викликів, вибір напрямку руху, зміну відстані до перехрестя, проходження окремих перехресть та завершення поїздки. Це дозволяє аналізувати поведінку транспортного засобу та перевіряти правильність його взаємодії з іншими компонентами системи.

У процесі руху модуль «Автомобіль» періодично формує повідомлення про свій поточний стан, які містять iдентифікатор виклику, тип служби,

напрямок руху та відстань до перехрестя. Фрагмент програмного коду, що реалізує формування такого повідомлення, наведено у лістингу 3.3.

```
String payload = "VEHICLE;";
payload += String(currentEmergencyId);
payload += ";";
payload += vehicleId;
payload += ";";
payload += serviceType;
payload += ";";
payload += rfidTag;
payload += ";";
payload += direction;
payload += ";";
payload += String(distanceToIntersection);

mqttClient.publish(VEHICLE_STATUS_TOPIC, payload.c_str());
```

Лістинг 3.3 – Формування MQTT-повідомлення про стан спецавтомобіля

Передавання повідомлень здійснюється через топік smartcity/vehicle/status з періодом 1 секунда, що забезпечує регулярне оновлення інформації для модуля «Перехрестя» та веб-інтерфейсу.

Після завершення маршруту модуль «Автомобіль» формує службове повідомлення типу DONE, яке використовується для інформування модуля диспетчеризації про завершення обслуговування виклику. Відповідний фрагмент коду наведено у лістингу 3.4.

```
String payload = "DONE;";
payload += vehicleId;
payload += ";";
payload += serviceType;
payload += ";";
```

```
payload += rfidTag;

mqttClient.publish(VEHICLE_DONE_TOPIC, payload.c_str());
```

Лістинг 3.4 – Формування повідомлення про завершення виклику

Отримання цього повідомлення дозволяє модулю диспетчеризації перевести транспортний засіб у стан повернення та після завершення затримки знову включити його до списку доступних автомобілів.

У процесі моделювання транспортний засіб функціонує як автономний мережевий вузол, який незалежно виконує обробку викликів, генерацію параметрів руху та передавання службової інформації через MQTT–мережу. Використання ESP32 дозволяє одночасно підтримувати бездротове з’єднання, виконувати логіку моделювання руху та забезпечувати стабільний обмін повідомленнями без необхідності використання додаткових обчислювальних пристроїв. Завдяки цьому модуль може працювати як окремий компонент розподіленої кіберфізичної системи.

Додатково програмне забезпечення модуля «Автомобіль» виконує функцію безперервної синхронізації стану транспортного засобу з іншими компонентами кіберфізичної системи. Завдяки періодичному передаванню MQTT–повідомлень модуль «Перехрестя» отримує актуальну інформацію про наближення спецтранспорту та може своєчасно активувати режим пріоритетного пропуску. Одночасно ці повідомлення використовуються веб–інтерфейсом оператора для візуалізації поточного положення транспортного засобу, напрямку його руху та активних екстрених викликів у системі. Такий підхід забезпечує централізований моніторинг функціонування кіберфізичної системи в режимі реального часу.

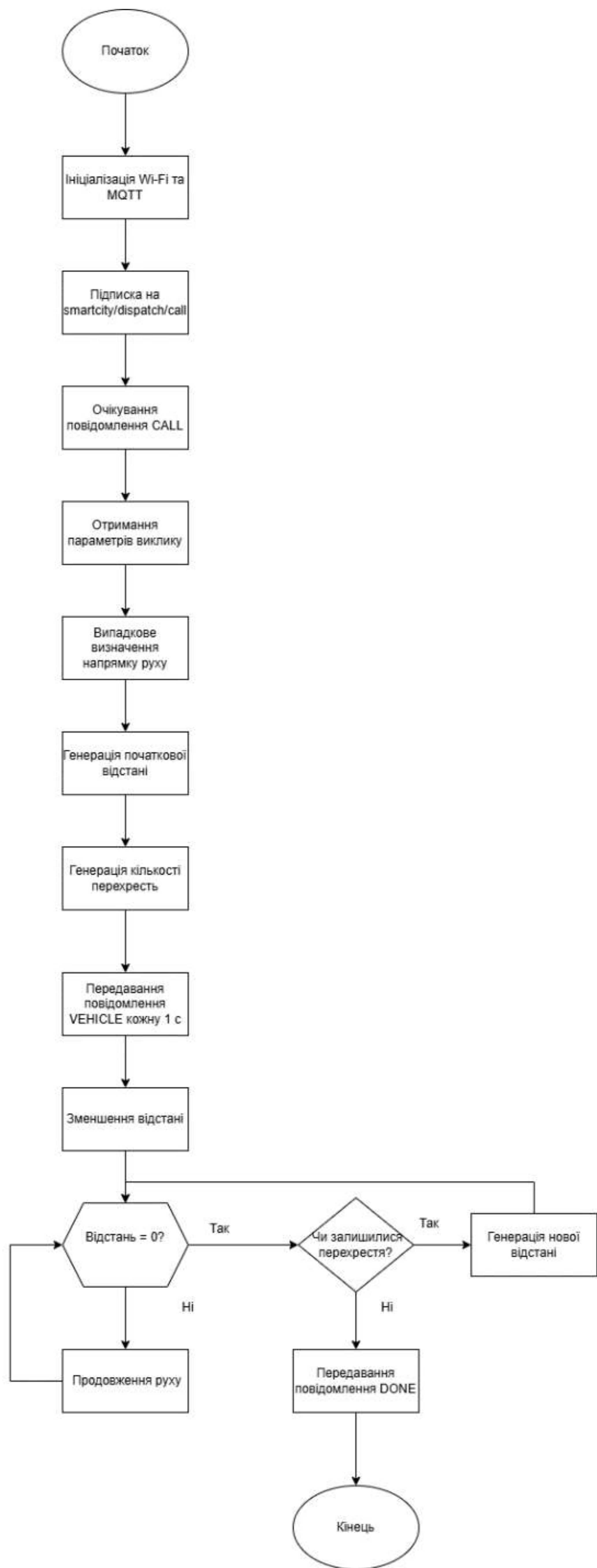


Рисунок 3.4 – Блок–схема роботи програмного забезпечення модуля «Автомобіль»

3.4 Реалізація програмного забезпечення модуля «Перехрестя»

Програмне забезпечення модуля «Перехрестя» реалізує основну логіку функціонування кіберфізичної системи пріоритетного пропуску спецтранспорту. Саме цей модуль виконує приймання даних про рух спеціалізованих транспортних засобів, перевірку їх допустимості, аналіз активних запитів, вибір режиму роботи системи та безпосереднє керування світлофорною індикацією. Таким чином, модуль «Перехрестя» є центральним програмно–апаратним компонентом, у якому зосереджено алгоритми прийняття рішень у режимі реального часу.

Програмна реалізація виконана мовою C++ з використанням Arduino Framework для мікроконтролера ESP32. Для забезпечення мережевої взаємодії застосовано бібліотеки WiFi.h та PubSubClient, а для керування світлодіодною індикацією використовуються стандартні функції роботи з GPIO–виводами. Після запуску модуль встановлює з’єднання з бездротовою мережею, підключається до MQTT–брокера та підписується на топіки smartcity/vehicle/status і smartcity/intersection/operator.

У структурі програмного забезпечення реалізовано базу допустимих RFID–ідентифікаторів, яка представлена масивом рядків. До цього масиву включено ідентифікатори всіх спеціалізованих транспортних засобів, визначених у системі. Після отримання повідомлення типу VEHICLE виконується перевірка параметра rfidTag. Якщо переданий ідентифікатор відсутній у базі, транспортний засіб ігнорується, а система не переходить до режиму пріоритетного керування.

Для зберігання інформації про транспортні засоби, що знаходяться у зоні обслуговування, використовується структура EmergencyRequest. У ній зберігаються ідентифікатор виклику, ідентифікатор транспортного засобу, тип служби, RFID–ідентифікатор, напрямок руху, поточна відстань та час останнього оновлення. Масив таких структур формує внутрішню чергу активних запитів,

яка розміщується у пам’яті ESP32 та постійно оновлюється відповідно до нових MQTT–повідомлень.

Однією з основних функцій програмного забезпечення є вибір транспортного засобу, для якого має бути сформовано «зелений коридор». Якщо у черзі одночасно присутні декілька транспортних засобів, програма виконує пошук найближчого до перехрестя автомобіля, відстань до якого не перевищує порогове значення, що залежить від поточного рівня навантаження.

У програмній реалізації передбачено чотири режими роботи системи: NORMAL, PRIORITY, MANUAL та SAFE. У режимі NORMAL виконується стандартне циклічне перемикання світлофорних фаз. Режим PRIORITY активується при наближенні допустимого транспортного засобу та забезпечує безперешкодний проїзд. Режим MANUAL використовується для ручного керування через веб–інтерфейс. У межах реалізованої моделі режим SAFE використовується як програмно передбачений безпечний стан, який може бути активований командою оператора або у випадку некоректної обробки пріоритетного запиту.

Керування світлофорними сигналами здійснюється через дванадцять GPIO–виводів, до яких підключені червоні, жовті та зелені світлодіоди для чотирьох напрямків руху. У програмному коді реалізовано окремі функції для встановлення основних станів: setAllRed(), setNorthSouthGreen(), setNorthSouthYellow(), setEastWestGreen() та setEastWestYellow().

При активації режиму PRIORITY програма спочатку переводить світлофор у безпечний стан «усе червоне», після чого вмикає зелений сигнал для напрямку руху пріоритетного транспортного засобу. Підтримка цього режиму триває до моменту, коли значення параметра distance зменшиться до нуля, що означає повне проходження транспортного засобу через перехрестя.

Для реалізації стандартного режиму роботи використовується послідовність фаз, що циклічно повторюється: зелений сигнал для напрямків північ–південь, жовтий сигнал, стан «усе червоне», зелений сигнал для

напряmkів схід–захід, жовтий сигнал та повторний стан «усе червоне». Перемикання фаз виконується на основі системного таймера millis() без використання блокуючих затримок.

Програмне забезпечення також реалізує обробку команд оператора, що надходять через MQTT–топiк smartcity/intersection/operator. Підтримуються команди переходу в автоматичний режим, ручного встановлення світлофорної фази та зміни рівня транспортного навантаження.

Для забезпечення візуального контролю та інтеграції з веб–інтерфейсом модуль періодично формує MQTT–повідомлення STATUS і VEHICLES, які містять інформацію про поточний режим роботи, активну фазу світлофора, рівень навантаження та список транспортних засобів, що перебувають у зоні обслуговування.

У процесі роботи програмне забезпечення веде детальне журналювання через послідовний порт. У консоль виводяться повідомлення про отримання MQTT–повідомлень, результати перевірки RFID, зміну режимів роботи, активацію пріоритетного пропуску та повернення системи до штатного режиму.

Одним із етапів обробки повідомлень є перевірка RFID–ідентифікатора транспортного засобу. Для цього використовується функція, що порівнює отримане значення з переліком дозволених ідентифікаторів. Реалізацію цієї функції наведено у лістингу 3.5.

```
bool isRfidAllowed(String rfidTag) {
    for (int i = 0; i < ALLOWED_RFID_COUNT; i++) {
        if (allowedRfidTags[i] == rfidTag) {
            return true;
        }
    }
    return false;
}
```

Лістинг 3.5 – Перевірка RFID–ідентифікатора

Одним із ключових етапів обробки повідомлень є перевірка RFID-ідентифікатора транспортного засобу. Для цього використовується функція, що порівнює отримане значення з переліком дозволених ідентифікаторів. Реалізацію цієї функції наведено у лістингу 3.5.

У разі одночасного надходження декількох запитів модуль «Перехрестя» визначає транспортний засіб, що знаходиться найближче до перехрестя. Відповідний фрагмент програмного коду наведено у лістингу 3.6.

```
if (requests[i].distance < closestDistance) {
    closestDistance = requests[i].distance;
    selectedIndex = i;
}
```

Лістинг 3.6 – Вибір найближчого транспортного засобу

Наведений код реалізує вибір запиту з мінімальним значенням параметра distance, що забезпечує першочерговий пропуск найближчого транспортного засобу.

Після визначення пріоритетного напрямку виконується перемикання світлофорної індикації. Вибір необхідної фази реалізовано функцією, наведеною у лістингу 3.7.

```
void setPriorityGreen(String direction) {
    if (direction == "NORTH" || direction == "SOUTH") {
        setNorthSouthGreen();
    } else {
        setEastWestGreen();
    }
}
```

Лістинг 3.7 – Перемикання світлофора для пріоритетного напрямку

Залежно від напрямку руху транспортного засобу функція активує зелений сигнал для напрямків «північ–південь» або «схід–захід», що забезпечує формування «зеленого коридору» для безперешкодного проїзду спецтранспорту.

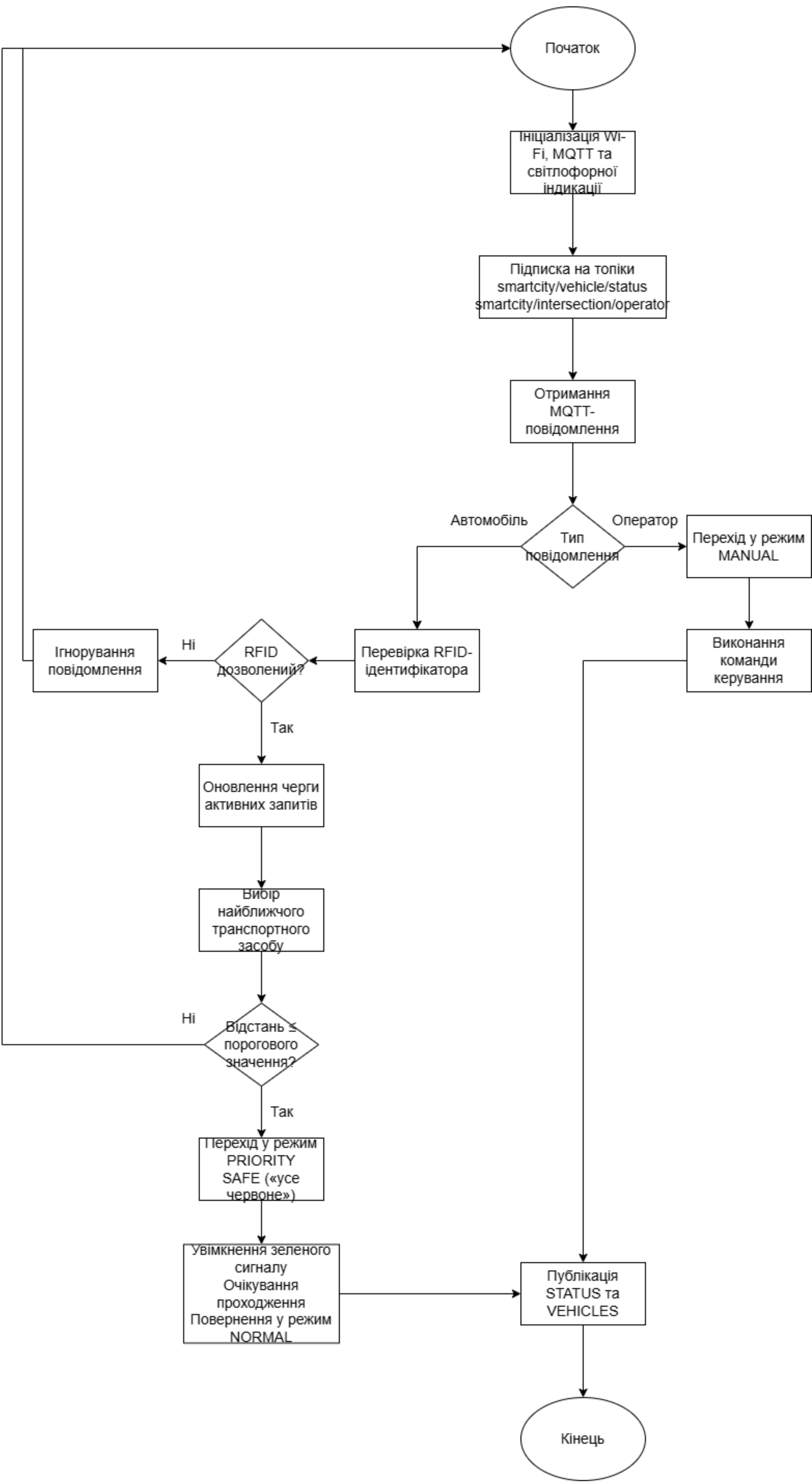


Рисунок 3.5 – Блок–схема роботи програмного забезпечення модуля «Перехрестя»

3.5 Реалізація інформаційної взаємодії та MQTT–обміну

Реалізація інформаційної взаємодії між модулями кіберфізичної системи виконана на основі протоколу MQTT, який забезпечує асинхронний обмін повідомленнями між розподіленими компонентами через брокерну модель «публікація–підписка». Такий підхід дозволяє організувати взаємодію між модулями без необхідності встановлення прямих з’єднань між ними та забезпечує високу масштабованість системи.

У програмному забезпеченні всіх ESP32–модулів використано бібліотеку PubSubClient, яка реалізує функції підключення до MQTT–брокера, підписки на топіки, публікації повідомлень та обробки вхідних даних. Для мережевого доступу застосовується бібліотека WiFi.h, що забезпечує підключення контролерів до бездротової мережі Wokwi–GUEST.

Як MQTT–брокер у процесі моделювання використовується публічний сервер HiveMQ (broker.hivemq.com) з портом 1883 для ESP32–модулів та WebSocket–портом 8884 для веб–інтерфейсу. Така конфігурація дозволяє одночасно підключати як мікроконтролери, так і браузерний клієнт без додаткового серверного програмного забезпечення.

У процесі моделювання використання публічного MQTT–брокера є цілком прийнятним, оскільки дозволяє швидко організувати інформаційну взаємодію між модулями без необхідності розгортання додаткової серверної інфраструктури. Разом із тим, такий підхід орієнтований насамперед на дослідницькі та навчальні цілі. Для реального впровадження системи в умовах транспортної інфраструктури доцільно використовувати приватний MQTT–брокер із налаштованою автентифікацією користувачів, механізмами розмежування доступу та захищеним каналом зв’язку. Це дозволить підвищити інформаційну безпеку системи, запобігти несанкціонованому доступу до керувальних команд та забезпечити надійність функціонування в умовах експлуатації.

Після запуску кожен модуль виконує встановлення Wi-Fi-з'єднання, після чого формує унікальний ідентифікатор MQTT-клієнта та підключається до брокера. У разі розриву з'єднання програмне забезпечення автоматично виконує повторне підключення, що забезпечує безперервність функціонування системи.

Набір MQTT-топиків, що використовуються для взаємодії між модулями системи, а також їх функціональне призначення були визначені на етапі проєктування та наведені в таблиці 2.5. У процесі програмної реалізації всі модулі використовують зазначені топіки без змін, що забезпечує повну відповідність між проєктною моделлю та програмною реалізацією системи.

Усі повідомлення передаються у текстовому форматі, в якому параметри розділяються символом «;». Такий підхід не потребує використання додаткових бібліотек серіалізації, спрощує розбір повідомлень та зменшує обсяг передаваних даних, що є важливим для мікроконтролерних систем.

Формати повідомлень, які передаються через MQTT-протокол, наведені в таблиці 2.6. Усі програмні модулі реалізують обробку саме цих форматів, що забезпечує сумісність та коректну інтерпретацію передаваних даних.

Для обробки вхідних повідомлень у кожному модулі реалізовано callback-функцію, яка приймає назву топіка та вміст повідомлення, виконує його розбір та передає отримані дані до відповідних функціональних блоків. Залежно від типу повідомлення викликаються процедури генерації викликів, моделювання руху, керування світлофорами або оновлення веб-інтерфейсу.

Передавання повідомлень у системі здійснюється асинхронно. Кожен модуль самостійно публікує дані у відповідний топик, а інші модулі отримують їх лише за наявності активної підписки. Така організація дозволяє розділити функціональність системи на незалежні компоненти та спростити подальше розширення.

У процесі тестування MQTT-обмін забезпечив стабільне передавання повідомлень між усіма вузлами системи, включаючи одночасну роботу декількох транспортних засобів, веб-інтерфейсу та алгоритмів локального

керування. Це підтвердило придатність обраного протоколу для побудови кіберфізичних систем транспортного призначення.

3.6 Реалізація веб–інтерфейсу моніторингу стану перехрестя

Веб–інтерфейс моніторингу стану перехрестя реалізовано як окремий клієнтський застосунок, призначений для візуалізації параметрів роботи кіберфізичної системи та забезпечення можливості ручного керування. На відміну від мікроконтролерних модулів, веб–інтерфейс виконується у середовищі веб–браузера та взаємодіє з іншими компонентами системи через MQTT–протокол із використанням WebSocket–з’єднання.

Програмна реалізація виконана з використанням HTML, CSS та JavaScript без застосування додаткових серверних технологій. Такий підхід дозволяє запускати інтерфейс як звичайну HTML–сторінку, що значно спрощує розгортання та тестування системи.

Для організації обміну даними використовується JavaScript–бібліотека mqtt.min.js, яка забезпечує підключення до MQTT–брокера HiveMQ через захищений WebSocket–порт 8884. Після встановлення з’єднання веб–інтерфейс підписується на топіки smartcity/intersection/status та smartcity/intersection/vehicles, а також має можливість публікувати команди в топик smartcity/intersection/operator.

Набір MQTT–топиків і формати повідомлень, що використовуються веб–інтерфейсом, відповідають структурі, визначеній на етапі проєктування, та наведені в таблицях 2.5 і 2.6.

Структура веб–інтерфейсу включає кілька функціональних областей. У верхній частині сторінки розташовано блок загального стану системи, у якому відображаються статус підключення до MQTT–брокера, поточний режим роботи, активна фаза світлофора та рівень транспортного навантаження. Окремий блок призначений для відображення інформації про активний

пріоритетний транспортний засіб, включаючи його ідентифікатор та напрямок руху.

Для ручного керування передбачено панель оператора з кнопками переходу в автоматичний режим, переведення світлофора у стан «усе червоне», встановлення зеленого сигналу для напрямків північ–південь або схід–захід, а також вибору рівня транспортного навантаження. Натискання кнопок призводить до формування та передавання відповідних MQTT–команд до модуля «Перехрестя».

Для візуалізації активних транспортних засобів реалізовано динамічний список інформаційних блоків. Для кожного транспортного засобу відображаються його ідентифікатор, тип служби, напрямок руху, поточна відстань до перехрестя та стан обробки. Додатково використовується графічна шкала прогресу та спрощена схема дороги, на якій положення автомобіля змінюється відповідно до отриманого значення відстані.

Оновлення інформації виконується в реальному часі при надходженні MQTT–повідомлень. JavaScript–код аналізує отримані дані, оновлює значення на сторінці та створює або видаляє інформаційні блоки транспортних засобів залежно від поточного стану системи.

Структуру екрана веб–інтерфейсу наведено в таблиці 3.2.

Таблиця 3.2 – Структура екрана веб–інтерфейсу.

Елемент веб–інтерфейсу	Призначення
Блок стану системи	відображення режиму роботи, фази світлофора та рівня навантаження
Панель ручного керування	надсилання команд оператором
Блок пріоритетного транспорту	відображення активного транспортного засобу
Список активних транспортних засобів	відображення інформації про всі транспортні засоби

Кінець таблиці 3.2

Шкала прогресу та схема дороги	візуалізація наближення транспортного засобу до перехрестя
Індикатор підключення	відображення стану з’єднання з MQTT-брокером

Зовнішній вигляд веб-інтерфейсу наведено на рисунку 3.6.

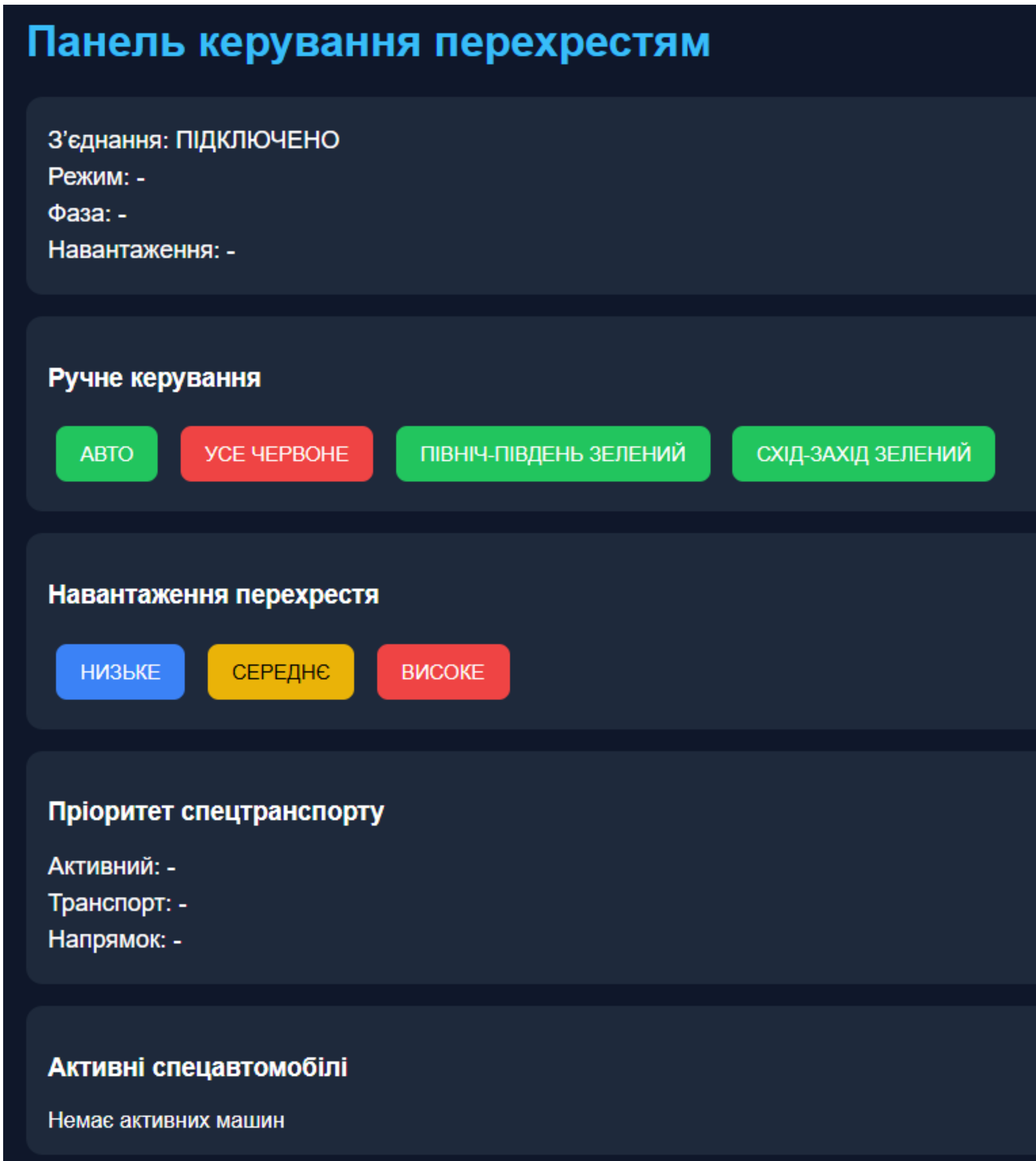


Рисунок 3.6 – Веб-інтерфейс.

На рисунку 3.6 показано веб-інтерфейс оператора, який містить блок стану підключення, поточний режим роботи системи, панель ручного керування, рівень навантаження перехрестя та список активних спецавтомобілів.

У процесі тестування веб-інтерфейс забезпечив коректне відображення стану системи, стабільне оновлення даних у режимі реального часу та успішне передавання команд керування до модуля «Перехрестя».

3.7 Тестування роботи системи та аналіз результатів моделювання

Після завершення програмної реалізації всіх модулів було проведено комплексне тестування кіберфізичної системи пріоритетного пропуску спецтранспорту у середовищі Wokwi. Метою тестування була перевірка коректності взаємодії між модулями, функціонування алгоритму адаптивного керування світлофорними фазами, стабільності MQTT-обміну та працездатності веб-інтерфейсу моніторингу.

Тестування виконувалося шляхом послідовного відтворення типових сценаріїв роботи системи. Для кожного сценарію аналізувалися повідомлення послідовного порту, стан світлодіодної індикації, дані веб-інтерфейсу та результати MQTT-обміну.

Першим етапом було перевірено функціонування системи у штатному режимі NORMAL. За відсутності активних транспортних засобів модуль «Перехрестя» здійснював стандартне циклічне перемикання світлофорних фаз. У веб-інтерфейсі відображалися поточний режим роботи, активна фаза світлофора та встановлений рівень транспортного навантаження.

На другому етапі тестувалася обробка одного екстреного виклику. Після генерації події модуль диспетчеризації обирав доступний транспортний засіб та надсилав повідомлення типу CALL. Модуль «Автомобіль» приймав виклик, генерував напрямок руху та починав передавання повідомлень типу VEHICLE. Після досягнення порогової відстані модуль «Перехрестя» переходив у режим PRIORITY, формував «зелений коридор» та підтримував його до повного проходження транспортного засобу через перехрестя.

Окремо перевірявся сценарій одночасного наближення декількох транспортних засобів. У цьому випадку модуль «Перехрестя» коректно формував внутрішню чергу активних запитів та визначав транспортний засіб з мінімальною відстанню до перехрестя. Після завершення обслуговування першого транспортного засобу система автоматично переходила до обробки наступного запиту.

Наступний сценарій стосувався ситуації відсутності доступних транспортних засобів певного типу. У такому випадку модуль диспетчеризації зберігав виклик у внутрішній черзі очікування. Після завершення роботи одного з транспортних засобів та завершення затримки повернення виклик автоматично призначався на вільний автомобіль. Це підтвердило коректність роботи механізму відкладеного призначення.

Також було протестовано механізм перевірки RFID-ідентифікаторів. У разі надходження повідомлення з невідомим ідентифікатором модуль «Перехрестя» не активував пріоритетний режим та залишав світлофор у поточному стані. Це підтвердило працездатність механізму автентифікації транспортних засобів.

Окремо перевірялася робота ручного режиму MANUAL. Через веб-інтерфейс оператор надсилав команди зміни світлофорних фаз, переходу в автоматичний режим та встановлення рівня транспортного навантаження. Усі команди успішно приймалися модулем «Перехрестя» та призводили до відповідної зміни стану системи.

Під час тестування веб-інтерфейсу було підтверджено коректне відображення параметрів роботи системи в реальному часі. Усі активні транспортні засоби відображалися окремими інформаційними блоками із зазначенням типу служби, напрямку руху, відстані до перехрестя та стану обробки. Положення транспортних засобів на графічній схемі змінювалося відповідно до переданих даних.

Результати тестування показали, що реалізована система забезпечує коректну взаємодію між усіма компонентами, стабільний MQTT-обмін, правильне формування «зеленого коридору» та автоматичне повернення до штатного режиму після проходження спецтранспорту.

Серед основних обмежень реалізованої моделі слід відзначити використання програмної імітації RFID-ідентифікації, відсутність реальних датчиків транспортного потоку та застосування одного віртуального

транспортного засобу для моделювання роботи різних спеціалізованих служб. Крім того, моделювання проводилося в умовах віртуального середовища Wokwi, що не враховує всі особливості реального функціонування транспортної інфраструктури.

Отримані результати підтвердили працездатність запропонованої архітектури та ефективність розробленого алгоритму адаптивного керування світлофорними фазами. Проведене моделювання показало можливість використання системи як основи для подальшого створення реального прототипу кіберфізичної системи пріоритетного пропуску спецтранспорт.

Для оцінювання працездатності розробленої кіберфізичної системи було проведено серію функціональних випробувань, що охоплювали як штатні режими роботи, так і ситуації з одночасною обробкою кількох екстрених викликів, відсутністю вільних транспортних засобів, використанням ручного керування та перевіркою механізмів захисту. Для кожного сценарію визначався очікуваний результат, після чого він порівнювався з фактичними результатами, отриманими під час моделювання у середовищі Wokwi та тестування веб–інтерфейсу. Узагальнені результати проведених випробувань наведено в таблиці 3.3.

Таблиця 3.3 – Результати тестування основних сценаріїв роботи системи

№	Сценарій тестування	Очікуваний результат	Фактичний результат
1	Робота системи у штатному режимі NORMAL	Світлофор циклічно перемикає фази відповідно до заданих часових інтервалів	Циклічне перемикання фаз виконується коректно
2	Надходження одного екстреного виклику	Диспетчеризація призначає вільний транспортний засіб	Виклик успішно сформовано та передано до модуля «Автомобіль»

Продовження таблиці 3.3				
3	Наближення одного спеціалізованого транспортного засобу	Модуль «Перехрестя» активує режим PRIORITY та формує «зелений коридор»	Пріоритетний режим активується, зелений сигнал вмикається для відповідного напрямку	
4	Проходження транспортного засобу через перехрестя (distance = 0)	Система завершує режим PRIORITY та повертається до режиму NORMAL	Повернення до штатного циклу виконується автоматично	
5	Одночасне наближення двох і більше транспортних засобів	Обирається транспортний засіб з мінімальною відстанню до перехрестя	Пріоритет надається найближчому транспортному засобу	
6	Відсутність вільного транспортного засобу потрібного типу	Виклик зберігається у внутрішній черзі диспетчеризації	Виклик успішно збережено та призначено після повернення автомобіля	
7	Отримання повідомлення з невідомим RFID-ідентифікатором	Запит ігнорується, пріоритетний режим не активується	Некоректний транспортний засіб не впливає на роботу системи	
8	Надсилання команди ALL_RED через веб-інтерфейс	Система переходить у безпечний режим SAFE	Усі напрямки переводяться у стан «усе червоне»	
9	Надсилання команди NS_GREEN або EW_GREEN	Активується відповідна світлофорна фаза в режимі MANUAL	Ручне керування виконується коректно	
				Арк. 72
Зм.	Арк.	№ докум.	Підпис	

Кінець таблиці 3.3

10	Надсилання команд LOAD_LOW, LOAD_MEDIUM, LOAD_HIGH	Змінюється порогова відстань активації пріоритетного режиму	Рівень навантаження змінюється та враховується алгоритмом
11	Відображення активних транспортних засобів у веб–інтерфейсі	На сторінці створюються окремі інформаційні блоки для кожного транспортного засобу	Веб–інтерфейс коректно оновлює список і візуалізацію руху
12	Втрата MQTT– з’єднання	Модулі автоматично виконують повторне підключення до брокера	З’єднання відновлюється без перезапуску системи

Результати перевірки основних функціональних сценаріїв роботи кіберфізичної системи наведено в таблиці 3.3.

Для узагальнення параметрів реалізованої кіберфізичної системи доцільно навести основні кількісні характеристики, які визначають структуру моделі, часові параметри роботи алгоритмів та обмеження внутрішніх структур даних. Ці параметри безпосередньо використовуються у програмній реалізації та були застосовані під час моделювання і тестування системи. Основні кількісні характеристики розробленої системи наведено в таблиці 3.4.

Таблиця 3.4 – Основні кількісні характеристики реалізованої системи

Показник	Значення
Кількість ESP32–модулів	3
Кількість напрямків регульованого перехрестя	4

Кінець таблиці 3.4

Кількість світлодіодів світлофорної індикації	12
Кількість екстрених служб	3 (швидка допомога, пожежна служба, поліція)
Кількість транспортних засобів у моделі	6
Кількість транспортних засобів кожного типу	2
Максимальна кількість викликів у черзі диспетчеризації	10
Максимальна кількість активних запитів у модулі «Перехрестя»	5
Період передавання повідомлень VEHICLE	1 с
Період публікації повідомлень STATUS	1 с
Період публікації повідомлень VEHICLES	1 с
Інтервал генерації екстрених викликів	5–12 с
Затримка повернення транспортного засобу до списку доступних	10 с
Час зеленого сигналу у штатному режимі	5 с
Час жовтого сигналу	2 с
Час проміжного стану «усе червоне»	1 с
Максимальна тривалість режиму PRIORITY	20 с
Порогова відстань активації пріоритету	25 м / 45 м / 70 м залежно від рівня навантаження

Незважаючи на успішне проходження всіх функціональних випробувань, розроблена модель має низку обмежень, пов’язаних з використанням імітаційного середовища та спрощених механізмів взаємодії. По–перше, у

процесі моделювання не враховуються реальні затримки та можливі втрати пакетів у бездротовому каналі зв'язку, що в реальних умовах можуть впливати на час реакції системи. По–друге, RFID–ідентифікація реалізована програмно шляхом передавання відповідного параметра у складі MQTT–повідомлення, без використання фізичних RFID–зчитувачів та міток. По–третє, у системі не використовується GPS, GNSS або інші засоби реального позиціонування транспортних засобів; відстань до перехрестя моделюється програмно та змінюється за заданим алгоритмом. По–четверте, для організації інформаційної взаємодії використовується публічний MQTT–брокер, що є зручним для дослідницького моделювання, але не забезпечує рівень надійності та безпеки, необхідний для промислового впровадження. По–п'яте, у реалізованій системі відсутня криптографічна автентифікація повідомлень, а перевірка транспортних засобів базується лише на порівнянні ідентифікаторів RFID з внутрішньою базою дозволених значень. По–шосте, модель не враховує поведінку звичайного транспортного потоку, накопичення черг автомобілів та зміну інтенсивності руху в реальному часі. По–сьоме, світлофорний об'єкт представлено світлодіодною індикацією, що демонструє логіку роботи системи, але не відображає особливостей взаємодії з промисловими контролерами дорожньої інфраструктури. Незважаючи на зазначені обмеження, створена модель повністю виконує поставлені функціональні завдання та підтверджує працездатність запропонованої архітектури, алгоритмів керування та механізмів інформаційної взаємодії.

3.8 Висновки до третього розділу

У третьому розділі виконано програмну реалізацію, моделювання та тестування кіберфізичної системи пріоритетного пропуску спецтранспорту на основі мікроконтролерів ESP32. На відміну від другого розділу, де було розроблено архітектуру та визначено принципи функціонування системи, у

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						75
Зм.	Арк.	№ докум.	Підпис	Дата		

цьому розділі реалізовано всі основні програмні модулі та перевірено їхню працездатність у середовищі Wokwi.

Створено віртуальну схему системи, яка включає модуль диспетчеризації, модуль «Автомобіль», модуль «Перехрестя», світлодіодну індикацію світлофорного об’єкта та веб–інтерфейс оператора. Реалізована модель забезпечила можливість відтворення роботи системи без використання фізичного обладнання та стала ефективним засобом налагодження програмного забезпечення.

Розроблено програмне забезпечення для всіх основних модулів системи. Модуль диспетчеризації реалізує генерацію екстрених подій, вибір доступних транспортних засобів та ведення внутрішньої черги викликів. Модуль «Автомобіль» забезпечує моделювання руху спецтранспорту та періодичне передавання даних про його стан. Модуль «Перехрестя» виконує перевірку RFID–ідентифікаторів, аналіз активних запитів, формування «зеленого коридору», керування світлофорними фазами та передавання інформації до веб–інтерфейсу.

Реалізовано підсистему інформаційної взаємодії на основі MQTT–протоколу, яка забезпечує асинхронний обмін повідомленнями між усіма вузлами системи. Розроблено веб–інтерфейс моніторингу, що дозволяє відображати стан перехрестя у режимі реального часу та здійснювати ручне керування.

Проведене тестування підтвердило коректність роботи всіх функціональних компонентів, включаючи обробку одиночних та множинних екстрених викликів, роботу внутрішніх черг, механізм перевірки RFID, формування пріоритетного режиму та повернення системи до штатного циклу роботи.

Отримані результати моделювання засвідчили працездатність запропонованої архітектури та ефективність реалізованих алгоритмів. Розроблена система може бути використана як основа для подальшого створення

фізичного прототипу та інтеграції з реальними засобами ідентифікації транспортних засобів, датчиками дорожнього руху та інфраструктурними компонентами інтелектуальних транспортних систем

Практичним результатом третього розділу є працездатна імітаційна модель, яка об’єднує ESP32–модулі, MQTT–обмін, світлодіодну модель світлофора та веб–інтерфейс оператора.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						77
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У кваліфікаційній роботі розв’язано актуальне прикладне завдання комп’ютерної інженерії, яке полягає у розробленні кіберфізичної системи пріоритетного пропуску спеціалізованого транспорту на регульованих перехрестях на основі мікроконтролерної платформи ESP32 та бездротових комунікаційних технологій. Актуальність дослідження зумовлена необхідністю скорочення часу прибуття екстрених служб до місця події, зменшенням затримок у міському транспортному середовищі та підвищенням рівня автоматизації сучасних інтелектуальних транспортних систем.

У першому розділі проведено аналіз предметної області, розглянуто принципи побудови кіберфізичних систем, особливості інтелектуальних транспортних систем та сучасні підходи до організації пріоритетного пропуску спецтранспорту. Проаналізовано існуючі технології бездротової взаємодії, засоби ідентифікації транспортних засобів та методи адаптивного керування світлофорними об’єктами. На основі виконаного аналізу сформульовано функціональні та технічні вимоги до розроблюваної системи та обґрунтовано вибір апаратно–програмних засобів.

У другому розділі виконано проєктування кіберфізичної системи та визначено її загальну архітектуру. Обґрунтовано використання гібридної розподіленої структури, у якій модуль диспетчеризації, модуль «Автомобіль», модуль «Перехрестя», MQTT–брокер та веб–інтерфейс оператора взаємодіють у межах єдиної інформаційної інфраструктури. Розроблено структуру окремих підсистем, алгоритм адаптивного керування світлофорними фазами, механізми інформаційної взаємодії та структуру веб–інтерфейсу моніторингу.

У третьому розділі здійснено програмну реалізацію всіх основних компонентів системи на базі ESP32 з використанням Arduino Framework та протоколу MQTT. У середовищі Wokwi створено імітаційну модель системи, що включає три взаємодіючі мікроконтролерні модулі, світлодіодну індикацію роботи

світлофорного об’єкта та веб–інтерфейс оператора. Реалізовано програмне забезпечення модуля диспетчеризації, який генерує екстрені події та керує чергою викликів; модуля «Автомобіль», що моделює рух спеціалізованого транспортного засобу; та модуля «Перехрестя», який виконує перевірку RFID–ідентифікаторів, аналізує активні запити та формує «зелений коридор». Також розроблено веб–інтерфейс, що забезпечує моніторинг стану системи в режимі реального часу та можливість ручного керування.

У результаті проведеного тестування підтверджено коректність функціонування системи в основних сценаріях роботи: обробка одиночних та множинних екстрених викликів, робота внутрішньої черги при відсутності вільних транспортних засобів, перевірка RFID–ідентифікаторів, автоматичне формування пріоритетного режиму, повернення до штатного циклу роботи та ручне керування через веб–інтерфейс. Проведене моделювання засвідчило стабільність MQTT–обміну та узгоджену взаємодію всіх програмних компонентів системи.

Практичне значення роботи полягає у створенні працездатної імітаційної моделі кіберфізичної системи на базі ESP32, яка може бути використана як основа для подальшого створення фізичного програмно–апаратного прототипу. Запропонована архітектура допускає масштабування шляхом підключення реальних RFID–зчитувачів, датчиків транспортного потоку, GPS–модулів, промислових світлофорних контролерів та захищеної серверної інфраструктури.

Перспективами подальших досліджень є впровадження фізичних модулів RFID–ідентифікації, інтеграція з реальними світлофорними контролерами, використання алгоритмів прогнозування транспортних потоків та побудова мережі взаємодіючих перехресть у масштабах міської транспортної інфраструктури.

Отже, поставлену мету кваліфікаційної роботи досягнуто, а всі сформульовані завдання виконано. Розроблена кіберфізична система підтвердила коректність закладених алгоритмів і можливість використання як основи для подальшого створення фізичного прототипу системи пріоритетного пропуску спеціалізованого транспорту.

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						79
Зм.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Rakha H. A., Ahn K., Du J., Farag M. Quantifying the impact of Cellular Vehicle-to-Everything (C-V2X) on transportation system efficiency, energy and environment. *Proceedings of the 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020. P. 1–7.

2. Araghi S., Khosravi A., Creighton D. Intelligent cuckoo search optimized traffic signal controllers for multi-intersection networks. *Expert Systems with Applications*. 2015. Vol. 42, No. 9. P. 4422–4431.

3. Garg D., Chli M., Vogiatzis G. Deep reinforcement learning for autonomous traffic light control. *Proceedings of the 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*. IEEE, 2018. P. 214–218. DOI: 10.1109/ICITE.2018.8492537.

4. Isradi M. et al. Traffic performance analysis of unsignalized intersection using the traffic conflict parameter technique. *Sinergi*. 2022. Vol. 26, No. 3. P. 397–406. DOI: 10.22441sinergi.2022.3.015.

5. Dani A. et al. RFID based automatic pathway for emergency vehicles. *Proceedings of the IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE, 2025. P. 1–5. DOI: 10.1109/SCEECS64059.2025.10940127.

6. Guerrero-Ibáñez J., Zeadally S., Contreras-Castillo J. Sensor technologies for intelligent transportation systems. *Sensors*. 2018. Vol. 18, No. 4. Art. 1212. DOI: 10.3390s18041212.

7. Dong C. et al. The hybrid systems method integrating STAMP and HFACS for causal analysis of road traffic accidents. *Ergonomics*. 2024. Vol. 67, No. 7. P. 971–994. DOI: 10.108000140139.2023.2270783.

8. El-afifi M. I., Sakr H. A. Intelligent traffic management systems: *A review Nile Journal of Communication and Computer Science*. 2023. P. 1–16. DOI: 10.21608njccs.2023.211812.1012.

9. Hamadeh N., Karouni A., Farhat Z. Intelligent transportation systems to mitigate road traffic congestion. *Intelligenza Artificiale*. 2022. Vol. 15, No. 2. P. 91–104. DOI: 10.3233IA–200079.
10. Degadwala N. D. S., Baria N. R. Differential analysis of emergency vehicle detection in urban traffic: A systematic review. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*. 2024. Vol. 10, No. 5. P. 47–56. DOI: 10.32628CSEIT2410585.
11. Romanowska A., Budzyński M. Investigating the impact of weather conditions and time of day on traffic flow characteristics. *Weather, Climate, and Society*. 2022. Vol. 14, No. 3. P. 823–833. DOI: 10.1175WCAS–D–22–0012.1.
12. Yunus S., Abdulkarim I. A. Road traffic crashes and emergency response optimization: A geo–spatial analysis. *Geomatics, Natural Hazards and Risk*. 2022. Vol. 13, No. 1. P. 1535–1555. DOI: 10.108019475705.2022.2086829.
13. Hao Z., Wang Y., Yang X. Every second counts: A comprehensive review of route optimization and priority control for urban emergency vehicles. *Sustainability*. 2024. Vol. 16, No. 7. Art. 2917. DOI: 10.3390su16072917.
14. Damaševičius R., Bacanin N., Misra S. From sensors to safety: Internet of Emergency Services (IoES). *Journal of Sensor and Actuator Networks*. 2023. Vol. 12, No. 3. Art. 41. DOI: 10.3390jsan12030041.
15. Chauhan S. et al. Adaptive traffic control system with emergency vehicle detection. *Proceedings of ICCSAI*. IEEE, 2025. P. 2048–2052. DOI: 10.1109ICCSAI64074.2025.11064659.
16. Tashtoush Y. et al. Dynamic traffic light system to reduce waiting time of emergency vehicles. *International Journal of Computers Communications & Control*. 2022. Vol. 17, No. 3. DOI: 10.15837ijccc.2022.3.4482.
17. Roy S. et al. Development of emergency vehicle preemption strategies on smart corridors. *ACM SIGSPATIAL*. 2024. P. 12–21. DOI: 10.11453681772. 3698213.
18. Long K. J. et al. A collaborative control framework for emergency vehicle priority. *Journal of Advanced Transportation*. 2024. Art. 9691419.

DOI: 10.115520249691419.

19. Hossain A. et al. Ambulance crash risk dynamics. *Transportmetrica A: Transport Science*. 2025. P. 1–39. DOI: 10.108023249935.2025.2481578.

20. Duarte E. K. et al. SafeSmart: A VANET–LTE–based solution. *IEEE CSCN*. 2023. P. 47–53. DOI: 10.1109CSCN60443.2023.10453175.

21. Ismail S. F. Real–time SDN–IoT integrated framework for emergency vehicle prioritization. *Mesopotamian Journal of Computer Science*. 2025. P. 186–201. DOI: 10.58496MJCSC2025011.

22. Schintler L. A. Intelligent Transportation Systems (ITS). *Encyclopedia of Big Data*. Springer, 2022. P. 577–580. DOI: 10.1007978–3–319–32010–6_378.

23. Luo G. et al. ESTNet: Embedded spatial–temporal network for modeling traffic flow dynamics. *IEEE T–ITS*. 2022. Vol. 23. P. 19201–19212. DOI: 10.1109TIV.2024.3367919.

24. Cordoş N. et al. Overview of ITS in Europe. *World Electric Vehicle Journal*. 2025. DOI: 10.3390wevj16070387.

25. Artificial intelligence for future intelligent transportation. CRC Press, 2024. 567 p.

26. Correia D. et al. Smart cities in EU. *Smart Cities*. 2022. DOI: 10.3390smartcities5040089.

27. Tariq M. U. Cyber–physical systems 2.0. CRC Press, 2024. 320p

28. Kociuba D. et al. Smart city ecosystem model. *Energies*. 2023. DOI: 10.3390en16062795.

29. Almulhim A., Yigitcanlar T. Smart governance. *Smart Cities*. 2025. DOI: 10.3390smartcities8040113.

30. Majstorović Ž. et al. Traffic signal control review. *Applied Sciences*. 2023. DOI: 10.3390app13074484.

31. José R. I., Rodrigues H. Smart city innovation challenges. *Smart Cities*. 2024. DOI: 10.3390smartcities7010006.

32. Haghani M. *Emergency vehicle prioritization in connected traffic*. 2024.
DOI: 10.18297etd4322.

33. Zhang K. et al. Industrial cyber–physical systems. *IEEE T–II*. 2022.
DOI: 10.1109TII.2022.3199481.

34. Hercog D. et al. ESP32 IoT devices. *Sensors*. 2023.
DOI: 10.3390s23156739.

35. Zulfiqar A. Hands–on ESP32 with Arduino IDE. Packt, 2024. 280p

36. Want R. RFID Explained. Springer Nature, 2022. 145p

37. Casella G. et al. RFID evolution. *Procedia CS*. 2022.
DOI: 10.1016j.procs.2022.01.359.

38. Shymchyshyn O. et al. ESP–NOW monitoring system. 2025.
DOI: 10.23939ictee2025.02.162.

39. Li C., Cao Z. LoRa networking survey. *ACM CSUR*. 2022.
DOI: 10.11453494673.

40. Wokwi. Arduino & ESP32 Simulation Platform Documentation [Електронний ресурс]. 2024. URL: <https://wokwi.com> (дата звернення 25.04.2026)

41. Song Y., Zlatkovic M., Porter R. J. Evaluation of GPS–based transit signal priority for mixed–traffic bus rapid transit. *Transportation Research Record*. 2016. Vol. 2539, No. 1. P. 30–39. DOI: 10.31412539–04.

42. Maha Vishnu V. C., Rajalakshmi M., Nedunchezhian R. Intelligent traffic video surveillance and accident detection system with dynamic traffic signal control. *Cluster Computing*. 2018. Vol. 21, No. 1. P. 135–147.
DOI: 10.1007s10586–017–0974–5.

43. Jagadeesh V., Reddy T. L., Sundari G. Traffic control system for emergency vehicles using RFID and sensors. *Proceedings of the 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2023. P. 1739–1744.
DOI: 10.1109ICICCS56967.2023.10142622.

44. Rabby M. K. M., Islam M. M., Imon S. M. A review of IoT application in a smart traffic management system. *Proceedings of the 5th International Conference on*

Advances in Electrical Engineering (ICAEE). IEEE, 2019. P. 280–285.
DOI: 10.1109/ICAEE48663.2019.8975582.

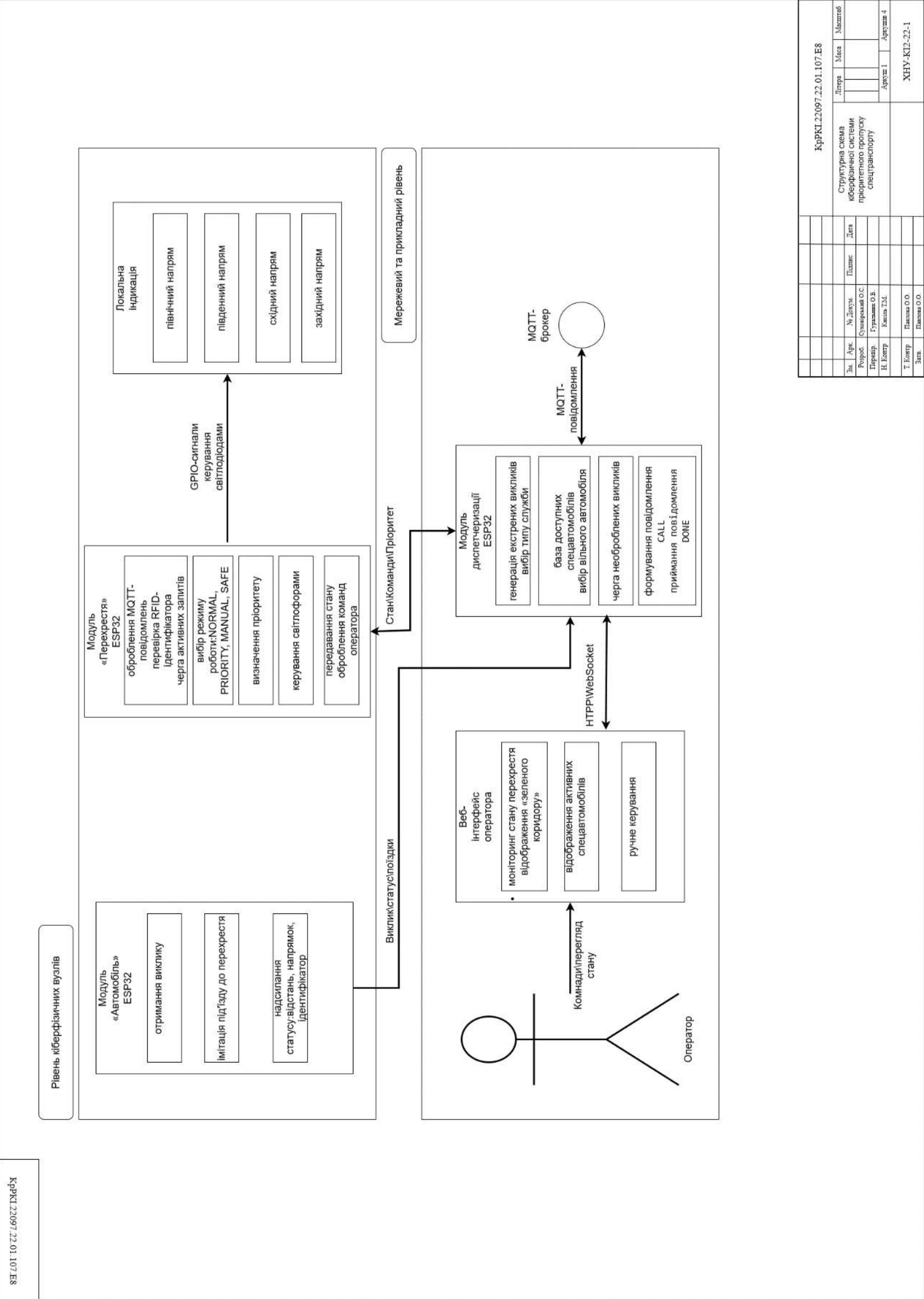
45. Mannoni V., Berg V., Sesia S., Perraud E. A comparison of the V2X communication systems: ITS–G5 and C–V2X. *Proceedings of the IEEE 89th Vehicular Technology Conference (VTC2019–Spring)*. IEEE, 2019. P. 1–5.
DOI: 10.1109/VTCSpring.2019.8746562.

46. Batyrgareieva V., Kolodyazhny M., Netesa N. War as a challenge to road safety: damage to society and the economy of Ukraine. *Baltic Journal of Economic Studies*. 2023. Vol. 9, No. 5. P. 48–56.
DOI: <https://doi.org/10.30525/2256-0742/2023-9-5-48>

					КвРКІ.022097.22.01.107 ПЗ	Арк.
						84
Зм.	Арк.	№ докум.	Підпис	Дата		

(обов'язковий)

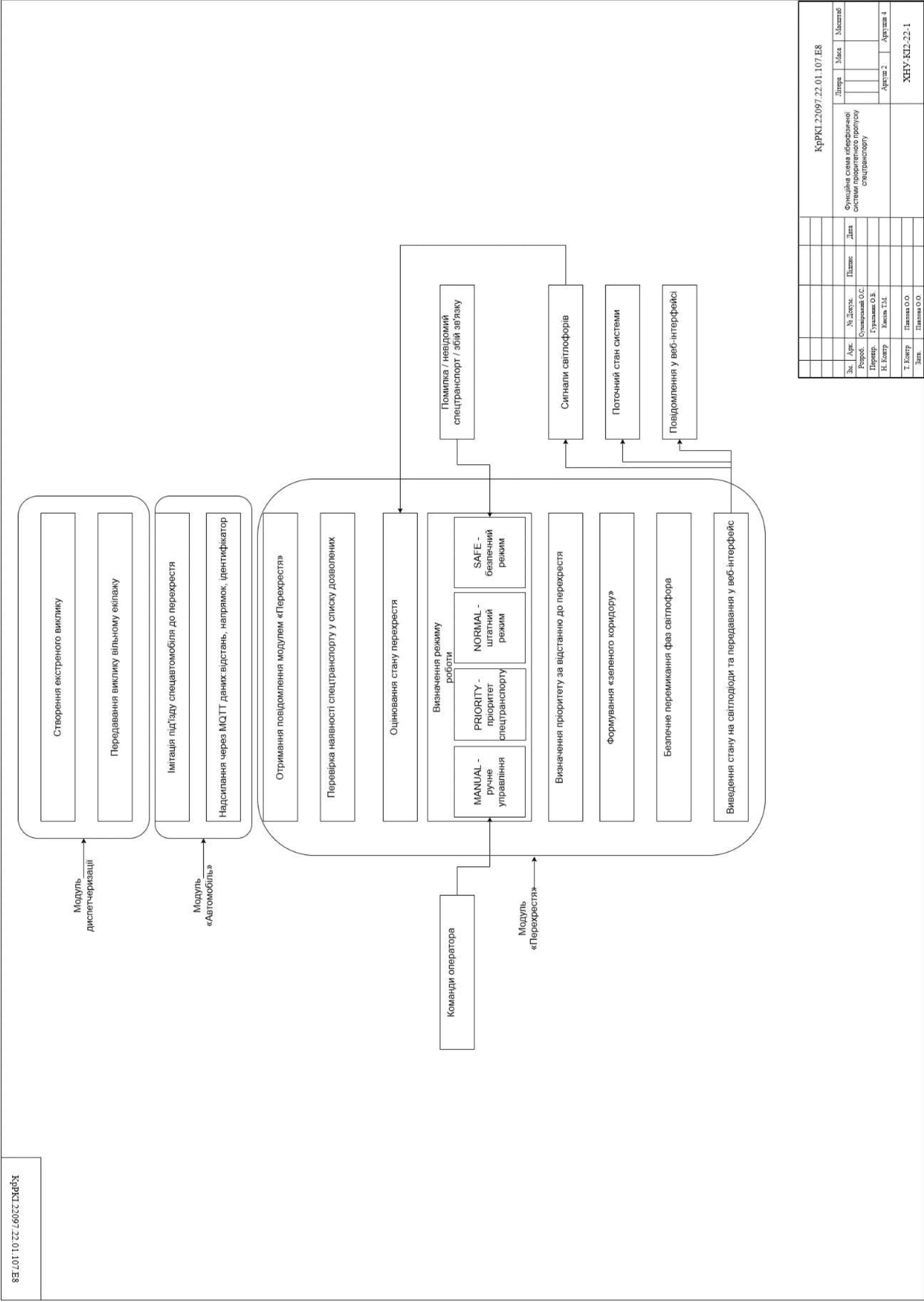
Копія креслення “Структурна схема кібефізичної системи



ДОДАТОК Б

(обов’язковий)

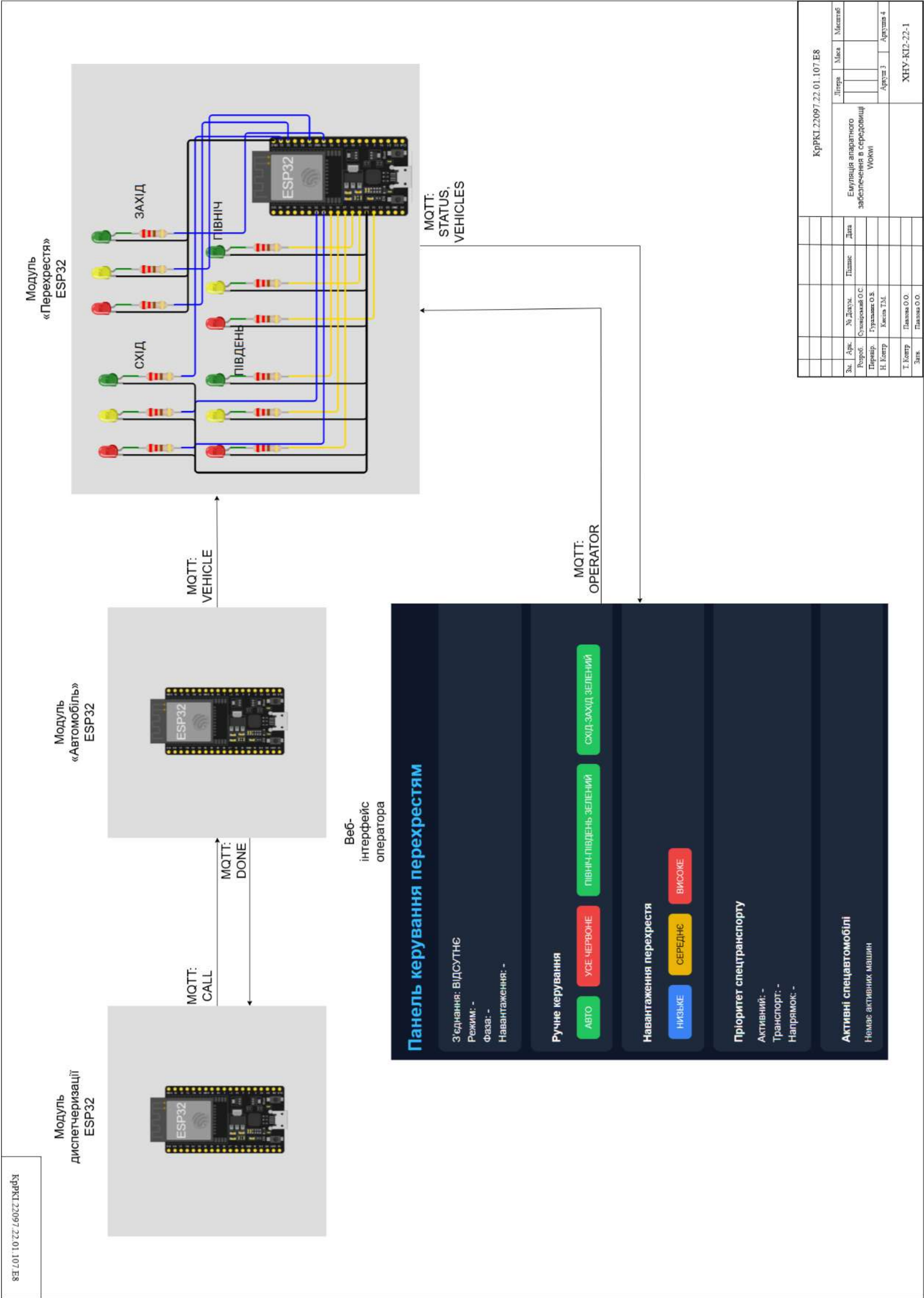
Копія креслення “Функційна схема кібефізичної системи”



ДОДАТОК В

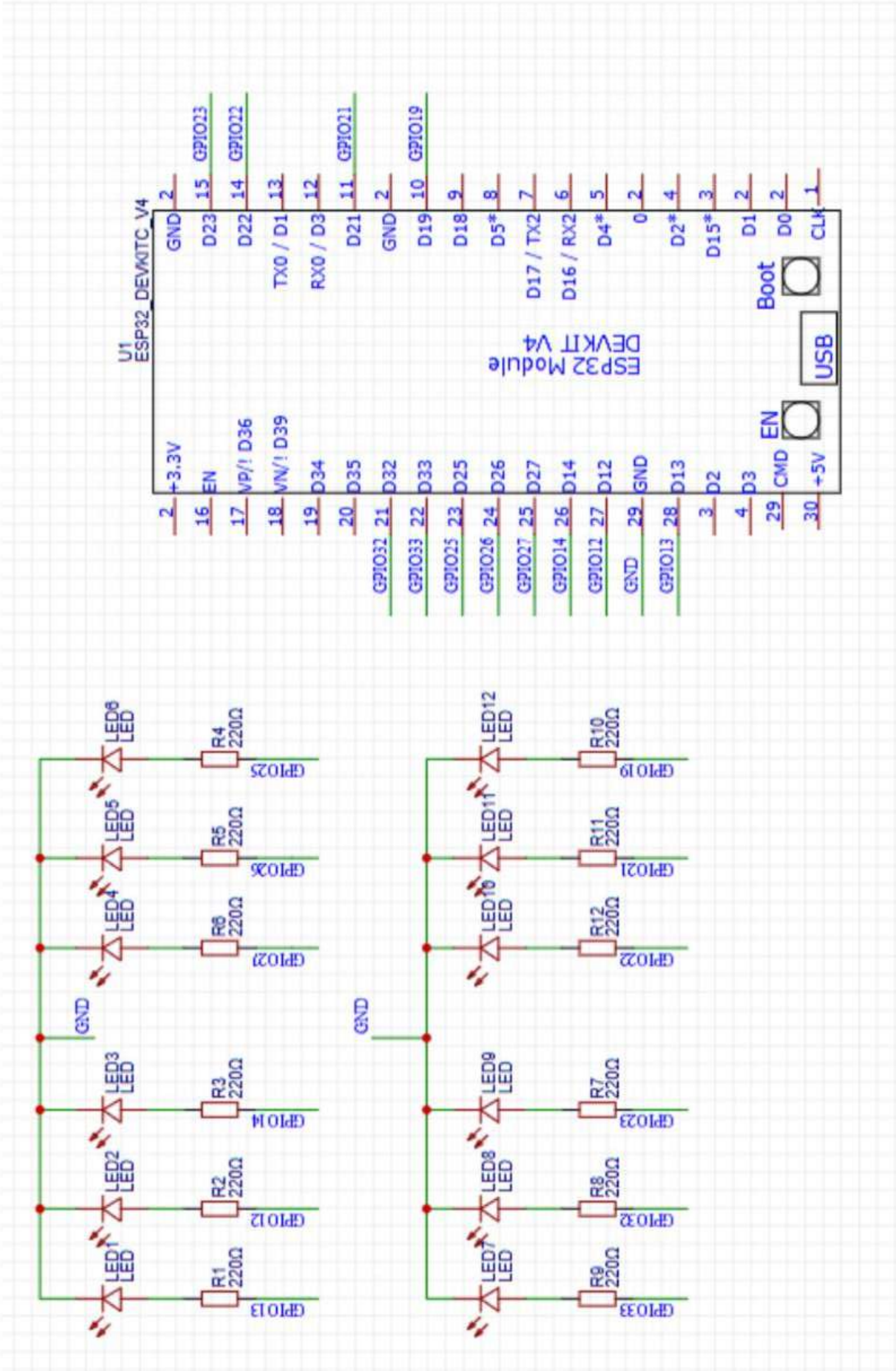
(обов’язковий)

Копія креслення “Емуляція програмного забезпечення в середовищі Wokwi



(обов'язковий)

Копія креслення “Принципова схема кібефізичної системи”

[illegible]

ДОДАТОК Д

(обов'язковий)

Код програмного забезпечення

```
#include <WiFi.h>
#include <PubSubClient.h>
const char* WIFI_SSID = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";
const char* MQTT_SERVER = "broker.hivemq.com";
const int MQTT_PORT = 1883;
const char* CALL_TOPIC = "smartcitydispatchcall";
const char* VEHICLE_DONE_TOPIC = "smartcityvehicledone";
WiFiClient espClient;
PubSubClient mqttClient(espClient);
struct Vehicle {
    String vehicleId;
    String serviceType;
    String rfidTag;
    bool available;
    bool waitingReturn;
    unsigned long returnAvailableAt;
};
Vehicle vehicles[] = {
    {"AMB-001", "AMBULANCE", "RFID-AMB-001", true, false, 0},
    {"AMB-002", "AMBULANCE", "RFID-AMB-002", true, false, 0},
    {"FIRE-001", "FIRE", "RFID-FIRE-001", true, false, 0},
    {"FIRE-002", "FIRE", "RFID-FIRE-002", true, false, 0},
    {"POL-001", "POLICE", "RFID-POL-001", true, false, 0},
    {"POL-002", "POLICE", "RFID-POL-002", true, false, 0}
};
const int VEHICLE_COUNT = sizeof(vehicles) / sizeof(vehicles[0]);
struct PendingEmergency {
    bool active;
    int emergencyId;
    String serviceType;
    unsigned long createdAt;
};
const int MAX_PENDING = 10;
PendingEmergency pendingQueue[MAX_PENDING];
unsigned long nextEmergencyTime = 0;
const unsigned long MIN_RANDOM_DELAY = 5000;
const unsigned long MAX_RANDOM_DELAY = 12000;
const unsigned long VEHICLE_RETURN_DELAY = 10000;
int emergencyCounter = 0;
void setup() {
    Serial.begin(115200);
    randomSeed(esp_random());
    clearPendingQueue();
    connectWiFi();
    mqttClient.setServer(MQTT_SERVER, MQTT_PORT);
    mqttClient.setCallback(onMqttMessage);
    connectMQTT();
    Serial.println("=== Dispatch Module Started ===");
    scheduleNextEmergency();
}
void loop() {
    if (!mqttClient.connected()) {
        connectMQTT();
    }
    mqttClient.loop();
    updateVehicleAvailability();
}
```

```

        handleRandomEmergency();
    }
    void connectWiFi() {
        Serial.print("[WIFI] Connecting");
        WiFi.begin(WIFI_SSID, WIFI_PASSWORD, 6);
        while (WiFi.status() != WL_CONNECTED) {
            delay(250);
            Serial.print(".");
        }
        Serial.println();
        Serial.println("[WIFI] Connected");
    }
    void connectMQTT() {
        while (!mqttClient.connected()) {
            String clientId = "dispatch-module-" + String(random(0xffff), HEX);
            Serial.print("[MQTT] Connecting... ");
            if (mqttClient.connect(clientId.c_str())) {
                Serial.println("connected");
                mqttClient.subscribe(VEHICLE_DONE_TOPIC);
                Serial.print("[MQTT] Subscribed to: ");
                Serial.println(VEHICLE_DONE_TOPIC);
            } else {
                Serial.print("failed, rc=");
                Serial.println(mqttClient.state());
                delay(2000);
            }
        }
    }
    void onMqttMessage(char* topic, byte* payload, unsigned int length) {
        String message = "";
        for (unsigned int i = 0; i < length; i++) {
            message += (char)payload[i];
        }
        Serial.println();
        Serial.print("[MQTT] Message received: ");
        Serial.println(message);
        if (message.startsWith("DONE;")) {
            handleVehicleDone(message);
        }
    }
    void handleVehicleDone(String message) {
        String parts[4];
        if (!splitMessage(message, parts, 4)) {
            Serial.println("[ERROR] Invalid DONE message format");
            return;
        }
        String vehicleId = parts[1];
        int vehicleIndex = findVehicleById(vehicleId);
        if (vehicleIndex == -1) {
            Serial.println("[ERROR] DONE from unknown vehicle");
            return;
        }
        vehicles[vehicleIndex].waitingReturn = true;
        vehicles[vehicleIndex].returnAvailableAt = millis() +
VEHICLE_RETURN_DELAY;
        Serial.print("[DISPATCH] Vehicle finished trip: ");
        Serial.println(vehicleId);
        Serial.println("[DISPATCH] Vehicle will become available in 10 sec");
    }
    void scheduleNextEmergency() {
        unsigned long delayTime = random(MIN_RANDOM_DELAY, MAX_RANDOM_DELAY +
1);

        nextEmergencyTime = millis() + delayTime;
        Serial.println();
    }

```

```

        Serial.print("[SYSTEM] Next emergency in ");
        Serial.print(delayTime 1000);
        Serial.println(" sec");
    }
    void handleRandomEmergency() {
        if (millis() < nextEmergencyTime) return;
        emergencyCounter++;
        String requiredService = getRandomServiceType();
        Serial.println();
        Serial.println("[EVENT] Emergency generated");
        Serial.print("[EVENT] Emergency ID: ");
        Serial.println(emergencyCounter);
        Serial.print("[EVENT] Required service: ");
        Serial.println(requiredService);
        processEmergency(emergencyCounter, requiredService);
        scheduleNextEmergency();
    }
    void processEmergency(int emergencyId, String serviceType) {
        int vehicleIndex = findAvailableVehicle(serviceType);
        if (vehicleIndex == -1) {
            Serial.println("[DISPATCH] No available vehicle. Emergency added to
pending queue.");
            addPendingEmergency(emergencyId, serviceType);
            return;
        }
        dispatchVehicle(vehicleIndex, emergencyId);
    }
    String getRandomServiceType() {
        int value = random(0, 3);
        if (value == 0) return "AMBULANCE";
        if (value == 1) return "FIRE";
        return "POLICE";
    }
    void dispatchVehicle(int vehicleIndex, int emergencyId) {
        vehicles[vehicleIndex].available = false;
        vehicles[vehicleIndex].waitingReturn = false;
        String payload = "CALL;";
        payload += String(emergencyId);
        payload += ";";
        payload += vehicles[vehicleIndex].vehicleId;
        payload += ";";
        payload += vehicles[vehicleIndex].serviceType;
        payload += ";";
        payload += vehicles[vehicleIndex].rfidTag;
        mqttClient.publish(CALL_TOPIC, payload.c_str());
        Serial.println("[DISPATCH] Vehicle selected and dispatched");
        Serial.print("[DATA] Vehicle ID: ");
        Serial.println(vehicles[vehicleIndex].vehicleId);
        Serial.print("[DATA] Service: ");
        Serial.println(vehicles[vehicleIndex].serviceType);
        Serial.print("[DATA] RFID: ");
        Serial.println(vehicles[vehicleIndex].rfidTag);
        Serial.print("[MQTT] Published: ");
        Serial.println(payload);
    }
    void updateVehicleAvailability() {
        for (int i = 0; i < VEHICLE_COUNT; i++) {
            if (vehicles[i].waitingReturn == false && millis() >=
vehicles[i].returnAvailableAt) {
                vehicles[i].waitingReturn = false;
                vehicles[i].available = true;
                Serial.println();
                Serial.print("[DISPATCH] Vehicle returned to available list: ");
                Serial.println(vehicles[i].vehicleId);
            }
        }
    }

```

```

        assignPendingEmergencyIfPossible(i);
    }
}

void assignPendingEmergencyIfPossible(int vehicleIndex) {
    int pendingIndex = findOldestPendingByService(vehicles[vehicleIndex].serviceType);
    if (pendingIndex == -1) {
        Serial.println("[QUEUE] No pending emergency for this service");
        return;
    }
    int emergencyId = pendingQueue[pendingIndex].emergencyId;
    Serial.println("[QUEUE] Pending emergency assigned to returned vehicle");
    Serial.print("[QUEUE] Emergency ID: ");
    Serial.println(emergencyId);
    pendingQueue[pendingIndex].active = false;
    dispatchVehicle(vehicleIndex, emergencyId);
}

void clearPendingQueue() {
    for (int i = 0; i < MAX_PENDING; i++) {
        pendingQueue[i].active = false;
    }
}

void addPendingEmergency(int emergencyId, String serviceType) {
    int slot = findFreePendingSlot();
    if (slot == -1) {
        Serial.println("[QUEUE] Pending queue is full. Emergency lost.");
        return;
    }
    pendingQueue[slot].active = true;
    pendingQueue[slot].emergencyId = emergencyId;
    pendingQueue[slot].serviceType = serviceType;
    pendingQueue[slot].createdAt = millis();
    Serial.println("[QUEUE] Emergency saved");
    Serial.print("[QUEUE] Emergency ID: ");
    Serial.println(emergencyId);
    Serial.print("[QUEUE] Required service: ");
    Serial.println(serviceType);
}

int findFreePendingSlot() {
    for (int i = 0; i < MAX_PENDING; i++) {
        if (!pendingQueue[i].active) {
            return i;
        }
    }
    return -1;
}

int findOldestPendingByService(String serviceType) {
    int selectedIndex = -1;
    unsigned long oldestTime = ULONG_MAX;
    for (int i = 0; i < MAX_PENDING; i++) {
        if (!pendingQueue[i].active) continue;
        if (pendingQueue[i].serviceType == serviceType) {
            if (pendingQueue[i].createdAt < oldestTime) {
                oldestTime = pendingQueue[i].createdAt;
                selectedIndex = i;
            }
        }
    }
    return selectedIndex;
}

int findAvailableVehicle(String serviceType) {
    for (int i = 0; i < VEHICLE_COUNT; i++) {

```

```

        if (vehicles[i].available && vehicles[i].serviceType == serviceType)
    {
        return i;
    }
    }
    return -1;
}
int findVehicleById(String vehicleId) {
    for (int i = 0; i < VEHICLE_COUNT; i++) {
        if (vehicles[i].vehicleId == vehicleId) {
            return i;
        }
    }
    return -1;
}
bool splitMessage(String message, String parts[], int expectedParts) {
    int partIndex = 0;
    int startIndex = 0;
    while (partIndex < expectedParts) {
        int separatorIndex = message.indexOf(';', startIndex);
        if (separatorIndex == -1) {
            if (partIndex == expectedParts - 1) {
                parts[partIndex] = message.substring(startIndex);
                return true;
            }
            return false;
        }
        parts[partIndex] = message.substring(startIndex, separatorIndex);
        startIndex = separatorIndex + 1;
        partIndex++;
    }
    return false;
}
#include <WiFi.h>
#include <PubSubClient.h>
const char* WIFI_SSID = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";
const char* MQTT_SERVER = "broker.hivemq.com";
const int MQTT_PORT = 1883;
const char* CALL_TOPIC = "smartcitydispatchcall";
const char* VEHICLE_DONE_TOPIC = "smartcityvehicledone";
const char* VEHICLE_STATUS_TOPIC = "smartcityvehiclestatus";
WiFiClient espClient;
PubSubClient mqttClient(espClient);
struct VehicleTrip {
    bool active;
    String emergencyId;
    String vehicleId;
    String serviceType;
    String rfidTag;
    String direction;
    int distanceToIntersection;
    int counter;
    unsigned long lastMovementUpdate;
};
const int MAX_ACTIVE_TRIPS = 5;
VehicleTrip activeTrips[MAX_ACTIVE_TRIPS];
const unsigned long MOVEMENT_INTERVAL = 1000;
const int DISTANCE_STEP = 10;
const int MIN_START_DISTANCE = 80;
const int MAX_START_DISTANCE = 140;
void setup() {
    Serial.begin(115200);
    randomSeed(esp_random());

```

```

clearTrips();
connectWiFi();
mqttClient.setServer(MQTT_SERVER, MQTT_PORT);
mqttClient.setCallback(onMqttMessage);
connectMQTT();
Serial.println("=== Multi Vehicle Module Started ===");
Serial.println("[SYSTEM] Waiting for dispatch calls...");
}
void loop() {
    if (!mqttClient.connected()) {
        connectMQTT();
    }
    mqttClient.loop();
    simulateAllVehicles();
}
void connectWiFi() {
    Serial.print("[WIFI] Connecting");
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD, 6);
    while (WiFi.status() != WL_CONNECTED) {
        delay(250);
        Serial.print(".");
    }
    Serial.println();
    Serial.println("[WIFI] Connected");
}
void connectMQTT() {
    while (!mqttClient.connected()) {
        String clientId = "vehicle-module-" + String(random(0xffff), HEX);
        Serial.print("[MQTT] Connecting... ");
        if (mqttClient.connect(clientId.c_str())) {
            Serial.println("connected");
            mqttClient.subscribe(CALL_TOPIC);
            Serial.print("[MQTT] Subscribed to: ");
            Serial.println(CALL_TOPIC);
        } else {
            Serial.print("failed, rc=");
            Serial.println(mqttClient.state());
            delay(2000);
        }
    }
}
void onMqttMessage(char* topic, byte* payload, unsigned int length) {
    String message = "";
    for (unsigned int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    Serial.println();
    Serial.print("[MQTT] Dispatch message received: ");
    Serial.println(message);
    handleDispatchCall(message);
}
void handleDispatchCall(String message) {
    if (!message.startsWith("CALL;")) {
        Serial.println("[IGNORE] Unknown message type");
        return;
    }
    String parts[5];
    if (!splitMessage(message, parts, 5)) {
        Serial.println("[ERROR] Invalid CALL message format");
        return;
    }
    String emergencyId = parts[1];
    String vehicleId = parts[2];
    String serviceType = parts[3];

```

```

String rfidTag = parts[4];
if (isVehicleAlreadyActive(vehicleId)) {
    Serial.print("[IGNORE] Vehicle already active: ");
    Serial.println(vehicleId);
    return;
}
int slot = findFreeTripSlot();
if (slot == -1) {
    Serial.println("[ERROR] No free trip slots. Call ignored.");
    return;
}
startVehicleTrip(slot, emergencyId, vehicleId, serviceType, rfidTag);
}

void startVehicleTrip(
    int index,
    String emergencyId,
    String vehicleId,
    String serviceType,
    String rfidTag
) {
    activeTrips[index].active = true;
    activeTrips[index].emergencyId = emergencyId;
    activeTrips[index].vehicleId = vehicleId;
    activeTrips[index].serviceType = serviceType;
    activeTrips[index].rfidTag = rfidTag;
    activeTrips[index].direction = getRandomDirection();
    activeTrips[index].distanceToIntersection = random(MIN_START_DISTANCE,
MAX_START_DISTANCE + 1);
    activeTrips[index].counter = 0;
    activeTrips[index].lastMovementUpdate = millis();
    Serial.println("[CALL] Dispatch call accepted");
    Serial.print("[DATA] Slot: ");
    Serial.println(index);
    Serial.print("[DATA] Emergency ID: ");
    Serial.println(activeTrips[index].emergencyId);
    Serial.print("[DATA] Vehicle ID: ");
    Serial.println(activeTrips[index].vehicleId);
    Serial.print("[DATA] Service Type: ");
    Serial.println(activeTrips[index].serviceType);
    Serial.print("[DATA] RFID Tag: ");
    Serial.println(activeTrips[index].rfidTag);
    Serial.print("[DATA] Direction: ");
    Serial.println(activeTrips[index].direction);
    Serial.print("[DATA] Start distance: ");
    Serial.println(activeTrips[index].distanceToIntersection);
    sendVehicleStatus(index);
}

void finishVehicleTrip(int index) {
    Serial.println();
    Serial.print("[TRIP] Vehicle passed intersection: ");
    Serial.println(activeTrips[index].vehicleId);
    sendVehicleStatus(index);
    sendVehicleDone(index);
    activeTrips[index].active = false;
    Serial.print("[SYSTEM] Slot ");
    Serial.print(index);
    Serial.println(" released");
}

void sendVehicleDone(int index) {
    String payload = "DONE;";
    payload += activeTrips[index].vehicleId;
    payload += ";";
    payload += activeTrips[index].serviceType;
    payload += ";";
}

```

```

        payload += activeTrips[index].rfidTag;
        mqttClient.publish(VEHICLE_DONE_TOPIC, payload.c_str());
        Serial.print("[MQTT] Published done: ");
        Serial.println(payload);
    }
    void simulateAllVehicles() {
        for (int i = 0; i < MAX_ACTIVE_TRIPS; i++) {
            if (!activeTrips[i].active) continue;
            if (millis() - activeTrips[i].lastMovementUpdate < MOVEMENT_INTERVAL)
            {
                continue;
            }
            activeTrips[i].lastMovementUpdate = millis();
            activeTrips[i].distanceToIntersection -= DISTANCE_STEP;
            if (activeTrips[i].distanceToIntersection < 0) {
                activeTrips[i].distanceToIntersection = 0;
            }
            activeTrips[i].counter++;
            sendVehicleStatus(i);
            if (activeTrips[i].distanceToIntersection == 0) {
                finishVehicleTrip(i);
            }
        }
    }
    void sendVehicleStatus(int index) {
        String payload = "VEHICLE;";
        payload += activeTrips[index].emergencyId;
        payload += ";";
        payload += activeTrips[index].vehicleId;
        payload += ";";
        payload += activeTrips[index].serviceType;
        payload += ";";
        payload += activeTrips[index].rfidTag;
        payload += ";";
        payload += activeTrips[index].direction;
        payload += ";";
        payload += String(activeTrips[index].distanceToIntersection);
        mqttClient.publish(VEHICLE_STATUS_TOPIC, payload.c_str());
        Serial.print("[MQTT] Published status: ");
        Serial.println(payload);
    }
    void clearTrips() {
        for (int i = 0; i < MAX_ACTIVE_TRIPS; i++) {
            activeTrips[i].active = false;
        }
    }
    int findFreeTripSlot() {
        for (int i = 0; i < MAX_ACTIVE_TRIPS; i++) {
            if (!activeTrips[i].active) {
                return i;
            }
        }
        return -1;
    }
    bool isVehicleAlreadyActive(String vehicleId) {
        for (int i = 0; i < MAX_ACTIVE_TRIPS; i++) {
            if (activeTrips[i].active && activeTrips[i].vehicleId == vehicleId)
            {
                return true;
            }
        }
        return false;
    }
    String getRandomDirection() {

```

```

    int value = random(0, 4);
    if (value == 0) return "NORTH";
    if (value == 1) return "SOUTH";
    if (value == 2) return "EAST";
    return "WEST";
}
bool splitMessage(String message, String parts[], int expectedParts) {
    int partIndex = 0;
    int startIndex = 0;
    while (partIndex < expectedParts) {
        int separatorIndex = message.indexOf(';', startIndex);
        if (separatorIndex == -1) {
            if (partIndex == expectedParts - 1) {
                parts[partIndex] = message.substring(startIndex);
                return true;
            }
            return false;
        }
        parts[partIndex] = message.substring(startIndex, separatorIndex);
        startIndex = separatorIndex + 1;
        partIndex++;
    }
    return false;
}
#include <WiFi.h>
#include <PubSubClient.h>
const char* WIFI_SSID = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";
const char* MQTT_SERVER = "broker.hivemq.com";
const int MQTT_PORT = 1883;
const char* VEHICLE_STATUS_TOPIC = "smartcityvehiclestatus";
const char* INTERSECTION_STATUS_TOPIC = "smartcityintersectionstatus";
const char* VEHICLES_UI_TOPIC = "smartcityintersectionvehicles";
const char* OPERATOR_TOPIC = "smartcityintersectionoperator";
WiFiClient espClient;
PubSubClient mqttClient(espClient);
const int N_RED = 13;
const int N_YELLOW = 12;
const int N_GREEN = 14;
const int S_RED = 27;
const int S_YELLOW = 26;
const int S_GREEN = 25;
const int E_RED = 33;
const int E_YELLOW = 32;
const int E_GREEN = 23;
const int W_RED = 22;
const int W_YELLOW = 21;
const int W_GREEN = 19;
const int LOAD_BUTTON = 18;
const unsigned long GREEN_TIME = 5000;
const unsigned long YELLOW_TIME = 2000;
const unsigned long ALL_RED_TIME = 1000;
const unsigned long MAX_PRIORITY_HOLD_TIME = 200000;
const unsigned long STATUS_INTERVAL = 1000;
const unsigned long VEHICLES_INTERVAL = 1000;
const String allowedRfidTags[] = {
    "RFID-AMB-001",
    "RFID-AMB-002",
    "RFID-FIRE-001",
    "RFID-FIRE-002",
    "RFID-POL-001",
    "RFID-POL-002"
};

```

```

        const int ALLOWED_RFID_COUNT = sizeof(allowedRfidTags)
sizeof(allowedRfidTags[0]);
enum TrafficPhase {
    NS_GREEN,
    NS_YELLOW,
    EW_GREEN,
    EW_YELLOW,
    ALL_RED
};
enum TrafficLoad {
    LOW_LOAD,
    MEDIUM_LOAD,
    HIGH_LOAD
};
TrafficPhase currentPhase = ALL_RED;
TrafficLoad currentLoad = MEDIUM_LOAD;
struct EmergencyRequest {
    bool active;
    String emergencyId;
    String vehicleId;
    String serviceType;
    String rfidTag;
    String direction;
    int distance;
    unsigned long lastUpdateTime;
};
const int MAX_REQUESTS = 10;
EmergencyRequest requests[MAX_REQUESTS];
bool priorityActive = false;
bool priorityVehiclePassed = false;
bool manualMode = false;
String priorityVehicleId = "";
String priorityDirection = "";
String priorityServiceType = "";
int priorityVehicleDistance = -1;
unsigned long priorityStartTime = 0;
int lastLoadButtonState;
int currentLoadButtonState;
unsigned long lastPhaseChange = 0;
int normalStep = 0;
unsigned long lastStatusPublish = 0;
unsigned long lastVehiclesPublish = 0;
void setup() {
    Serial.begin(115200);
    setupTrafficPins();
    pinMode(LOAD_BUTTON, INPUT_PULLUP);
    currentLoadButtonState = digitalRead(LOAD_BUTTON);
    clearRequests();
    setAllRed();
    connectWiFi();
    mqttClient.setServer(MQTT_SERVER, MQTT_PORT);
    mqttClient.setCallback(onMqttMessage);
    connectMQTT();
    Serial.println("=== Intersection Module Started ===");
    Serial.println("[SYSTEM] RFID + priority queue + web panel
integration");
    printCurrentLoad();
}
void loop() {
    if (!mqttClient.connected()) {
        connectMQTT();
    }
    mqttClient.loop();
    handleLoadButton();
}

```

```

    publishIntersectionStatus();
    publishVehiclesStatus();
    if (!priorityActive && !manualMode) {
        checkEmergencyRequests();
        runNormalTrafficCycleStep();
    }
}
void connectWiFi() {
    Serial.print("[WIFI] Connecting");
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD, 6);
    while (WiFi.status() != WL_CONNECTED) {
        delay(250);
        Serial.print(".");
    }
    Serial.println();
    Serial.println("[WIFI] Connected");
}
void connectMQTT() {
    while (!mqttClient.connected()) {
        String clientId = "intersection-module-" + String(random(0xffff),
HEX);

        Serial.print("[MQTT] Connecting... ");
        if (mqttClient.connect(clientId.c_str())) {
            Serial.println("connected");
            mqttClient.subscribe(VEHICLE_STATUS_TOPIC);
            mqttClient.subscribe(OPERATOR_TOPIC);
            Serial.print("[MQTT] Subscribed to: ");
            Serial.println(VEHICLE_STATUS_TOPIC);
            Serial.print("[MQTT] Subscribed to: ");
            Serial.println(OPERATOR_TOPIC);
        } else {
            Serial.print("failed, rc=");
            Serial.println(mqttClient.state());
            delay(2000);
        }
    }
}
void onMqttMessage(char* topic, byte* payload, unsigned int length) {
    String message = "";
    for (unsigned int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    String topicString = String(topic);
    Serial.println();
    Serial.print("[MQTT] Message received on ");
    Serial.print(topicString);
    Serial.print(": ");
    Serial.println(message);
    if (topicString == VEHICLE_STATUS_TOPIC) {
        handleVehicleStatus(message);
        return;
    }
    if (topicString == OPERATOR_TOPIC) {
        handleOperatorCommand(message);
        return;
    }
}
void handleVehicleStatus(String message) {
    if (!message.startsWith("VEHICLE;")) {
        Serial.println("[IGNORE] Unknown message type");
        return;
    }
    String parts[7];
    if (!splitMessage(message, parts, 7)) {

```

```

        Serial.println("[ERROR] Invalid VEHICLE message format");
        return;
    }
    String emergencyId = parts[1];
    String vehicleId = parts[2];
    String serviceType = parts[3];
    String rfidTag = parts[4];
    String direction = parts[5];
    int distance = parts[6].toInt();
    Serial.print("[DATA] Vehicle ID: ");
    Serial.println(vehicleId);
    Serial.print("[DATA] Service: ");
    Serial.println(serviceType);
    Serial.print("[DATA] RFID: ");
    Serial.println(rfidTag);
    Serial.print("[DATA] Direction: ");
    Serial.println(direction);
    Serial.print("[DATA] Distance: ");
    Serial.println(distance);
    if (!isRfidAllowed(rfidTag)) {
        Serial.println("[SECURITY] RFID is not allowed. Vehicle ignored.");
        blinkAllYellowOnce();
        return;
    }
    Serial.println("[SECURITY] RFID verified. Vehicle is authorized.");
    if (distance <= 0) {
        Serial.println("[VEHICLE] Vehicle reached intersection. Removing from
system.");
        removeRequestByVehicleId(vehicleId);
        if (priorityActive && vehicleId == priorityVehicleId) {
            priorityVehicleDistance = 0;
            priorityVehiclePassed = true;
            Serial.println();
            Serial.print("[PRIORITY] Vehicle ");
            Serial.print(vehicleId);
            Serial.println(" passed the intersection");
        }
        lastVehiclesPublish = 0;
        publishVehiclesStatus();
        return;
    }
    if (priorityActive && vehicleId == priorityVehicleId) {
        priorityVehicleDistance = distance;
        Serial.print("[PRIORITY] Active vehicle still approaching, distance:
");

        Serial.println(distance);
        lastVehiclesPublish = 0;
        publishVehiclesStatus();
        return;
    }
    updateEmergencyRequest(
        emergencyId,
        vehicleId,
        serviceType,
        rfidTag,
        direction,
        distance
    );
}
void clearRequests() {
    for (int i = 0; i < MAX_REQUESTS; i++) {
        requests[i].active = false;
    }
}

```

```

void updateEmergencyRequest(
    String emergencyId,
    String vehicleId,
    String serviceType,
    String rfidTag,
    String direction,
    int distance
) {
    int index = findRequestByVehicleId(vehicleId);
    if (index == -1) {
        index = findFreeRequestSlot();
    }
    if (index == -1) {
        Serial.println("[QUEUE] Emergency queue is full. Request ignored.");
        return;
    }
    requests[index].active = true;
    requests[index].emergencyId = emergencyId;
    requests[index].vehicleId = vehicleId;
    requests[index].serviceType = serviceType;
    requests[index].rfidTag = rfidTag;
    requests[index].direction = direction;
    requests[index].distance = distance;
    requests[index].lastUpdateTime = millis();
    Serial.println("[QUEUE] Emergency request updated");
}

int findRequestByVehicleId(String vehicleId) {
    for (int i = 0; i < MAX_REQUESTS; i++) {
        if (requests[i].active && requests[i].vehicleId == vehicleId) {
            return i;
        }
    }
    return -1;
}

int findFreeRequestSlot() {
    for (int i = 0; i < MAX_REQUESTS; i++) {
        if (!requests[i].active) {
            return i;
        }
    }
    return -1;
}

void checkEmergencyRequests() {
    int selectedIndex = findClosestVehicleForPriority();
    if (selectedIndex == -1) {
        return;
    }
    EmergencyRequest selected = requests[selectedIndex];
    requests[selectedIndex].active = false;
    Serial.println();
    Serial.println("[DECISION] Conflict resolver selected emergency
vehicle");
    Serial.print("[DECISION] Selected vehicle: ");
    Serial.println(selected.vehicleId);
    Serial.print("[DECISION] Service: ");
    Serial.println(selected.serviceType);
    Serial.print("[DECISION] Direction: ");
    Serial.println(selected.direction);
    Serial.print("[DECISION] Distance: ");
    Serial.println(selected.distance);
    activatePriorityMode(
        selected.direction,
        selected.vehicleId,
        selected.serviceType,

```

```

        selected.distance
    );
}
int findClosestVehicleForPriority() {
    int threshold = getActivationThreshold();
    int selectedIndex = -1;
    int closestDistance = 9999;
    for (int i = 0; i < MAX_REQUESTS; i++) {
        if (!requests[i].active) continue;
        if (millis() - requests[i].lastUpdateTime > 10000) {
            Serial.print("[QUEUE] Removed outdated request: ");
            Serial.println(requests[i].vehicleId);
            requests[i].active = false;
            continue;
        }
        if (requests[i].distance <= threshold) {
            if (requests[i].distance < closestDistance) {
                closestDistance = requests[i].distance;
                selectedIndex = i;
            }
        }
    }
    return selectedIndex;
}

void activatePriorityMode(String direction, String vehicleId, String
serviceType, int startDistance) {
    priorityActive = true;
    priorityVehicleId = vehicleId;
    priorityDirection = direction;
    priorityServiceType = serviceType;
    priorityVehicleDistance = startDistance;
    priorityVehiclePassed = false;
    priorityStartTime = millis();
    Serial.println("[PRIORITY] Preparing green corridor");
    if (isDirectionAlreadyGreen(direction)) {
        Serial.println("[PRIORITY] Required direction is already green");
    } else {
        switchToSafeAllRed();
    }
    setPriorityGreen(direction);
    Serial.print("[PRIORITY] Green corridor active for ");
    Serial.println(direction);
    Serial.print("[PRIORITY] Waiting until vehicle ");
    Serial.print(vehicleId);
    Serial.println(" passes the intersection");
    while (!priorityVehiclePassed) {
        if (!mqttClient.connected()) {
            connectMQTT();
        }
        mqttClient.loop();
        publishIntersectionStatus();
        publishVehiclesStatus();
        if (millis() - priorityStartTime > MAX_PRIORITY_HOLD_TIME) {
            Serial.println("[PRIORITY] Max hold time reached. Returning to
normal mode");
            break;
        }
        delay(50);
    }
    Serial.println("[PRIORITY] Priority finished. Returning to normal
mode");
    setAllRed();
    delay(ALL_RED_TIME);
    priorityVehicleId = "";
}

```

```

    priorityDirection = "";
    priorityServiceType = "";
    priorityVehicleDistance = -1;
    priorityVehiclePassed = false;
    lastPhaseChange = millis();
    normalStep = 0;
    priorityActive = false;
    lastStatusPublish = 0;
    lastVehiclesPublish = 0;
    publishIntersectionStatus();
    publishVehiclesStatus();
}
int getActivationThreshold() {
    if (currentLoad == LOW_LOAD) return 25;
    if (currentLoad == MEDIUM_LOAD) return 45;
    return 70;
}
void handleLoadButton() {
    lastLoadButtonState = currentLoadButtonState;
    currentLoadButtonState = digitalRead(Load_BUTTON);
    if (lastLoadButtonState == HIGH && currentLoadButtonState == LOW) {
        changeTrafficLoad();
        delay(150);
    }
}
void changeTrafficLoad() {
    if (currentLoad == LOW_LOAD) {
        currentLoad = MEDIUM_LOAD;
    } else if (currentLoad == MEDIUM_LOAD) {
        currentLoad = HIGH_LOAD;
    } else {
        currentLoad = LOW_LOAD;
    }
    printCurrentLoad();
}
void printCurrentLoad() {
    Serial.println();
    if (currentLoad == LOW_LOAD) {
        Serial.println("[LOAD] Current traffic load: LOW");
        Serial.println("[LOAD] Activation threshold: 25 m");
    } else if (currentLoad == MEDIUM_LOAD) {
        Serial.println("[LOAD] Current traffic load: MEDIUM");
        Serial.println("[LOAD] Activation threshold: 45 m");
    } else {
        Serial.println("[LOAD] Current traffic load: HIGH");
        Serial.println("[LOAD] Activation threshold: 70 m");
    }
}
void handleOperatorCommand(String command) {
    Serial.println();
    Serial.print("[OPERATOR] Command received: ");
    Serial.println(command);
    if (priorityActive && command != "LOAD_LOW" && command != "LOAD_MEDIUM"
    && command != "LOAD_HIGH") {
        Serial.println("[OPERATOR] Command ignored: priority mode is
active");
        return;
    }
    if (command == "AUTO") {
        manualMode = false;
        Serial.println("[OPERATOR] Auto mode enabled");
        lastPhaseChange = millis();
        normalStep = 0;
        publishIntersectionStatus();
    }
}

```

```

        return;
    }
    if (command == "ALL_RED") {
        manualMode = true;
        setAllRed();
    } else if (command == "NS_GREEN") {
        manualMode = true;
        setNorthSouthGreen();
    } else if (command == "EW_GREEN") {
        manualMode = true;
        setEastWestGreen();
    } else if (command == "LOAD_LOW") {
        currentLoad = LOW_LOAD;
        printCurrentLoad();
    } else if (command == "LOAD_MEDIUM") {
        currentLoad = MEDIUM_LOAD;
        printCurrentLoad();
    } else if (command == "LOAD_HIGH") {
        currentLoad = HIGH_LOAD;
        printCurrentLoad();
    } else {
        Serial.println("[OPERATOR] Unknown command");
    }
    publishIntersectionStatus();
}

void publishIntersectionStatus() {
    if (millis() - lastStatusPublish < STATUS_INTERVAL) {
        return;
    }
    lastStatusPublish = millis();
    String payload = "STATUS;";
    payload += manualMode ? "MANUAL" : "AUTO";
    payload += ";";
    payload += trafficPhaseToString(currentPhase);
    payload += ";";
    payload += trafficLoadToString(currentLoad);
    payload += ";";
    payload += priorityActive ? "YES" : "NO";
    payload += ";";
    payload += priorityVehicleId == "" ? "-" : priorityVehicleId;
    payload += ";";
    payload += priorityDirection == "" ? "-" : priorityDirection;
    mqttClient.publish(INTERSECTION_STATUS_TOPIC, payload.c_str());
}

void publishVehiclesStatus() {
    if (millis() - lastVehiclesPublish < VEHICLES_INTERVAL) {
        return;
    }
    lastVehiclesPublish = millis();
    String payload = "VEHICLES;";
    bool hasVehicles = false;
    if (priorityActive && priorityVehicleId != "") {
        payload += priorityVehicleId;
        payload += ";";
        payload += priorityServiceType;
        payload += ";";
        payload += priorityDirection;
        payload += ";";
        payload += String(priorityVehicleDistance);
        payload += ";";
        payload += "PRIORITY";
        hasVehicles = true;
    }
    for (int i = 0; i < MAX_REQUESTS; i++) {

```

```

        if (!requests[i].active) continue;
        if (priorityActive && requests[i].vehicleId == priorityVehicleId) {
            continue;
        }
        if (hasVehicles) {
            payload += "|";
        }
        payload += requests[i].vehicleId;
        payload += ";";
        payload += requests[i].serviceType;
        payload += ";";
        payload += requests[i].direction;
        payload += ";";
        payload += String(requests[i].distance);
        payload += ";";
        payload += "WAITING";
        hasVehicles = true;
    }
    if (!hasVehicles) {
        payload += "EMPTY";
    }
    mqttClient.publish(VEHICLES_UI_TOPIC, payload.c_str());
}

void setupTrafficPins() {
    pinMode(N_RED, OUTPUT);
    pinMode(N_YELLOW, OUTPUT);
    pinMode(N_GREEN, OUTPUT);
    pinMode(S_RED, OUTPUT);
    pinMode(S_YELLOW, OUTPUT);
    pinMode(S_GREEN, OUTPUT);
    pinMode(E_RED, OUTPUT);
    pinMode(E_YELLOW, OUTPUT);
    pinMode(E_GREEN, OUTPUT);
    pinMode(W_RED, OUTPUT);
    pinMode(W_YELLOW, OUTPUT);
    pinMode(W_GREEN, OUTPUT);
}

void runNormalTrafficCycleStep() {
    unsigned long now = millis();
    if (now - lastPhaseChange < getCurrentStepDuration(normalStep)) {
        return;
    }
    lastPhaseChange = now;
    if (normalStep == 0) {
        setNorthSouthGreen();
    } else if (normalStep == 1) {
        setNorthSouthYellow();
    } else if (normalStep == 2) {
        setAllRed();
    } else if (normalStep == 3) {
        setEastWestGreen();
    } else if (normalStep == 4) {
        setEastWestYellow();
    } else if (normalStep == 5) {
        setAllRed();
    }
    normalStep++;
    if (normalStep > 5) {
        normalStep = 0;
    }
}

unsigned long getCurrentStepDuration(int step) {
    if (step == 0 || step == 3) return GREEN_TIME;
    if (step == 1 || step == 4) return YELLOW_TIME;

```

```

        return ALL_RED_TIME;
    }
    bool isDirectionAlreadyGreen(String direction) {
        if ((direction == "NORTH" || direction == "SOUTH") && currentPhase ==
NS_GREEN) {
            return true;
        }
        if ((direction == "EAST" || direction == "WEST") && currentPhase ==
EW_GREEN) {
            return true;
        }
        return false;
    }
    void switchToSafeAllRed() {
        if (currentPhase == NS_GREEN) {
            setNorthSouthYellow();
            delay(YELLOW_TIME);
        } else if (currentPhase == EW_GREEN) {
            setEastWestYellow();
            delay(YELLOW_TIME);
        }
        setAllRed();
        delay(ALL_RED_TIME);
    }
    void setPriorityGreen(String direction) {
        if (direction == "NORTH" || direction == "SOUTH") {
            setNorthSouthGreen();
        } else {
            setEastWestGreen();
        }
    }
    void turnOffAllLights() {
        digitalWrite(N_RED, LOW);
        digitalWrite(N_YELLOW, LOW);
        digitalWrite(N_GREEN, LOW);
        digitalWrite(S_RED, LOW);
        digitalWrite(S_YELLOW, LOW);
        digitalWrite(S_GREEN, LOW);
        digitalWrite(E_RED, LOW);
        digitalWrite(E_YELLOW, LOW);
        digitalWrite(E_GREEN, LOW);
        digitalWrite(W_RED, LOW);
        digitalWrite(W_YELLOW, LOW);
        digitalWrite(W_GREEN, LOW);
    }
    void setAllRed() {
        currentPhase = ALL_RED;
        turnOffAllLights();
        digitalWrite(N_RED, HIGH);
        digitalWrite(S_RED, HIGH);
        digitalWrite(E_RED, HIGH);
        digitalWrite(W_RED, HIGH);
        Serial.println("[STATE] ALL RED");
    }
    void setNorthSouthGreen() {
        currentPhase = NS_GREEN;
        turnOffAllLights();
        digitalWrite(N_GREEN, HIGH);
        digitalWrite(S_GREEN, HIGH);
        digitalWrite(E_RED, HIGH);
        digitalWrite(W_RED, HIGH);
        Serial.println("[STATE] NORTH-SOUTH GREEN, EAST-WEST RED");
    }
    void setNorthSouthYellow() {

```

```

    currentPhase = NS_YELLOW;
    turnOffAllLights();
    digitalWrite(N_YELLOW, HIGH);
    digitalWrite(S_YELLOW, HIGH);
    digitalWrite(E_RED, HIGH);
    digitalWrite(W_RED, HIGH);
    Serial.println("[STATE] NORTH-SOUTH YELLOW, EAST-WEST RED");
}

void setEastWestGreen() {
    currentPhase = EW_GREEN;
    turnOffAllLights();
    digitalWrite(E_GREEN, HIGH);
    digitalWrite(W_GREEN, HIGH);
    digitalWrite(N_RED, HIGH);
    digitalWrite(S_RED, HIGH);
    Serial.println("[STATE] EAST-WEST GREEN, NORTH-SOUTH RED");
}

void setEastWestYellow() {
    currentPhase = EW_YELLOW;
    turnOffAllLights();
    digitalWrite(E_YELLOW, HIGH);
    digitalWrite(W_YELLOW, HIGH);
    digitalWrite(N_RED, HIGH);
    digitalWrite(S_RED, HIGH);
    Serial.println("[STATE] EAST-WEST YELLOW, NORTH-SOUTH RED");
}

void blinkAllYellowOnce() {
    digitalWrite(N_YELLOW, HIGH);
    digitalWrite(S_YELLOW, HIGH);
    digitalWrite(E_YELLOW, HIGH);
    digitalWrite(W_YELLOW, HIGH);
    delay(200);
    digitalWrite(N_YELLOW, LOW);
    digitalWrite(S_YELLOW, LOW);
    digitalWrite(E_YELLOW, LOW);
    digitalWrite(W_YELLOW, LOW);
}

bool isRfidAllowed(String rfidTag) {
    for (int i = 0; i < ALLOWED_RFID_COUNT; i++) {
        if (allowedRfidTags[i] == rfidTag) {
            return true;
        }
    }
    return false;
}

String trafficPhaseToString(TrafficPhase phase) {
    if (phase == NS_GREEN) return "NS_GREEN";
    if (phase == NS_YELLOW) return "NS_YELLOW";
    if (phase == EW_GREEN) return "EW_GREEN";
    if (phase == EW_YELLOW) return "EW_YELLOW";
    return "ALL_RED";
}

String trafficLoadToString(TrafficLoad load) {
    if (load == LOW_LOAD) return "LOW";
    if (load == MEDIUM_LOAD) return "MEDIUM";
    return "HIGH";
}

bool splitMessage(String message, String parts[], int expectedParts) {
    int partIndex = 0;
    int startIndex = 0;
    while (partIndex < expectedParts) {
        int separatorIndex = message.indexOf(';', startIndex);
        if (separatorIndex == -1) {
            if (partIndex == expectedParts - 1) {

```

```

        parts[partIndex] = message.substring(startIndex);
        return true;
    }
    return false;
}
parts[partIndex] = message.substring(startIndex, separatorIndex);
startIndex = separatorIndex + 1;
partIndex++;
}
return false;
}
void removeRequestByVehicleId(String vehicleId) {
    for (int i = 0; i < MAX_REQUESTS; i++) {
        if (requests[i].active && requests[i].vehicleId == vehicleId) {
            requests[i].active = false;
            Serial.print("[QUEUE] Removed vehicle from queue: ");
            Serial.println(vehicleId);
        }
    }
}
}

```

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Олександр СУХОВІРСЬКИЙ

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

Anti-Plagiarism (<http://ap.km.ua>) v-15.701

Максимальне співпадіння з одним документом 13.0%

Словники перевірки: en_US, ru_RU, ua_UA. Помилки в документах: 13%

ID: 272549 Назва: БКР Кіберфізична система пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій Додано в БД: 2026-05-28 Автора: Олександр СУХОВІРСЬКИЙ Керівник: Олександр ГУРАЛЬНИК Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	113044	820	15701 (14%)	118 (14%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
269546	Назва: Звіт з ПДП Кіберфізична система пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій Додано в БД: 2026-02-26 Автора: Суховірський О.С. Керівник: Павлова О.О. Консультанти: Опоненти:	14840 (13.0%)	113 (14.0%)

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Олександр СУХОВІРСЬКИЙ

Співавтор:

Назва: Кіберфізична система пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій

Експерт: Олександр ГУРАЛЬНИК

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1: 3.13%

Коефіцієнт подібності 2: 1%

Мікропробіли: 3

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-27 21:08:09.0

Після аналізу Звіту подібності констатую наступне:

☒ Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.

☐ Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.

☐ Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедур. Таким чином робота не приймається.

Обґрунтування:

2026-05-28

Дата

Доцент Андрій Нічепорук

експерт

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Суховірському Олександру Сергійовичу

Тема: Кіберфізична система пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 60

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проектування, програмна реалізація та моделювання кіберфізичної системи пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі мікроконтролерної платформи ESP32.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі кваліфікаційної роботи проведено дослідження предметної області та виконано постановку задачі. Розглянуто проблему затримки спеціалізованого транспорту на регульованих перехрестях, проаналізовано статистичний вплив таких затримок на ефективність реагування екстрених служб, досліджено концепцію інтелектуальних транспортних систем та Smart City. Також виконано порівняння сучасних рішень пріоритетного пропуску спецтранспорту. На основі аналізу визначено функціональні й нефункціональні вимоги до системи та сформульовано постановку задачі.

У другому розділі кваліфікаційної роботи виконано проектування кіберфізичної системи пріоритетного пропуску спецтранспорту. Розроблено загальну гібридну розподілену архітектуру системи, у якій модуль диспетчеризації, модуль «Автомобіль», модуль «Перехрестя», MQTT-брокер та веб-інтерфейс оператора взаємодіють у межах єдиної інформаційної інфраструктури. Обґрунтовано

використання мікроконтролерів ESP32 як базової апаратної платформи. Спроектовано підсистеми модуля диспетчеризації, модуля автомобіля та модуля «Перехрестя», розроблено алгоритм адаптивного керування світлофорними фазами, визначено режими роботи системи NORMAL, PRIORITY, MANUAL та SAFE, описано структуру MQTT-топиків і форматів повідомлень, а також розроблено веб-інтерфейс моніторингу стану перехрестя.

У третьому розділі кваліфікаційної роботи виконано програмну реалізацію, моделювання та тестування кіберфізичної системи. Реалізовано віртуальну схему системи на основі ESP32 у середовищі Wokwi, програмне забезпечення модуля диспетчеризації, модуля «Автомобіль» та модуля «Перехрестя». Реалізовано інформаційну взаємодію на основі MQTT-протоколу із використанням публічного MQTT-брокера HiveMQ, а також веб-інтерфейс оператора для відображення стану системи в режимі реального часу та ручного керування світлофорними фазами. Проведено тестування штатного режиму роботи, обробки одного й кількох екстрених викликів, функціонування ручного режиму. Результати тестування підтвердили працездатність запропонованої архітектури та коректність реалізованих алгоритмів.

4. Позитивні сторони роботи: реалізація модульної архітектури з окремими підсистемами диспетчеризації, автомобіля, перехрестя та веб-моніторингу.

5. Негативні сторони роботи: окремі елементи системи реалізовано в імітаційному вигляді, зокрема RFID-ідентифікацію.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Василь Руденко Олександрович, доц. кафедри КЦ

“ 1 ” 06 2026 р.

 (підпис)

Зав. кафедри КПС
д-р. філософії Ользі ПАВЛЮВІЙ

Олександр СУХОВІРСЬКИЙ

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року



РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Кіберфізична система пріоритетного пропуску спецтранспорту на регульованих перехрестях на основі ESP32 та бездротових ідентифікаційних технологій

Автор Олександр СУХОВІРСЬКИЙ

Освітня програма Комп'ютерна інженерія та програмування

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія

Науковий керівник: Олександр ГУРАЛЬНИК

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

- 1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
- 2) окремі виявлені збіги є загальноновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
- 3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.
- 4) значна частина знайденого плагіату відноситься до списку використаних джерел

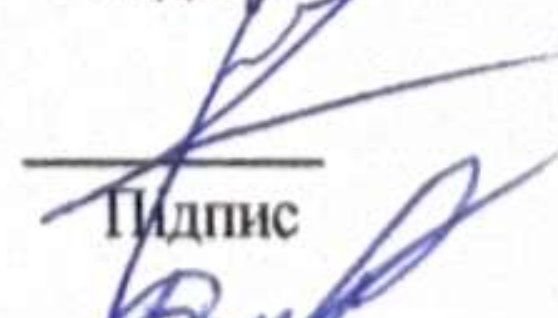
Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 4,28%; та системою Anti-Plagiarism складає 13%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри

Гарант освітньої програми

Керівник кваліфікаційної роботи


Підпис

Підпис

Підпис

Ольга ПАВЛОВА
Ім'я, ПРІЗВИЩЕ

Андрій НІЧЕПОРУК
Ім'я, ПРІЗВИЩЕ

Олександр ГУРАЛЬНИК
Ім'я, ПРІЗВИЩЕ