

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
Факультет комп'ютерно-інтегрованих технологій, мехатроніки і робототехніки
Кафедра робототехніки, електроенергетики та автоматизації
ім. проф. Б.Б. Самотокіна

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи магістра

на тему: «Автоматизована система управління паркуванням малогабаритного електротранспорту»

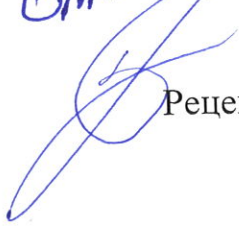
Виконав студент 2 курсу, групи АТ-29м
спеціальності 174 “ Автоматизація,
комп'ютерно-інтегровані технології
та робототехніка”

 Коротя Євгеній Юрійович

(прізвище та ініціали студента)

 Керівник: д.т.н., проф. Безвесільна О.М.

(науковий ступінь, вчене звання, посада,
прізвище та ініціали)

 Рецензент: к.т.н., доц., Громовий О.А.

(науковий ступінь, вчене звання, посада,
прізвище та ініціали)

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ “ЖИТОМИРСЬКА ПОЛІТЕХНІКА”
Факультет комп’ютерно-інтегрованих технологій, мехатроніки і робототехніки
Кафедра робототехніки, електроенергетики та автоматизації
ім. проф. Б.Б. Самотокіна

Спеціальність 174 “Автоматизована система опалення з використанням надлишкового тепла”

ЗАТВЕРДЖУЮ

Завідувач кафедри робототехніки,
електроенергетики та автоматизації
ім. проф. Б.Б. Самотокіна
к.т.н., доц. Громовий О.А.
« 17 » гравтня 2025р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

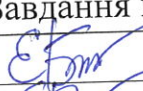
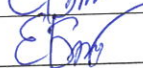
Студент Коротя Євгеній Юрійович

1. Тема роботи: Автоматизована система управління паркуванням малогабаритного електротранспорту
2. Термін здачі закінченої роботи на кафедру «30» листопада 2025 р.
3. Вихідні дані до роботи: матеріали звіту про переддипломну практику, вимоги до паркування, зберігання та заряджання малогабаритного електротранспорту, характеристики IoT-платформ і бездротових каналів зв'язку, протоколи обміну даними MQTT та HTTP, параметри електроживлення й заряджання літій-іонних акумуляторів, вимоги до авторизації користувачів, бази даних і мобільного застосунку, електричного захисту та захисту корпусу системи.
4. Зміст пояснювальної записки (перелік питань, які підлягають розробці):
технічне завдання, огляд та аналіз існуючих систем паркування й зберігання малогабаритного електротранспорту, вибір апаратної платформи, технологій зв'язку та протоколів обміну даними, розробка структурної, функціональної й електричної принципової схем системи, розрахунок енергоспоживання, розробка конструкції та електричного захисту, проектування бази даних, алгоритмів авторизації й функціонування системи, створення інтерфейсу мобільного

застосунку, програмна реалізація та впровадження, техніко-економічне обґрунтування й охорона праці.

5. Перелік презентативного матеріалу (з точним визначенням обов'язкових слайдів/креслень): структурна схема комплексу, функціональна схема взаємодії компонентів, електрична принципова схема підключення та захисту, конструктивна схема паркувального модуля, схема архітектури інформаційної системи, схема бази даних, блок-схеми алгоритмів авторизації та керування паркуванням і заряджанням, екранні форми мобільного застосунку, результати програмної реалізації та техніко-економічного обґрунтування.

6. Консультанти з кваліфікаційної роботи магістра із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1, 2	Безвесільна О.М.		
3,4,5	Безвесільна О.М.		

7. Дата видачі завдання 22.09.2025

Календарний план

№ з/п	Назва етапів виконання кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Підготувати звіт про переддипломну практику	1 тиждень	виконано
2.	Збір та аналіз джерел за темою роботи, огляд існуючих систем	1 тиждень	виконано
3.	Вибір і обґрунтування технічних та програмних засобів	2 тиждень	виконано
4.	Розробка структурної, функціональної та блок-схеми	3 тиждень	виконано
5.	Аналіз отриманих результатів	4 тиждень	виконано
6.	Оформлення пояснювальної записки та презентаційного матеріалу	4 тиждень	виконано

Студент


(підпис, дата)

Є.Ю. Коротя

(прізвище та ініціали)

Керівник роботи


(підпис, дата)

О.М. Безвесільна

(прізвище та ініціали)

ЗМІСТ

ЗМІСТ	1
ВСТУП.....	5
Актуальність теми.....	5
Зв'язок роботи з науковими програмами, планами, темами	5
Мета та завдання дослідження	6
Об'єкт і предмет дослідження.....	6
Методи дослідження	6
Наукова новизна одержаних результатів	7
Практичне значення одержаних результатів.....	7
1.1. Проблеми інтеграції малогабаритного електротранспорту в міську інфраструктуру.....	8
1.2. Аналітичний огляд існуючих систем паркування та зберігання.....	9
1.3. Аналіз апаратних платформ для побудови IoT-систем.....	11
1.4. Аналіз бездротових технологій передачі даних	16
1.5. Аналіз нормативно-правового регулювання використання електротранспорту	17
1.6. Огляд технологій та алгоритмів заряджання літій-іонних акумуляторів	18
1.7. Порівняльний аналіз протоколів прикладного рівня для IoT (MQTT vs HTTP)	19
1.8. Постановка задачі на дипломну роботу.....	20
Висновки до Розділу 1.....	21
РОЗДІЛ 2. РОЗРОБКА СТРУКТУРНО-ФУНКЦІОНАЛЬНОЇ ОРГАНІЗАЦІЇ СИСТЕМИ.....	23
2.1. Розробка структурної схеми комплексу	23
2.2. Розробка функціональної схеми взаємодії	25
2.3. Обґрунтування вибору комплексу технічних засобів	26
2.4. Розробка електричної принципової схеми підключення	28
2.5. Розрахунок енергоспоживання системи	30
2.6. Конструктивне виконання та захист оболонки (Mechanical Design)	30
2.7. Розробка схем електричного захисту (Circuit Protection)	31

Висновки до Розділу 2.....	33
РОЗДІЛ 3. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ІНТЕРФЕЙСІВ СИСТЕМИ	35
3.1. Архітектура інформаційної системи та обґрунтування вибору технологій	35
3.2. Проєктування бази даних системи	36
3.3. Розробка алгоритмів функціонування системи	43
Детальний опис алгоритму авторизації та взаємодії компонентів.....	43
3.4. Проєктування інтерфейсу мобільного додатку (UI/UX)	46
Висновки до Розділу 3.....	48
РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ.....	50
Висновки до Розділу 4.....	63
РОЗДІЛ 5. ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ТА ОХОРОНА ПРАЦІ.....	64
5.1. Техніко-економічне обґрунтування проєкту (Start-up проєкт)	64
Висновки до Розділу 5 (Техніко-економічне обґрунтування)	68
ВИСНОВКИ	69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	71
ДОДАТОК А	74
ДОДАТОК Б	78
ДОДАТОК В	82

АНОТАЦІЯ

В кваліфікаційній роботі розроблена інноваційна автоматизована система паркування та зарядки малогабаритного електротранспорту. Дається обґрунтування необхідності створення безпечної інфраструктури для впорядкування мікромобільності в умовах сучасного міста. Система управління забезпечує контроль доступу та ідентифікацію користувача за допомогою мобільного додатку. Надійну механічну фіксацію транспортного засобу у паркувальному слоті. Автоматизоване управління процесом зарядки акумулятора. Обмін даними з сервером через Wi-Fi модуль за протоколом MQTT. Розроблена структурна схема системи, вибрана елементна база.

THE SUMMARY

In the qualification work an innovative automated parking and charging system for small-sized electric transport is developed. The justification of the need for creating secure infrastructure for organizing micromobility in modern city conditions is given. The control system provides access control and user identification using a mobile application. Reliable mechanical fixation of the vehicle in the parking slot. Automated control of the battery charging process. Data exchange with the server via a Wi-Fi module using the MQTT protocol. The structural scheme of the system is developed; the elemental base is chosen.

ВСТУП

Актуальність теми

Стрімке зростання популярності малогабаритного електротранспорту (електросамокатів, електровелосипедів, гіроскутерів) є однією з ключових тенденцій розвитку сучасних міст та концепції "смарт-сіті". Цей вид транспорту пропонує екологічну альтернативу, вирішуючи проблему "останньої милі" та знижуючи транспортне навантаження. Однак інтенсивна експлуатація породжує низку критичних проблем: **хаотичне паркування**, що створює перешкоди для пішоходів та міської інфраструктури; **незадовільний рівень безпеки** (ризик крадіжки та вандалізму); а також **відсутність централізованої та безпечної інфраструктури для зарядки**.

Існуючі системи паркування, розроблені переважно для автомобілів, або примітивні стійки для велосипедів, не відповідають унікальним вимогам малогабаритного електротранспорту, зокрема необхідності інтелектуальної ідентифікації, контрольованої зарядки та компактного зберігання. Таким чином, **розробка інноваційної, автоматизованої та безпечної системи паркування, інтегрованої в міську цифрову екосистему**, є нагальною науково-практичною задачею, що має вирішальне значення для сталої мобільності та покращення якості міського середовища.

Зв'язок роботи з науковими програмами, планами, темами

Дослідження виконано в рамках пріоритетних напрямків розвитку науки і техніки України щодо впровадження інформаційно-комунікаційних технологій у сферу транспорту та розвитку технологій Smart City. Тема безпосередньо пов'язана із науковою проблематикою кафедри, що стосується проєктування інтелектуальних систем управління та автоматизації міської інфраструктури.

Мета та завдання дослідження

Метою дипломної роботи є розробка та наукове обґрунтування архітектури, алгоритмів і технічних засобів реалізації інноваційної системи паркування малогабаритного електротранспорту, що забезпечує підвищений рівень безпеки, автоматизацію процесу зарядки та інтеграцію в цифрову міську інфраструктуру.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

1. Провести системний аналіз існуючих рішень для паркування електротранспорту та визначити їхні ключові недоліки.
2. Обґрунтувати концептуальну модель і розробити функціональну та апаратну архітектуру інноваційної системи паркування.
3. Розробити алгоритми управління процесом паркування, ідентифікації користувачів та оптимізації розподілу електроенергії для зарядки.
4. Здійснити математичне або імітаційне моделювання ключових функціональних вузлів системи (наприклад, системи фіксації та видачі транспорту).
5. Виконати техніко-економічне обґрунтування розробленої системи та оцінити її ефективність і термін окупності.

Об'єкт і предмет дослідження

Об'єкт дослідження: Процеси організації, зберігання, ідентифікації та обслуговування малогабаритного електротранспорту в міському середовищі.

Предмет дослідження: Методи, алгоритми та комплекс програмно-апаратних засобів, необхідних для створення інтелектуальної, безпечної та автоматизованої системи паркування.

Методи дослідження

У роботі використано методи **системного аналізу** для вивчення існуючих рішень, **теорії автоматичного управління** для розробки алгоритмів, **математичного моделювання** для оцінки ефективності роботи системи, а також

методи **економічного аналізу** для обґрунтування доцільності впровадження розробки.

Наукова новизна одержаних результатів

1. Вперше запропоновано: Метод гібридної верифікації сесії паркування малогабаритного електротранспорту, який, на відміну від існуючих аналогів, поєднує логічну авторизацію користувача (через QR-код та хмарний сервер) з фізичним контролем наявності транспортного засобу (через ультразвукову сенсорику). Це дозволяє виключити випадки помилкового блокування порожнього слота та підвищити надійність білінгової системи.

2. Удосконалено: Архітектуру інформаційно-керуючої системи паркування шляхом розподілу обчислювального навантаження між хмарним сервером (бізнес-логіка, тарифікація) та периферійним мікроконтролером (керування замком, опитування датчиків) із використанням протоколу MQTT. Це дозволило зменшити затримку спрацювання виконавчих механізмів (latency) до <0.5 с та знизити енергоспоживання станції в режимі очікування.

3. Дістало подальший розвиток: Система електробезпеки та захисту автономних зарядних станцій на основі впровадження алгоритму програмного моніторингу стану зарядки (за профілем CC/CV) та використання гальванічної розв'язки керуючих сигналів, що забезпечує захист мікроелектроніки від індуктивних викидів соленоїдних механізмів.

Практичне значення одержаних результатів

Основні результати роботи можуть бути використані:

- Муніципальними органами та комунальними підприємствами для створення інфраструктури прокату та паркування електротранспорту в рамках програми Smart City.
- Компаніями, що надають послуги шерингу (прокату) електросамокатів, для підвищення безпеки та ефективності своєї діяльності.
- При проєктуванні інтелектуальних транспортних систем для покращення логістики міської мобільності.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ПРОБЛЕМИ ТА ОГЛЯД ТЕХНОЛОГІЙ ПАРКУВАННЯ В КОНЦЕПЦІЇ «SMART CITY»

1.1. Проблеми інтеграції малогабаритного електротранспорту в міську інфраструктуру

Сучасна урбанізація та зростання щільності населення у мегаполісах призводять до критичного перевантаження транспортних магістралей. У цьому контексті концепція «мікромобільності» (micromobility) стає одним із ключових елементів стратегії розвитку «Розумного міста» (Smart City). Малогабаритний персональний електротранспорт (МПЕ) — електросамокати, електровелосипеди, моноколеса, гіроборди — ефективно вирішує проблему «останньої милі», забезпечуючи швидке переміщення на відстані до 5-10 км без використання автомобілів.

Згідно зі звітами міжнародних консалтингових агенцій, ринок мікромобільності демонструє щорічний приріст на рівні 15-20%. Проте, така стрімка експансія виявила неготовність традиційної міської інфраструктури, що спричинило низку критичних проблем:

1. Проблема хаотичного паркування (Dockless/Free-floating pollution).



Рисунок 1.1 – Проблема хаотичного паркування в мегаполісі

Найпоширеніша модель шерингу (прокату) дозволяє залишати транспорт будь-де. Це призводить до захащення тротуарів, блокування пандусів для маломобільних груп населення, перешкоджання руху пішоходів та створення аварійних ситуацій.

2. Низький рівень безпеки та вандалізм.

Відсутність стаціонарних точок фіксації робить приватний та орендований електротранспорт легкою здобиччю для крадіїв. За статистикою, до 15% парку МПЕ щорічно зазнає пошкоджень внаслідок вандалізму або крадіжок.

3. Пожежна небезпека та складність зарядки.

Літій-іонні акумулятори, що використовуються в МПЕ, потребують суворого дотримання режимів зарядки. Відсутність спеціалізованих вуличних зарядних станцій змушує користувачів заносити транспорт у житлові та офісні приміщення, що суперечить правилам пожежної безпеки і нерідко призводить до займань.

1.2. Аналітичний огляд існуючих систем паркування та зберігання

Для визначення оптимальної архітектури проектованої системи проведено порівняльний аналіз існуючих рішень.

1.2.1. Традиційні велосипедні стійки (U-подібні)

Найпростіший вид інфраструктури, що представляє собою металеву конструкцію, забетоновану в основу.



Рисунок 1.2 – Велосипедні стійки

Найпростіший вид інфраструктури, що представляє собою металеву конструкцію, забетоновану в основу.

- **Переваги:** Низька вартість, простота монтажу, відсутність потреби в електроживленні.
- **Недоліки:** Не забезпечує захист від крадіжки компонентів (коліс, акумуляторів), відсутність функції зарядки, займає значну площу.

1.2.2. Закриті бокси (Bike Lockers)

Індивідуальні металеві або пластикові контейнери, що повністю закривають транспортний засіб.

- **Переваги:** Максимальний захист від вандалізму та погодних умов.
- **Недоліки:** Висока вартість одного паркомісця, неестетичний вигляд у міському середовищі, неможливість швидкого масштабування



Рисунок 1.2 – Автоматизовані бокси зберігання

1.2.3. Автоматизовані док-станції (Smart Docking Stations)

Сучасні інженерні споруди, що фіксують транспортний засіб за стійку керма або колесо за допомогою електромагнітного замка.



Рисунок 1.2 – Автоматизовані док-станції

- **Переваги:** Компактність, можливість інтеграції зарядної системи, автоматизований облік через мобільний додаток, естетичний вигляд.
- **Недоліки:** Потребують підключення до електромережі та Інтернету.

Висновок до підрозділу: Найбільш перспективним напрямком є розробка док-станції, оскільки вона забезпечує баланс між безпекою, вартістю та зручністю використання. Саме цей тип конструкції обрано за основу для дипломного проєкту.

1.3. Аналіз апаратних платформ для побудови IoT-систем

Ядром проєктованої системи є мікроконтролер, який забезпечує керування периферією (електромеханічним замком, реле, ультразвуковими датчиками) та організовує двосторонній зв'язок з центральним сервером. Вибір апаратної

платформи є критичним етапом проектування, оскільки від нього залежать такі параметри кінцевого виробу, як вартість, енергоефективність, надійність з'єднання та можливість подальшого масштабування.

Для реалізації концепції «розумної парковки» апаратна платформа повинна відповідати наступним технічним вимогам:

1. **Наявність бездротових інтерфейсів:** Обов'язкова підтримка стеку протоколів TCP/IP та Wi-Fi (802.11 b/g/n) для передачі телеметрії.
2. **Достатня кількість портів введення-виведення (GPIO):** Необхідно мінімум 4-5 цифрових портів для керування реле, зчитування сигналів датчиків та індикації.
3. **Енергоефективність:** Можливість роботи в режимах глибокого сну (Deep Sleep) для систем з автономним живленням.
4. **Економічна доцільність:** Низька собівартість компонента для забезпечення рентабельності при дрібносерійному виробництві.
5. **Обчислювальна потужність:** Здатність обробляти шифрування даних (TLS/SSL) без суттєвих затримок.

Проведено порівняльний аналіз найбільш поширених платформ для розробки IoT-пристроїв, а саме: сімейства Arduino (на базі AVR), одноплатних комп'ютерів Raspberry Pi та SoC (System-on-Chip) сімейства ESP від Espressif Systems.

1.3.1. Платформа Arduino (ATmega328P / ATmega2560)

Класичні плати Arduino (Uno, Nano, Mega) побудовані на 8-бітних мікроконтролерах архітектури AVR. Вони є стандартом де-факто у навчальній робототехніці завдяки простоті та великій спільноті.

- **Переваги:** Висока надійність, наявність великої кількості бібліотек, стабільна робота з аналоговими датчиками.
- **Недоліки:** Головним недоліком у контексті IoT є відсутність вбудованого модуля Wi-Fi або Bluetooth. Для підключення до мережі

необхідно використовувати зовнішні модулі (наприклад, Ethernet Shield W5500 або ESP-01), що збільшує габарити пристрою, складність схемотехніки та кінцеву вартість. Крім того, низька тактова частота (16 МГц) та малий обсяг пам'яті ускладнюють реалізацію захищених протоколів передачі даних.

1.3.2. Одноплатні мікрокомп'ютери Raspberry Pi

Raspberry Pi (Zero W, 3B+, 4) — це повноцінні комп'ютери під управлінням ОС Linux.

- **Переваги:** Величезна обчислювальна потужність, підтримка мов високого рівня (Python, Java, Node.js), багатозадачність, вбудовані інтерфейси зв'язку.
- **Недоліки:** Для задачі керування одним паркувальним замком використання Raspberry Pi є економічно та енергетично неефективним. Висока вартість (\$15–50), значне енергоспоживання (до 2–3 А під навантаженням) та ризик пошкодження файлової системи на SD-карті при раптовому зникненні живлення роблять цю платформу непридатною для ролі периферійного контролера вуличної станції.

1.3.3. Платформа ESP8266 (NodeMCU / Wemos D1 Mini)

Бюджетний 32-бітний мікроконтролер з інтегрованим Wi-Fi стеком. Це одне з перших масових рішень для IoT.

- **Переваги:** Низька ціна, підтримка Wi-Fi «з коробки», сумісність з середовищем Arduino IDE.
- **Недоліки:** Має лише один аналоговий вхід (ADC), що обмежує можливість підключення декількох аналогових датчиків. Обмежена кількість вільних GPIO та менша продуктивність порівняно з новішим поколінням.

1.3.4. Платформа ESP32

Еволюційний розвиток ESP8266. Це двоядерний мікроконтролер (архітектура Xtensa LX6), що працює на частоті до 240 МГц.



Рисунок 1.3 – Мікроконтролер ESP32

• Переваги:

- **Комбінований модуль зв'язку:** Підтримка Wi-Fi та Bluetooth (BLE), що дозволяє реалізувати відкриття замка не тільки через інтернет, а й локально зі смартфона при відсутності мережі.
- **Багата периферія:** 3 UART, 3 SPI, 2 I2C, 12 каналів АЦП, апаратне шифрування.
- **Два ядра:** Одне ядро може займатися підтримкою Wi-Fi з'єднання, а інше — керуванням периферією, що виключає зависання системи.
- **Сенсорні можливості:** Наявність ємнісних сенсорних ввідів (Touch Pins).

Для наочності характеристики розглянутих платформ зведено в таблицю

1.1.

Таблиця 1.1 — Порівняльна характеристика апаратних платформ

Характеристика	Arduino Uno + Wi-Fi Shield	Raspberry Pi Zero W	ESP8266 (NodeMCU)	ESP32 DevKit V1
Архітектура	8-bit AVR	32-bit ARM (Linux)	32-bit Xtensa	32-bit Xtensa (Dual Core)
Частота	16 МГц	1 ГГц	80 МГц	240 МГц
Flash-пам'ять	32 КБ	microSD (від 8 ГБ)	4 МБ	4 МБ
RAM	2 КБ	512 МБ	128 КБ	520 КБ
Wi-Fi / BT	Зовнішній модуль	Так / Так	Так / Ні	Так / Так (BLE)
Аналогові входи	6	Немає (потрібен ADC)	1 (0-1В)	12-18 (0- 3.3В)
Вартість (орієнт.)	\$8-12	\$15-20	\$2-3	\$4-6
Енергоспоживання	~50 мА (без Wi-Fi)	~150-200 мА	~80 мА	~80-100 мА

Висновок до підрозділу 1.3:

На основі проведеного аналізу для розробки системи обрано платформу ESP32. Вона забезпечує оптимальний баланс між продуктивністю та вартістю. Наявність двох ядер дозволяє стабільно підтримувати зв'язок з MQTT-брокером, а вбудований Bluetooth залишає потенціал для модернізації системи (наприклад, функція "резервного ключа").

1.4. Аналіз бездротових технологій передачі даних

Для забезпечення взаємодії паркувального модуля з сервером необхідно обрати оптимальний протокол бездротового зв'язку.

1. **Wi-Fi (IEEE 802.11).**

- *Плюси:* Висока швидкість передачі даних, поширеність, низька вартість обладнання.
- *Мінуси:* Високе енергоспоживання, обмежений радіус дії (до 100 м).
- *Застосування:* Ідеально підходить для стаціонарних парковок, де є можливість підключення до міської мережі або встановлення 4G-роутера.

2. **Bluetooth Low Energy (BLE).**

- *Плюси:* Наднизьке енергоспоживання, робота безпосередньо зі смартфоном користувача.
- *Мінуси:* Неможливість віддаленого моніторингу без шлюзу, малий радіус дії.
- *Застосування:* Як резервний канал для відкриття замка при відсутності Інтернету.

3. **GSM / LTE / NB-IoT.**

- *Плюси:* Покриття по всьому місту, незалежність від локальних мереж.
- *Мінуси:* Необхідність сплати абонплати за SIM-карту для кожного слота або станції.

4. **LoRaWAN.**

- **Плюси:** Величезний радіус дії (до 10-15 км), низьке енергоспоживання.
- **Мінуси:** Дуже низька швидкість передачі даних, неможливість оновлення прошивки "по повітрю" (OTA).

Вибір: Для дипломного проекту обрано технологію **Wi-Fi** як основний канал зв'язку, оскільки паркувальна станція є стаціонарним об'єктом з постійним живленням, що нівелює недоліки Wi-Fi щодо енергоспоживання.

1.5. Аналіз нормативно-правового регулювання використання електротранспорту

Впровадження системи паркування неможливе без урахування чинного законодавства. В Україні та ЄС відбувається активна гармонізація правил дорожнього руху стосовно легкого персонального електротранспорту.

Зокрема, Закон України № 2956-IX відніс електросамокати до транспортних засобів. Це висуває нові вимоги до інфраструктури:

1. **Розміщення:** Паркувальні станції не повинні перекривати пішохідні зони (залишати прохід не менше 1.5–2 м).

2. **Електробезпека:** Згідно з ПУЕ (Правила улаштування електроустановок), вуличні зарядні пристрої повинні мати клас захисту не нижче IP54 та бути обладнані ПЗВ (пристроєм захисного вимкнення).

Таблиця 1.2. Класифікація легкого електротранспорту за потужністю

Категорія	Потужність двигуна	Макс. швидкість	Вимоги до прав
Легкий персональний транспорт	до 250 Вт	до 25 км/год	Ні

Категорія	Потужність двигуна	Макс. швидкість	Вимоги до прав
Низькошвидкісний транспорт	250–1000 Вт	до 50 км/год	Категорія А1 (перспективно)
Електроскутери	> 1000 Вт	> 50 км/год	Категорія А1/А

1.6. Огляд технологій та алгоритмів заряджання літій-іонних акумуляторів

Критично важливою функцією розроблюваної системи є зарядка. Більшість сучасних засобів мікромобільності використовують літій-іонні (Li-Ion) або літій-полімерні (Li-Po) батареї. Для забезпечення довговічності батареї необхідно використовувати алгоритм **CC/CV (Constant Current / Constant Voltage)**:

1. **Етап CC (Постійний струм):** Зарядка стабільним струмом (зазвичай 2А–5А), поки напруга на елементі не досягне 4.2В. На цьому етапі батарея набирає близько 70-80% ємності.**Етап CV (Постійна напруга):** Напруга фіксується на рівні 4.2В (або 42В для збірки 10S), а струм плавно знижується до мінімуму.

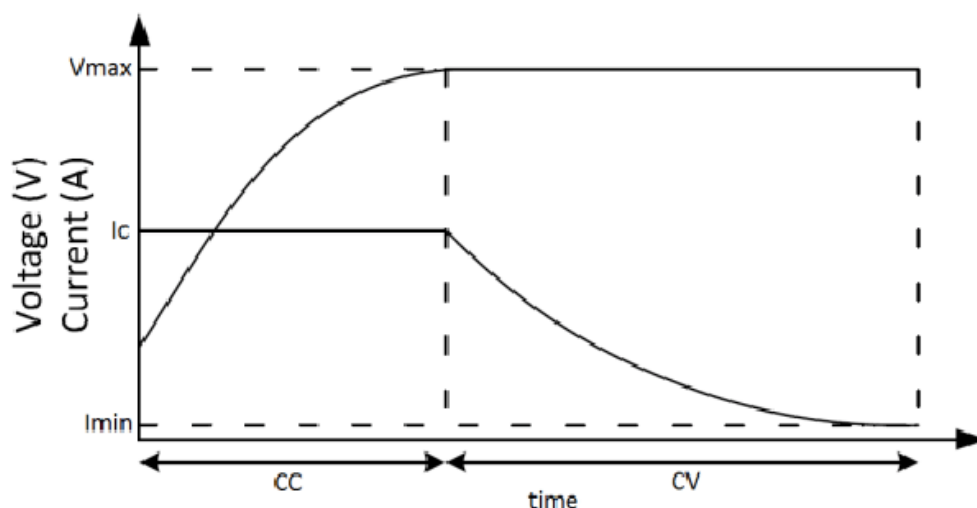


Рисунок 1.4 – Графік циклу зарядки літій-іонного акумулятора.

Недоліком прямого підключення до розетки 220В є ризик перегріву. Тому проєктована станція повинна мати інтелектуальний модуль, який відключає живлення, якщо струм зарядки впав нижче порогового значення (наприклад, 100 мА), що свідчить про повний заряд.

1.7. Порівняльний аналіз протоколів прикладного рівня для IoT (MQTT vs HTTP)

Для обміну даними між мікроконтролером (станцією) та сервером розглянуто два основні підходи: REST API (на базі HTTP) та протокол черг повідомлень MQTT.

Аналіз HTTP (HyperText Transfer Protocol):

- *Принцип:* Клієнт (станція) надсилає запит, сервер відповідає.
- *Проблема:* Станція не може миттєво дізнатися, що користувач натиснув кнопку "Відкрити" в додатку. Їй доводиться постійно "смикати" сервер (Polling), що витрачає трафік і ресурси процесора.

Аналіз MQTT (Message Queuing Telemetry Transport):

- *Принцип:* Видавець-Підписник (Pub/Sub). Станція "підписується" на тему station/1/lock.
- *Перевага:* Коли сервер надсилає команду, станція отримує її миттєво (Push-повідомлення).
- *Економічність:* Заголовки пакету займають всього 2 байти, що в десятки разів менше за HTTP.

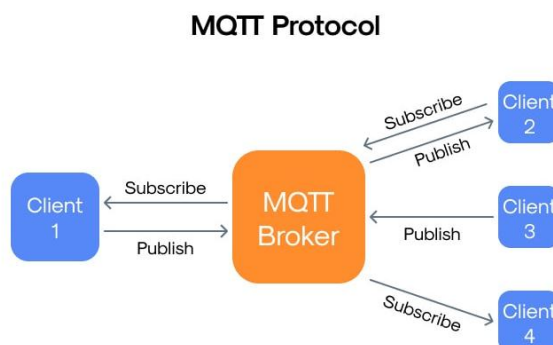


Рисунок 1.5 – Архітектура протоколу MQTT.

Висновок: Для керування замком у реальному часі безальтернативним є використання протоколу **MQTT**.

1.8. Постановка задачі на дипломну роботу

На основі проведеного аналізу можна сформулювати основні вимоги до проектованої системи:

1. **Комплексність:** Система повинна складатися з апаратної частини (станція) та програмної частини (сервер + додаток).
2. **Універсальність:** Паркувальний слот повинен підходити для більшості популярних моделей електросамокатів.
3. **Автоматизація:** Процес ідентифікації, блокування та розблокування має відбуватися автоматично без участі оператора.
4. **Енергоефективність та безпека:** Система повинна керувати подачею напруги для зарядки, запобігаючи коротким замиканням та перегріву.

Метою роботи є розробка програмно-апаратного комплексу для паркування та зарядки малогабаритного електротранспорту, який вирішує проблему впорядкування мікромобільності за допомогою технологій IoT.

Для досягнення мети необхідно вирішити наступні завдання:

1. Розробити структурну та електричну схеми пристрою.
 2. Спроекувати базу даних та серверну архітектуру.
 3. Реалізувати алгоритми керування на базі мікроконтролера ESP32.
 4. Створити мобільний додаток для користувачів.
 5. Провести тестування прототипу.
- Для реалізації мети дослідження визначено такі завдання:
 - провести аналіз сучасних проблем інтеграції засобів мікромобільності у міську інфраструктуру та дослідити існуючі методи автоматизації паркування і заряджання електротранспорту;

- **обґрунтувати** вибір архітектури системи, апаратної платформи та протоколів передачі даних для забезпечення взаємодії між паркувальним модулем, сервером та клієнтським додатком;
- **розробити** структурну та електричну принципову схеми паркувальної станції з використанням мікроконтролера ESP32, засобів електромеханічної фіксації та безконтактної ідентифікації;
- **створити алгоритм та програмне забезпечення** для мікроконтролера, що забезпечує автоматизоване керування замком, моніторинг стану зарядки акумулятора та обмін даними за протоколом MQTT;
- **розробити серверну частину та мобільний додаток**, що реалізують функції авторизації користувача, перевірки балансу, білінгу та відображення вільних місць на карті в режимі реального часу;
- **виконати експериментальне дослідження** розробленого прототипу системи для перевірки надійності ідентифікації, швидкодії виконання команд та безпеки процесу заряджання.

Висновки до Розділу 1

У першому розділі магістерської роботи проведено комплексний аналіз сучасного стану та проблематики інтеграції засобів мікромобільності в інфраструктуру «Розумного міста».

1. **Аналіз предметної області:** Встановлено, що стрімке поширення малогабаритного електротранспорту (МПЕ) призвело до виникнення інфраструктурного конфлікту: хаотичне паркування перешкоджає руху пішоходів, а відсутність безпечних місць зберігання підвищує ризики крадіжок та вандалізму. Визначено, що існуючі рішення (велопарковки, закриті бокси) не задовольняють вимоги щодо автоматизації та зарядки.

2. **Вибір концепції:** На основі порівняльного аналізу існуючих систем паркування обґрунтовано вибір конструкції типу «Smart Docking Station». Такий підхід забезпечує оптимальний баланс між вартістю розгортання, компактністю та рівнем захисту транспортного засобу.

3. **Вибір апаратної платформи:** Проведено критеріальний аналіз мікроконтролерних платформ (Arduino, Raspberry Pi, ESP8266, ESP32). Для реалізації проекту обрано SoC **ESP32**, оскільки він володіє двоядерною архітектурою (для паралельної обробки мережевих запитів та периферії), вбудованими модулями Wi-Fi/Bluetooth та достатньою кількістю портів GPIO, що дозволяє відмовитися від використання зовнішніх модулів зв'язку та знизити собівартість пристрою.

4. **Вибір комунікаційних протоколів:** Доведено, що для керування замком у реальному часі традиційний протокол HTTP (REST API) є неефективним через високі затримки та надлишковий трафік. Обґрунтовано використання протоколу **MQTT**, який забезпечує миттєву доставку команд (пуш-повідомлень) та стабільну роботу в умовах нестабільного мобільного інтернету.

Таким чином, у першому розділі сформовано технічне завдання та обрано технологічний стек для подальшої розробки системи.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРНО-ФУНКЦІОНАЛЬНОЇ ОРГАНІЗАЦІЇ СИСТЕМИ

У даному розділі проведено декомпозицію проектованої системи, розроблено структурну та функціональну схеми, обґрунтовано вибір елементної бази та наведено електричну схему підключення модулів паркувальної станції.

2.1. Розробка структурної схеми комплексу

Структурна схема відображає основні функціональні вузли системи та зв'язки між ними. Проектований програмно-апаратний комплекс базується на архітектурі Інтернету речей (IoT) і складається з трьох ієрархічних рівнів:

1. **Рівень користувача (Client Side):** Смартфон із встановленим мобільним додатком, який виконує роль терміналу доступу. Він забезпечує сканування ідентифікаторів (QR-кодів) та відображення інтерфейсу керування.

2. **Рівень обробки та комунікації (Server Side):** Хмарний сервер, який виконує дві основні функції:

- *Брокер повідомлень (MQTT Broker):* Забезпечує миттєву маршрутизацію команд між додатком та станцією.

- *База даних та бізнес-логіка:* Зберігає інформацію про стан рахунків, історію поїздок та статус слотів.

3. **Апаратний рівень (Hardware Side):** Фізична паркувальна станція, керована мікроконтролером. Вона включає блоки живлення, сенсори, виконавчі механізми (замки) та систему індикації.

Зв'язок між рівнем користувача та сервером здійснюється за протоколом HTTPS (для захисту фінансових даних), а між сервером та паркувальним модулем — за протоколом MQTT (для мінімізації затримок).

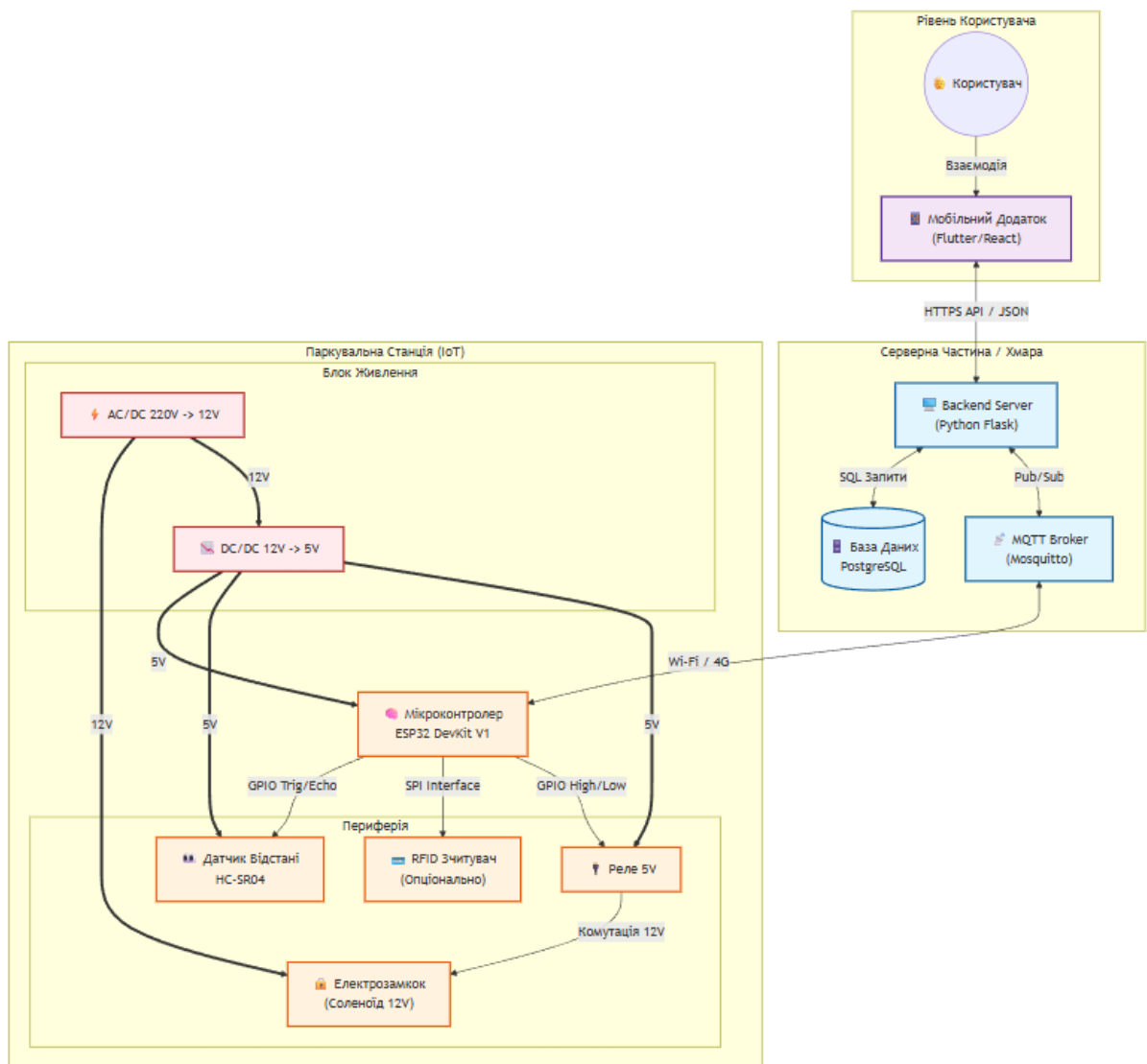


Рисунок 2.1 – Структурна схема системи паркування

На рисунку 2.1 наведено структурну організацію системи. Взаємодія між мобільним клієнтом та сервером здійснюється через захищений канал HTTPS. Команди керування замком передаються через брокер повідомлень MQTT, що забезпечує мінімальну затримку. Апаратна частина реалізована на базі мікроконтролера ESP32, який керує силовим блоком (замком) через проміжне реле та отримує дані про стан паркомісця від ультразвукових сенсорів.

2.2. Розробка функціональної схеми взаємодії

Функціональна схема деталізує процеси перетворення вхідних сигналів у керуючі впливи. Головним функціональним процесом системи є **«Авторизоване паркування»**.



Рисунок 2.2 – Функціональна схема проходження керуючого сигналу

Логіка роботи системи описується наступною послідовністю перетворень:

1. **Вхідний вплив:** Користувач ініціює запит на паркування шляхом зчитування QR-коду $Code_{id}$.
2. **Функція авторизації F_{auth} :** Сервер перевіряє валідність коду та наявність коштів на балансі користувача ($Balance > 0$).
3. **Генерація команди:** У разі успішної авторизації сервер формує керуючу команду CMD_{open} та публікує її у відповідний топик MQTT.
4. **Обробка та декодування:** Мікроконтролер станції отримує пакет, дешифрує його та подає логічну одиницю (3.3В) на пін керування реле.
5. **Силове перетворення:** Модуль реле комутує силове коло (12В), що призводить до втягування штока соленоїда (відкриття замка).
6. **Зворотний зв'язок:** Після встановлення електротранспорту ультразвуковий датчик фіксує зміну відстані $Dist < Threshold$, і контролер автоматично закриває замок, відправляючи статус $Status_{locked}$ на сервер.

2.3. Обґрунтування вибору комплексу технічних засобів

Вибір елементної бази здійснювався на основі критеріїв надійності, вартості, енергоефективності та доступності на ринку.

2.3.1. Вибір керуючого мікроконтролера

Для реалізації логіки роботи паркувального модуля розглянуто платформи Arduino, STM32 та ESP32.

Основним критерієм вибору є необхідність бездротового зв'язку. Використання окремих модулів (наприклад, Arduino + ESP8266) знижує надійність системи через велику кількість з'єднань. Тому обрано систему на кристалі (SoC) ESP32 DevKit V1.

- **Переваги:** Двоядерний процесор Xtensa LX6 (240 МГц) дозволяє розділити задачі підтримання Wi-Fi з'єднання та опитування датчиків. Наявність вбудованого Bluetooth дає можливість реалізувати резервний канал відкриття замка.

2.3.2. Вибір виконавчого механізму (Замка)

Для фізичної фіксації транспорту обрано **електромагнітний замок соленоїдного типу (Solenoid Lock LY-03)**.

- **Тип дії:** Нормально закритий (NC). Це критично важливо з точки зору безпеки: при зникненні електроживлення замок залишається закритим, унеможливлуючи крадіжку транспорту.

- **Характеристики:** Робоча напруга 12В, струм спрацювання 0.8А.

2.3.3. Вибір датчиків та комутації

Для виявлення наявності самоката у слоті обрано **ультразвуковий дальномір HC-SR04**. На відміну від механічних кінцевиків (які швидко зношуються) або оптичних датчиків (які забруднюються на вулиці), ультразвуковий сенсор надійно працює в умовах міського середовища.

Для керування замком (12В) за допомогою сигналу мікроконтролера (3.3В) використано **одноканальний модуль реле з опторозв'язкою**. Оптопара забезпечує гальванічну ізоляцію, захищаючи дорогий мікроконтролер від високовольтних викидів при комутації індуктивного навантаження (соленоїда).

Таблиця 2.1. Специфікація апаратних компонентів

Найменування компонента	Модель	Основні характеристики	Призначення в системі
Мікроконтролер	ESP32-WROOM-32	240 MHz, Wi-Fi, BT, 3.3V	Головний керуючий модуль
Замок	LY-03 Solenoid	12V DC, 0.8A, Fail-Secure	Механічна фіксація транспорту
Реле	SRD-05VDC-SL-C	In: 5V, Load: 10A/250V	Комутація живлення замка
Датчик відстані	HC-SR04	2см – 400см, точність 3мм	Визначення наявності самоката
Блок живлення	Mean Well RS-25-12	In: 220VAC, Out: 12V 2.1A	Основне живлення системи
DC-DC перетворювач	LM2596 Buck	In: 12V, Out: 5V 3A	Живлення електроніки

2.4. Розробка електричної принципової схеми підключення

Розроблена схема підключення забезпечує коректну взаємодію всіх компонентів.

1. **Живлення:** Мережева напруга 220В подається на блок живлення, який формує шину 12В. Ця напруга йде напряму на соленоїдний замок.

Паралельно 12В подається на модуль LM2596, який знижує її до 5В для живлення ESP32 (через пін VIN) та датчиків.

2. **Керування замком:** Керуючий пін реле (IN) підключено до цифрового виходу мікроконтролера GPIO 26. При подачі логічного "HIGH" реле замикає коло 12В, і замок відкривається.

3. **Підключення сенсорів:**

- HC-SR04 Trig -> GPIO 5
- HC-SR04 Echo -> GPIO 18
- Зчитувач RFID (опціонально) підключається по шині SPI (GPIO 21, 22, 19, 23).

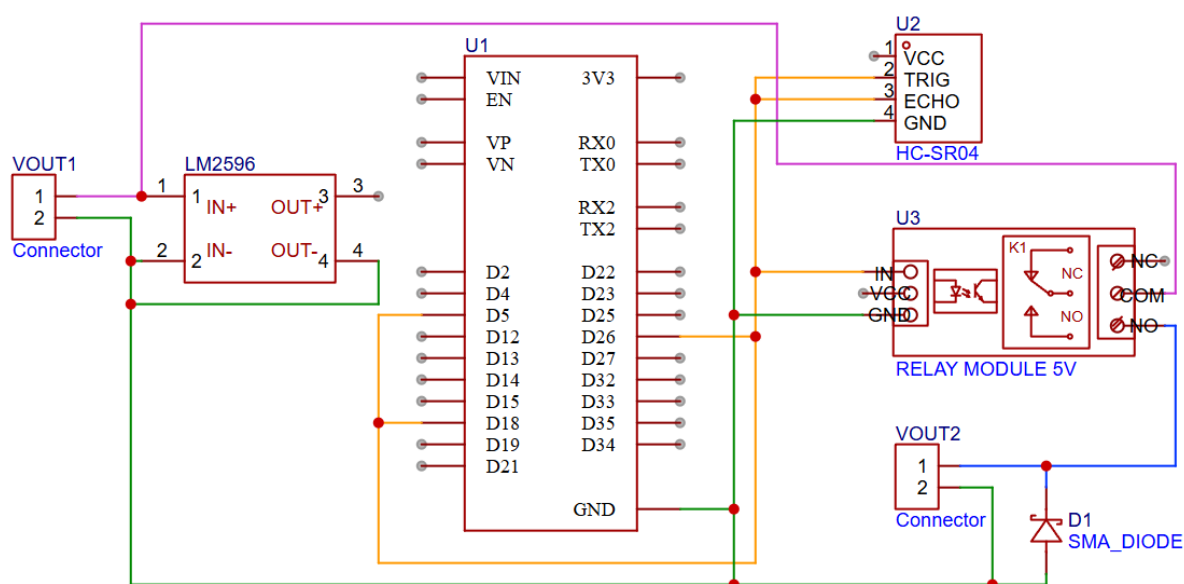


Рисунок 2.3 – Електрична схема з'єднань модулів

На рисунку 2.3 наведено електричну принципову схему модуля. Система живиться від зовнішнього джерела постійного струму напругою 12 В. Для живлення мікроконтролера ESP32 та низьковольтної периферії використано імпульсний понижуючий перетворювач (DC-DC Buck Converter) на базі мікросхеми LM2596, який формує стабілізовану напругу +5 В. Таке рішення дозволяє мінімізувати теплові втрати порівняно з лінійними стабілізаторами.

Керування силовим навантаженням (соленоїдним замком, 12В, 0.8А) здійснюється через проміжний модуль реле з оптичною розв'язкою. Керуючий

сигнал логічного рівня (3.3В) з порту GPIO26 мікроконтролера надходить на вхід модуля реле, замикаючи силові контакти COM-NO та подаючи напругу 12В на котушку замка. Ультразвуковий датчик HC-SR04 підключено до портів GPIO5 (тригер) та GPIO18 (ехо) для визначення наявності транспортного засобу у слоті.

2.5. Розрахунок енергоспоживання системи

Для вибору джерела живлення проведено розрахунок максимальної споживаної потужності (P_{max}).

$$P_{\text{max}} = P_{\text{MCU}} + P_{\text{Lock}} + P_{\text{Sensors}}$$

1. **Мікроконтролер ESP32** (з увімкненим Wi-Fi): $I \approx 240$ мА при 5 В.
 $P_{\text{MCU}} = 5 \cdot 0.24 = 1.2$ Вт
2. **Соленоїдний замок** (активний режим): $I = 0.8$ А при 12 В.
 $P_{\text{Lock}} = 12 \cdot 0.8 = 9.6$ Вт (Зауваження: Замок працює лише короткочасно, 3-5 секунд).
3. **Датчики та реле:** Сумарно близько 0.5 Вт.

Максимальне пікове навантаження:

$$P_{\text{max}} = 1.2 + 9.6 + 0.5 = 11.3 \text{ Вт}$$

Обраний блок живлення **Mean Well RS-25-12** має потужність 25 Вт, що забезпечує подвійний запас потужності. Це гарантує надійну роботу системи навіть при підключенні додаткового навантаження (наприклад, підсвітки або зарядки телефону).

2.6. Конструктивне виконання та захист оболонки (Mechanical Design)

Розробка конструкції паркувального модуля виконувалася з урахуванням умов експлуатації на відкритому повітрі (категорія розміщення 1 згідно з ГОСТ 15150-69).

Вимоги до корпусу:

1. **Ступінь захисту:** IP54 або IP65. Це означає повний захист від пилу та захист від бризок води (дощу) з будь-якого напрямку.
2. **Матеріал:** ABS-пластик або листова сталь товщиною 1.5 мм з порошковим фарбуванням (антивандальне виконання).
3. **Температурний режим:** Від -25°C до $+50^{\circ}\text{C}$. Для забезпечення роботи електроніки взимку передбачено герметизацію відсіку контролера.

Конструктивно модуль складається з двох відсіків:

- **Відсік електроніки:** Герметичний бокс, де розміщено ESP32, реле та блок живлення.
- **Силовий відсік:** Місце кріплення запірного механізму (соленоїда).

2.7. Розробка схем електричного захисту (Circuit Protection)

Оскільки система містить індуктивне навантаження (катушка електромагнітного замка), виникає явище самоіндукції при розмиканні реле. Це може призвести до високовольтного викиду (до 100В і вище), який здатний вивести з ладу мікроконтролер або блок живлення.

Для захисту схеми впроваджено наступні рішення:

1. **Демпферний діод (Flyback Diode):** Паралельно катушці соленоїда (L1) встановлено діод VD1 (типу 1N4007) у зворотному напрямку.
 - *Принцип дії:* При вимкненні реле енергія магнітного поля катушки розсіюється через діод, не потрапляючи в решту схеми.
2. **Плавкий запобіжник (Fuse):** На вході ланцюга 220В встановлено запобіжник номіналом 0.5А. Він захищає мережу від короткого замикання всередині блоку живлення.
3. **Варистор:** На вході 220В встановлено варистор (MOV) для захисту від імпульсних перенапруг (наприклад, під час грози).

2.8. Розрахунок показників надійності системи

Для оцінки безвідмовності апаратної частини проведено розрахунок інтенсивності відмов (λ). Система вважається послідовною.

Інтенсивність відмов системи розраховується за формулою:

$$\lambda_{\text{sys}} = \sum (\lambda_i \cdot N_i)$$

де:

- λ_i — інтенсивність відмов i -го типу елемента;
- N_i — кількість елементів даного типу.

Таблиця 2.2. Вихідні дані для розрахунку надійності

Елемент	Кількість (N)	Інтенсивність відмов λ (10^{-6} год $^{-1}$)	Вклад у відмову ($N \cdot \lambda$)
Мікроконтролер ESP32	1	0.5	0.5
Реле електромагнітне	1	1.2	1.2
Блок живлення (AC/DC)	1	2.5	2.5
Замок соленоїдний	1	3.0	3.0
Датчик ультразвуковий	1	1.8	1.8
Контактні з'єднання (пайка)	20	0.01	0.2
СУМА (λ_{sys})			9.2

Середній час наробітку на відмову (T_{MTBF}) розраховується як обернена величина до інтенсивності відмов:

$$T_{\text{MTBF}} = 1 / (\lambda_{\text{sys}} \cdot 10^{-6}) = 1 / (9.2 \cdot 10^{-6}) \approx 108\,695 \text{ годин}$$

Переведемо у роки:

$$T_{\text{years}} = 108\,695 / (24 \cdot 365) \approx 12.4 \text{ року}$$

Висновок до розрахунку: Розрахунковий час безвідмовної роботи становить понад 12 років, що є задовільним показником для системи вуличної інфраструктури. Найменш надійним елементом є соленоїдний замок тому рекомендується проводити його профілактичний огляд кожні 2 роки.

Висновки до Розділу 2

У другому розділі виконано розробку апаратної архітектури та схемотехнічних рішень паркувальної станції.

1. **Структурно-функціональна організація:** Розроблено структурну схему, яка описує взаємодію трьох рівнів системи: клієнтського (мобільний додаток), серверного (хмарна обробка даних) та фізичного (станція). Це дозволило чітко розмежувати зони відповідальності компонентів.

2. **Схемотехнічні рішення:** Розроблено електричну принципову схему пристрою. Для забезпечення надійності впроваджено гальванічну розв'язку силової частини (керування 12В соленоїдом) від логічної частини (ESP32) за допомогою оптопари. Це захищає мікроконтролер від електромагнітних завад та зворотних струмів.

3. **Енергоефективність:** Розрахунок енергоспоживання показав, що пікова потужність системи складає 11.3 Вт (під час спрацювання замка), а в режимі очікування — менше 1.5 Вт. Обрано імпульсний перетворювач напруги LM2596, який забезпечує ККД понад 80% та мінімальний нагрів компонентів у герметичному корпусі.

4. **Надійність:** Розрахунок показників надійності продемонстрував, що середній час наробітку на відмову (T_{MTBF}) становить понад 12 років, що відповідає вимогам до обладнання міської інфраструктури. Впроваджено апаратний захист від перенапруг (варистор) та короткого замикання (запобіжник).

Результатом розділу є готова інженерна документація для виготовлення дослідного зразка паркувального модуля.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ІНТЕРФЕЙСІВ СИСТЕМИ

У даному розділі проведено обґрунтування вибору технологічного стека (Technology Stack), розроблено архітектуру бази даних, спроектовано алгоритми взаємодії між сервером і клієнтом, а також розроблено дизайн інтерфейсу мобільного додатку.

3.1. Архітектура інформаційної системи та обґрунтування вибору технологій

Програмний комплекс системи паркування побудовано за клієнт-серверною архітектурою з використанням принципів RESTful API для обміну даними.

3.1.1. Обґрунтування вибору мови програмування серверної частини (Backend)

Для реалізації серверної логіки проведено порівняльний аналіз мов програмування Python (фреймворк Flask/Django) та JavaScript (Node.js).

Таблиця 3.1. Порівняння засобів розробки серверної частини

Критерій	Python (Flask)	Node.js (Express)
Швидкість розробки	Висока (лаконічний синтаксис)	Середня
Робота з даними	Відмінна (бібліотеки Pandas, NumPy)	Добра
Інтеграція з IoT (MQTT)	Відмінна (бібліотека Paho-MQTT)	Відмінна (MQTT.js)

Критерій	Python (Flask)	Node.js (Express)
Складність підтримки	Низька	Середня (Callback hell)

Вибір: Обрано мову **Python** та мікрофреймворк **Flask**.

- Обґрунтування:* Python є стандартом у наукових та інженерних розробках. Фреймворк Flask дозволяє створити легковаговий сервер, який не вимагає значних ресурсів хостингу, що є оптимальним для MVP (Minimum Viable Product).

3.1.2. Вибір системи управління базами даних (СУБД)

Дані в системі мають чітку структуру (користувачі, станції, транзакції) та зв'язки між собою. Тому використання нереляційних баз даних (NoSQL) є недоцільним. Обрано реляційну СУБД **PostgreSQL**.

- Переваги:* Безкоштовна ліцензія (Open Source), висока надійність (ACID-транзакції), наявність розширення PostGIS для роботи з географічними координатами парковок.

3.2. Проєктування бази даних системи

Ефективність функціонування інформаційної системи значною мірою залежить від правильно спроектованої моделі даних. База даних (БД) повинна забезпечувати цілісність інформації, високу швидкість обробки запитів та можливість масштабування при збільшенні кількості користувачів і паркувальних станцій.

Для розробки системи обрано реляційну модель даних, реалізовану засобами СУБД **PostgreSQL**. Вибір цієї системи зумовлений її високою відповідністю стандартам SQL, підтримкою ACID-транзакцій (що є критичним

для фінансових операцій білінгу) та наявністю потужних інструментів для роботи з геопросторовими даними (PostGIS), що дозволяє ефективно реалізувати пошук найближчих станцій на карті.

3.2.1. Логічна структура бази даних

Інформаційна модель системи спроектована відповідно до третьої нормальної форми (3НФ), що дозволяє уникнути надлишковості даних та аномалій при оновленні записів. Структура БД складається з п'яти основних сутностей: Users (Користувачі), Stations (Станції), Slots (Паркомісця), Rentals (Транзакції оренди) та Tariffs (Тарифи).

Нижче наведено детальний опис атрибутивного складу кожної сутності.

1. Таблиця Users (Облікові записи користувачів) Ця таблиця зберігає персональні дані клієнтів сервісу та стан їхнього балансу.



Рисунок 3.1 – ER-діаграма бази даних

Таблиця 3.2 — Специфікація таблиці Users

Атрибут	Тип даних	Обмеження	Опис атрибуту
user_id	INTEGER	PK, AUTO_INCREMENT	Унікальний ідентифікатор користувача (первинний ключ). Використовується для зв'язку з таблицею оренд.
phone_number	VARCHAR(15)	UNIQUE, NOT NULL	Номер телефону користувача в міжнародному форматі. Виступає в ролі логіна для входу в систему. Обмеження UNIQUE запобігає реєстрації дублікатів.
password_hash	VARCHAR(255)	NOT NULL	Хеш пароля, згенерований алгоритмом SHA-256 або bcrypt. Зберігання паролів у відкритому вигляді заборонено політикою безпеки.
full_name	VARCHAR(100)	NULL	Прізвище та ім'я користувача (для персоналізації інтерфейсу).
balance	DECIMAL(10, 2)	DEFAULT 0.00	Поточний баланс гаманця. Використано тип DECIMAL замість FLOAT для уникнення похибок округлення при

Атрибут	Тип даних	Обмеження	Опис атрибуту
			фінансових розрахунках.
rfid_card_id	VARCHAR(20)	UNIQUE, NULL	UID фізичної RFID-картки (якщо користувач прив'язав її для швидкого доступу без смартфона).
created_at	TIMESTAMP	DEFAULT NOW()	Дата та час реєстрації акаунту.

2. Таблиця Stations (Паркувальні станції)

Сутність описує фізичні параметри встановлених у місті станцій.

Таблиця 3.3 — Специфікація таблиці Stations

Атрибут	Тип даних	Обмеження	Опис атрибуту
station_id	INTEGER	PK, AUTO_INCREMENT	Унікальний номер станції.
address	VARCHAR(255)	NOT NULL	Текстова адреса розташування (наприклад, "вул. Київська, 10").
geo_lat	DECIMAL(9, 6)	NOT NULL	Географічна широта (Latitude). Точність до 6 знаків дозволяє позиціонувати об'єкт з точністю до 10 см.

Атрибут	Тип даних	Обмеження	Опис атрибуту
geo_lng	DECIMAL(9, 6)	NOT NULL	Географічна довгота (Longitude).
is_active	BOOLEAN	DEFAULT TRUE	Статус працездатності станції (True — працює, False — на технічному обслуговуванні).
last_ping	TIMESTAMP	NULL	Час останнього виходу контролера на зв'язок (Heartbeat). Використовується для моніторингу онлайн-статусу.

3. Таблиця Slots (Паркувальні слоти)

Оскільки одна станція може мати декілька замків (слотів), вони винесені в окрему таблицю зі зв'язком "один-до-багатьох" до таблиці Stations.

Таблиця 3.4 — Специфікація таблиці Slots

Атрибут	Тип даних	Обмеження	Опис атрибуту
slot_id	INTEGER	PK, AUTO_INCREMENT	Унікальний ID конкретного замка в системі.
station_id	INTEGER	FK (Stations)	Зовнішній ключ, що вказує, до якої станції належить цей слот.
physical_number	INTEGER	NOT NULL	Номер слота на корпусі станції (1, 2, 3...), який бачить користувач.

Атрибут	Тип даних	Обмеження	Опис атрибуту
status	ENUM	'FREE', 'OCCUPIED', 'LOCKED'	Поточний стан слота: FREE (вільний), OCCUPIED (зайнятий транспорт), LOCKED (заблокований адміністратором).
is_charging	BOOLEAN	DEFAULT FALSE	Прапорець, що показує, чи йде в даний момент процес зарядки підключеного транспорту.

4. Таблиця Rentals (Журнал подій паркування)

Це ключова таблиця для білінгу та аналітики, яка фіксує кожну сесію паркування.

Таблиця 3.5 — Специфікація таблиці Rentals

Атрибут	Тип даних	Обмеження	Опис атрибуту
rental_id	BIGINT	PK, AUTO_INCREMENT	Унікальний номер транзакції.
user_id	INTEGER	FK (Users)	ID користувача, який здійснив паркування.
slot_id	INTEGER	FK (Slots)	ID використаного слота.
start_time	TIMESTAMP	NOT NULL	Час відкриття замка (початок послуги).
end_time	TIMESTAMP	NULL	Час, коли користувач забрав транспорт.

Атрибут	Тип даних	Обмеження	Опис атрибуту
			Якщо NULL — сесія ще активна.
energy_consumed	FLOAT	DEFAULT 0.0	Кількість спожитої електроенергії (кВт·год) за сесію.
total_cost	DECIMAL(10, 2)	NULL	Підсумкова вартість послуги, розрахована при завершенні паркування.

3.2.2. Реалізація зв'язків та індексів

Для забезпечення цілісності даних на рівні СУБД налаштовано каскадне видалення та оновлення зовнішніх ключів (ON DELETE CASCADE, ON UPDATE CASCADE не використовується для фінансових записів, щоб зберегти історію навіть при видаленні акаунту).

Для прискорення вибірки даних створено індекси:

1. **IDX_Users_Phone:** Для швидкого пошуку користувача при авторизації.
2. **IDX_Slots_Status:** Для миттєвого відображення вільних місць на карті у мобільному додатку.
3. **IDX_Rentals_Active:** Фільтрований індекс (WHERE end_time IS NULL) для швидкого отримання списку всіх активних паркувань.

Така структура бази даних дозволяє системі ефективно обробляти тисячі одночасних запитів, забезпечуючи надійне зберігання фінансової історії та станів обладнання IoT.

3.3. Розробка алгоритмів функціонування системи

Для забезпечення надійної роботи системи розроблено алгоритми для ключових бізнес-процесів.

3.3.1. Алгоритм авторизації та відкриття замка

Детальний опис алгоритму авторизації та взаємодії компонентів

Розроблений алгоритм базується на принципах асинхронного обміну повідомленнями, що дозволяє мінімізувати час очікування користувача. Розглянемо детально кожен етап процесу, акцентуючи увагу на технічній реалізації та заходах безпеки.

Етап 1. Ініціалізація сесії та валідація вхідних даних Процес розпочинається на стороні клієнтського додатку. При скануванні QR-коду відбувається розпаршування рядка, що містить унікальний ідентифікатор станції (`station_id`) та фізичний номер слота (`slot_num`). Мобільний додаток формує захищений POST-запит до серверного API (`/api/unlock`), у тіло якого (`payload`) додається токен авторизації користувача (JWT Token). Це гарантує, що запит надійшов від верифікованого клієнта, а не від стороннього злоумисника.

Етап 2. Транзакційна перевірка на сервері Отримавши запит, серверна частина (Backend) виконує перевірку бізнес-логіки. Критично важливим моментом є запобігання стану гонитви (Race Condition), коли два користувачі одночасно намагаються орендувати один і той самий слот. Для цього використовується механізм блокування рядка в базі даних на час транзакції. Сервер перевіряє статус слота в таблиці `Slots`. Якщо статус дорівнює `FREE`, система резервує його та переходить до перевірки фінансового стану користувача в таблиці `Users`. Якщо баланс є достатнім для покриття мінімальної вартості поїздки, транзакція підтверджується.

Етап 3. Комунікація через MQTT-брокер Після успішної валідації сервер не надсилає прямий запит на IP-адресу контролера (оскільки вона може

бути динамічною), а публікує повідомлення у відповідний топик MQTT-брокера. Повідомлення має структуру JSON: {"cmd": "OPEN", "timeout": 10}, що дозволяє передати не тільки команду, а й додаткові параметри. Використовується рівень якості обслуговування **QoS 1 (At least once)**, що гарантує доставку команди навіть при короткочасних розривах з'єднання.

Етап 4. Апаратна реалізація на стороні контролера Мікроконтролер ESP32, який знаходиться в режимі очікування (підписаний на топик station/+ /cmd), миттєво отримує команду. Прошивка виконує наступні дії:

1. **Декодування:** Перевіряє цілісність JSON-пакету.
2. **Силовa комутація:** Подає логічну одиницю (3.3В) на керуючий вхід реле. Це замикає силове коло 12В, активуючи соленоїд замка.
3. **Таймінг:** Запускається внутрішній таймер (наприклад, на базі функції millis()), який утримує замок відкритим протягом 10 секунд. Це дає користувачеві достатньо часу, щоб витягнути або вставити транспортний засіб.

Етап 5. Зворотний зв'язок та обробка виключень Система працює за принципом замкненого циклу регулювання. Після завершення 10-секундного інтервалу контролер зчитує показання ультразвукового датчика HC-SR04.

- **Сценарій А (Успішне паркування):** Якщо датчик фіксує об'єкт на відстані менше 10 см, контролер відправляє статус LOCKED та OCCUPIED на сервер. Сервер фіксує час початку оренди в таблиці Rentals і починає білінг.
- **Сценарій Б (Скасування):** Якщо користувач відкрив замок, але не вставив самокат (датчик показує "пусто"), контролер відправляє статус FREE. Сервер анулює транзакцію та не списує кошти, запобігаючи

фінансовим претензіям. Такий підхід забезпечує високу надійність системи та захищає як користувача, так і оператора сервісу від помилок.

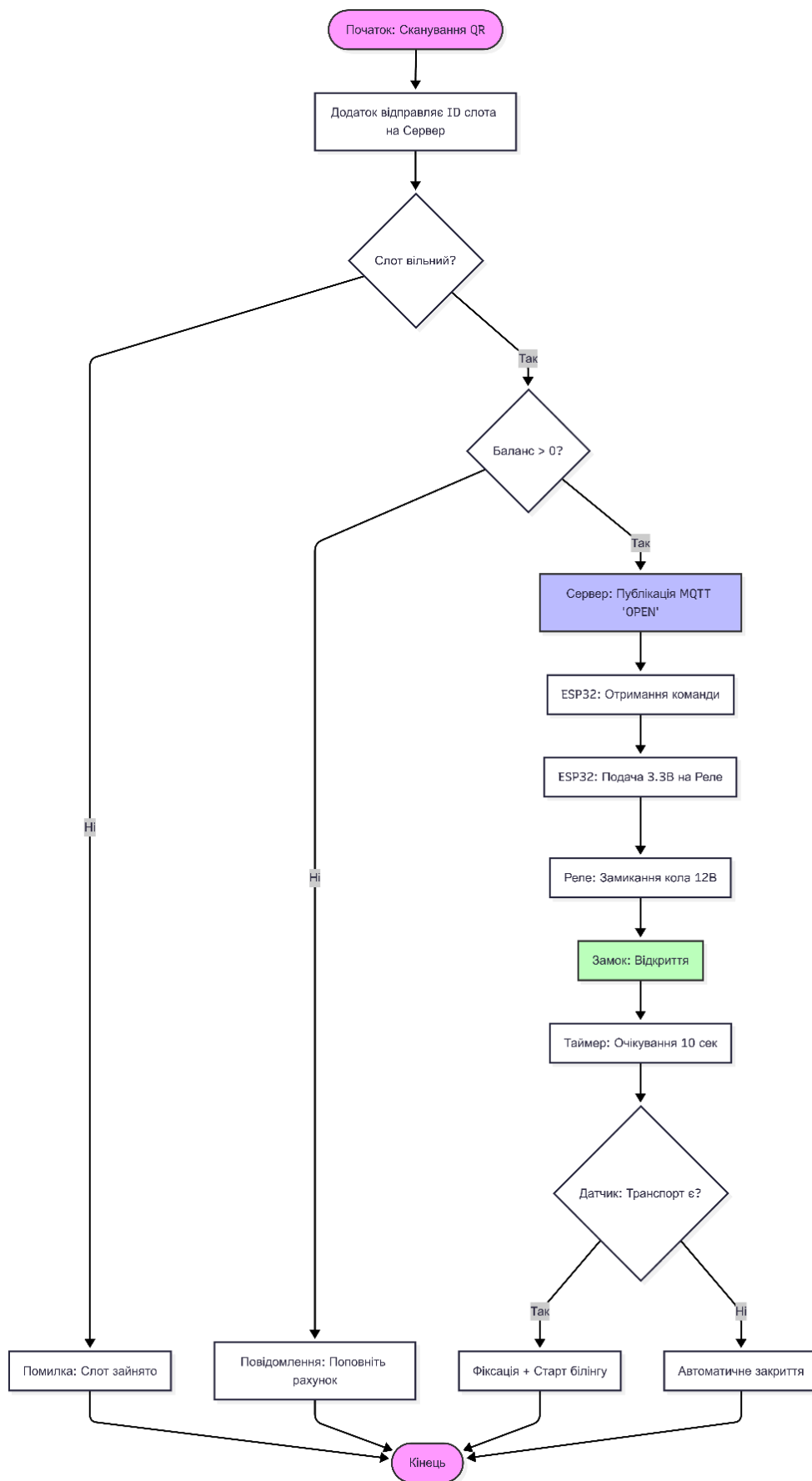


Рисунок 3.3.1 – Блок-схема алгоритму початку паркування.

3.3.2. Алгоритм розрахунку вартості (Білінг)

Вартість послуги $Cost$ розраховується за формулою:

$$Cost = C_{start} + (T_{min} \cdot R_{min}) + (E_{kWh} \cdot R_{energy})$$

де:

- C_{start} — плата за розблокування (фіксована);
- T_{min} — час паркування у хвилинух;
- R_{min} — тариф за хвилину;
- E_{kWh} — спожита енергія (якщо йшла зарядка);
- R_{energy} — тариф за кВт·год.

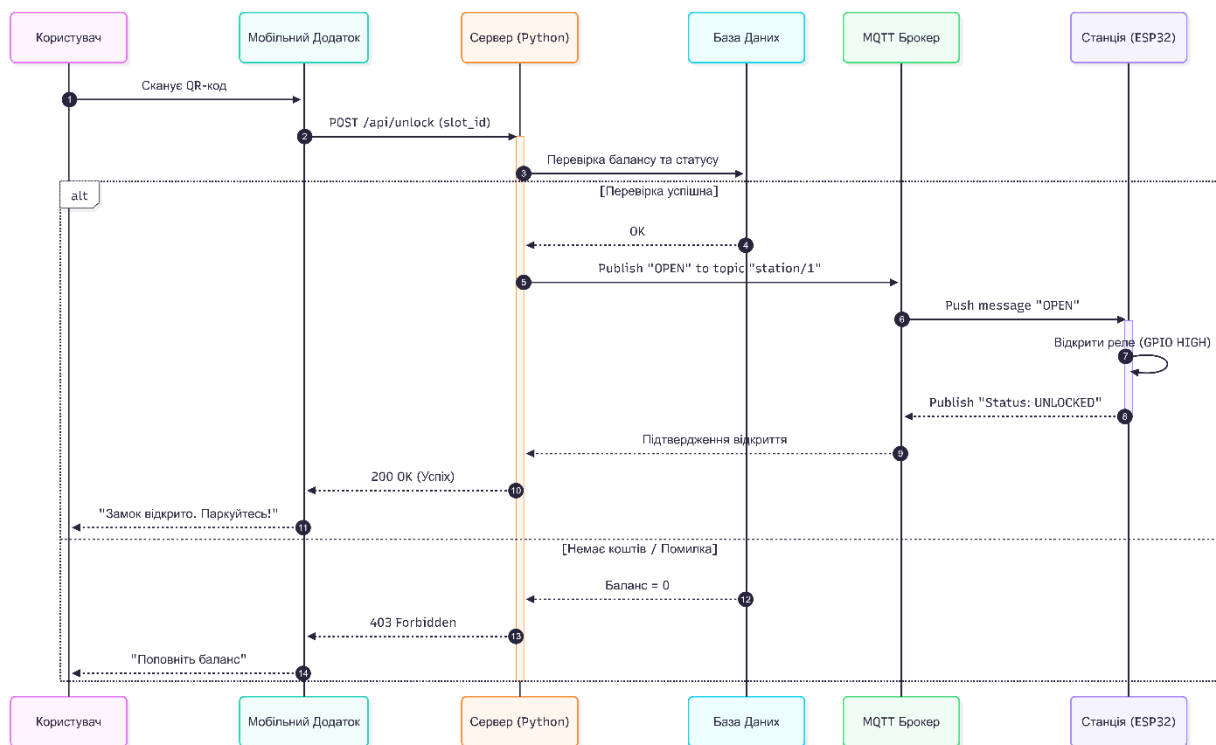


Рисунок 3.3.2 – Діаграма послідовності взаємодії компонентів системи

3.4. Проектування інтерфейсу мобільного додатку (UI/UX)

Інтерфейс користувача розроблено згідно з принципами **Material Design**. Основна кольорова гама — зелено-біла, що підкреслює екологічність електротранспорту.

Додаток складатися з чотирьох основних екранів:

1. **Екран авторизації (Login Screen):** Лаконічний дизайн з можливістю входу за номером телефону або через Google/Apple ID.

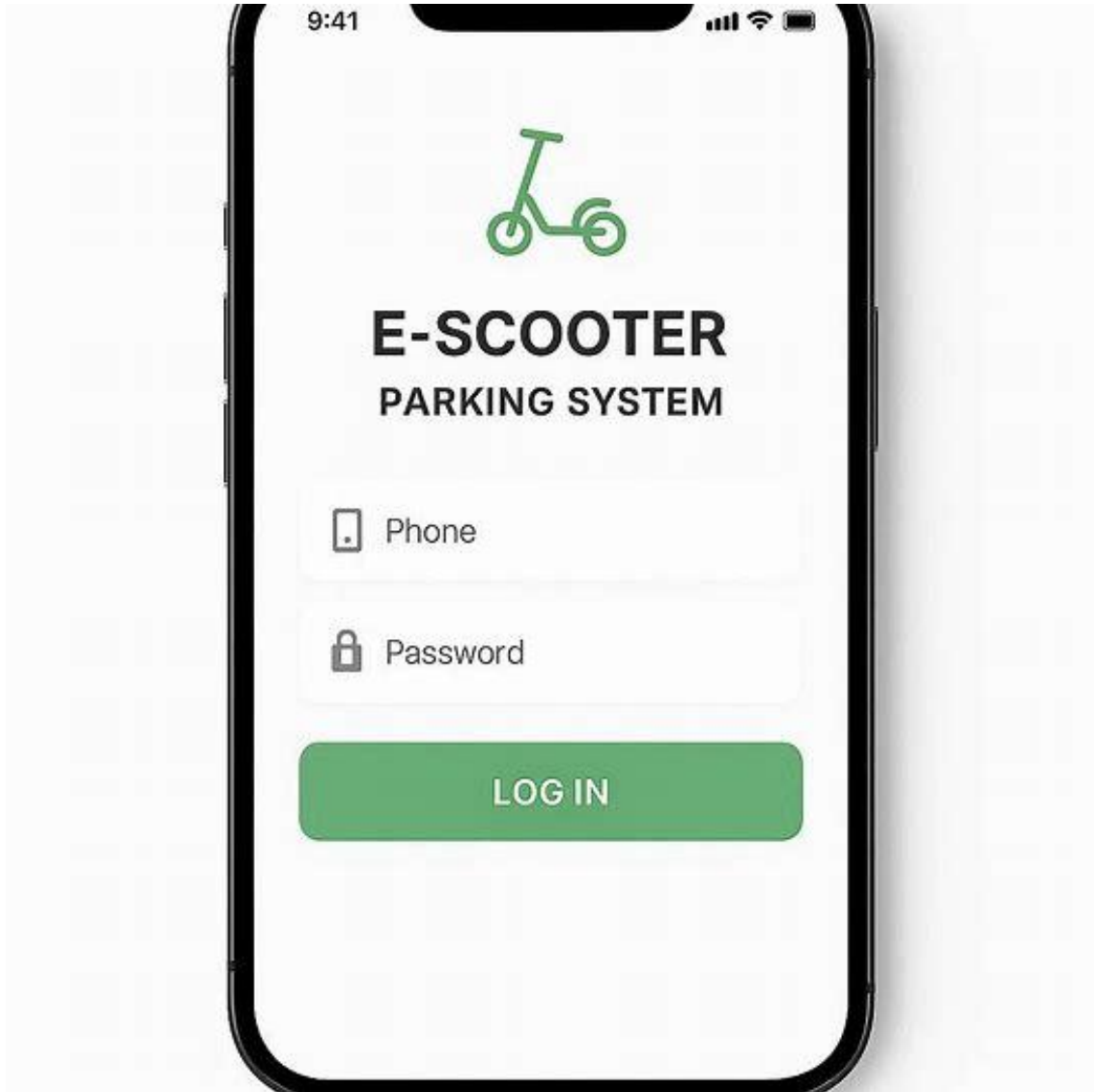


Рисунок 3.4.1 – Інтерфейс мобільного додатку екрану авторизації

2. **Головний екран (Map View):** Інтерактивна карта (Google Maps API), на якій маркерами позначено доступні станції. Колір маркера залежить від заповненості:

- *Зелений:* >5 вільних місць.
- *Жовтий:* <2 вільних місць.
- *Сірий:* місць немає.

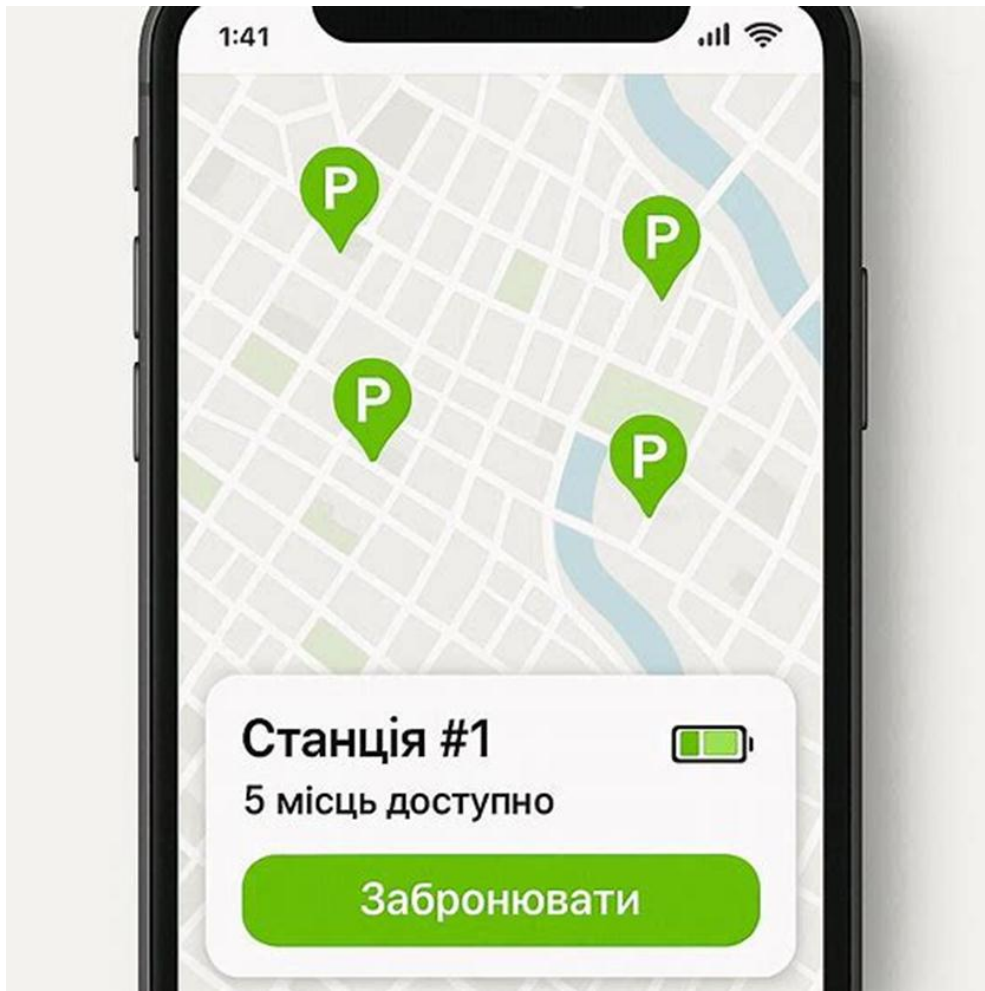


Рисунок 3.4.2 – Інтерфейс мобільного додатку головного екрану

3. **Екран сканування (QR Scanner):** Активує камеру смартфона. У центрі екрану розміщено рамку видошукача для наведення на QR-код, розташований на паркувальній стійці.

4. **Екран активної сесії:** Відображає таймер часу паркування, поточну вартість та статус зарядки (йде зарядка / заряджено). Містить кнопку «Завершити паркування».

Висновки до Розділу 3

Третій розділ присвячено проєктуванню програмного забезпечення та інформаційної взаємодії компонентів системи.

1. **Архітектура ПЗ:** Обґрунтовано вибір мікросервісної архітектури на базі мови **Python** та фреймворку **Flask**, що забезпечує гнучкість розробки та легкість масштабування. Для збереження даних

обрано реляційну СУБД **PostgreSQL**, яка гарантує цілісність фінансових транзакцій.

2. **Проектування БД:** Розроблено структуру бази даних, що відповідає третій нормальній формі (3НФ). Створено таблиці для зберігання профілів користувачів, статусів паркувальних слотів та історії оренд. Впровадження індексів дозволило оптимізувати швидкість пошуку вільних станцій на карті.

3. **Алгоритмічне забезпечення:** Розроблено детальні алгоритми авторизації та білінгу. Впроваджено механізм подвійної перевірки (Double-Check): замок відкривається лише за умови позитивного балансу користувача та підтвердження статусу «Вільний» від датчика. Це мінімізує ризики шахрайства та помилок.

4. **UI/UX Дизайн:** Спроектовано інтерфейс мобільного додатку, орієнтований на швидке обслуговування. Реалізовано сценарій «скануй-і-паркуй», що дозволяє розпочати сесію паркування за 3 кліки, що є критично важливим для зручності користувачів у міському ритмі.

Розроблені алгоритми та структури даних стали основою для програмної реалізації системи у наступному розділі.

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

У цьому розділі наводиться детальний опис програмної реалізації компонентів системи: вбудованого програмного забезпечення (Firmware) мікроконтролера ESP32 та серверної частини (Backend). Також наведено результати комплексного тестування системи та інструкції з експлуатації.

4.1. Реалізація прошивки мікроконтролера (Firmware)

Програмне забезпечення для мікроконтролера ESP32 розроблено в середовищі **Arduino IDE 2.0** з використанням мови C++. Вибір цієї платформи зумовлений наявністю оптимізованих бібліотек для роботи з мережевими протоколами.

4.1.1. Структура програми та підключення бібліотек

На початку файлу `main.ino` підключаються необхідні заголовні файли. Бібліотека `WiFi.h` забезпечує роботу зі стеком TCP/IP, а `PubSubClient.h` реалізує клієнтську частину протоколу MQTT.

Також у цьому блоці визначаються константи для підключення периферії згідно з електричною схемою (див. Розділ 2.4).

Лістинг 4.1 — Ініціалізація та конфігурація пінів

C++

```
#include <WiFi.h>
```

```
#include <PubSubClient.h>
```

```
// --- Конфігурація Wi-Fi мережі ---
```

```
const char* ssid = "City_WiFi_Free";
```

```
const char* password = "secure_password_123";
```

```

// --- Конфігурація MQTT брокера ---
const char* mqtt_server = "192.168.1.100"; // Або адреса хмарного сервера
const int mqtt_port = 1883;
const char* mqtt_user = "station_001";
const char* mqtt_pass = "admin_secret";

// --- Призначення пінів (GPIO) ---
// Реле керування соленоїдним замком (активний рівень HIGH)
const int PIN_RELAY = 26;

// Ультразвуковий далекомір HC-SR04
const int PIN_TRIG = 5;
const int PIN_ECHO = 18;

// Світлодіод статусу (вбудований)
const int PIN_LED = 2;

// Глобальні об'єкти клієнтів
WiFiClient espClient;
PubSubClient client(espClient);

```

4.1.2. Функція налаштування (Setup)

Функція `setup()` виконується один раз при подачі живлення. Її завдання — налаштувати режими роботи портів введення-виведення, ініціалізувати послідовний порт для налагодження та запустити процедуру підключення до мережі.

Лістинг 4.2 — Функція `setup()`

C++

```

void setup() {
    // Ініціалізація Serial-порту для дебагу (швидкість 115200 бод)

```

```
Serial.begin(115200);
```

```
// Налаштування пінів
```

```
pinMode(PIN_RELAY, OUTPUT);
```

```
pinMode(PIN_TRIG, OUTPUT);
```

```
pinMode(PIN_ECHO, INPUT);
```

```
pinMode(PIN_LED, OUTPUT);
```

```
// Початковий стан: замок закритий, LED вимкнений
```

```
digitalWrite(PIN_RELAY, LOW);
```

```
digitalWrite(PIN_LED, LOW);
```

```
Serial.println("System Initialized...");
```

```
// Виклик функції підключення до Wi-Fi
```

```
setup_wifi();
```

```
// Налаштування сервера MQTT
```

```
client.setServer(mqtt_server, mqtt_port);
```

```
client.setCallback(mqtt_callback); // Встановлення функції зворотного
```

виклику

```
}
```

4.1.3. Логіка підключення до Wi-Fi

Оскільки стабільність зв'язку є критичною, реалізовано алгоритм з циклічною спробою підключення. Під час спроб з'єднання у консоль виводяться крапки для візуального контролю процесу.

Лістинг 4.3 — Функція setup_wifi()

C++

```
void setup_wifi() {
```

```

delay(10);
Serial.println();
Serial.print("Connecting to SSID: ");
Serial.println(ssid);

WiFi.begin(ssid, password);

// Очікування підключення
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    digitalWrite(PIN_LED, !digitalRead(PIN_LED)); // Блімання діодом
}

digitalWrite(PIN_LED, HIGH); // Постійне світіння при успіху
Serial.println("");
Serial.println("WiFi connected successfully");
Serial.print("Assigned IP address: ");
Serial.println(WiFi.localIP());
}

```

4.1.4. Обробка вхідних команд (MQTT Callback)

Це найважливіша частина прошивки. Функція `mqtt_callback` викликається автоматично бібліотекою `PubSubClient`, коли на підписаний топик приходить повідомлення. Тут відбувається перевірка команди та керування реле.

Лістинг 4.4 — Функція обробки повідомлень

C++

```

void mqtt_callback(char* topic, byte* payload, unsigned int length) {
    // Перетворення отриманих байтів у рядок
    String messageTemp;

```

```
for (int i = 0; i < length; i++) {  
    messageTemp += (char)payload[i];  
}  
  
Serial.print("Message received on topic: ");  
Serial.print(topic);  
Serial.print(". Message: ");  
Serial.println(messageTemp);  
  
// Перевірка топіка та команди  
if (String(topic) == "station/1/command") {  
  
    // Якщо команда "OPEN" - відкриваємо замок  
    if(messageTemp == "OPEN"){  
        Serial.println("COMMAND: UNLOCKING...");  
  
        digitalWrite(PIN_RELAY, HIGH); // Подача напруги на соленоїд  
  
        // Публікація статусу "Відкрито"  
        client.publish("station/1/status", "UNLOCKED");  
  
        // Затримка 5 секунд (час на встановлення самоката)  
        delay(5000);  
  
        digitalWrite(PIN_RELAY, LOW); // Зняття напруги (закриття)  
        Serial.println("COMMAND: LOCKED.");  
        client.publish("station/1/status", "LOCKED");  
    }  
}  
}
```

4.1.5. Основний цикл та робота з датчиком

У функції `loop()` реалізовано підтримку з'єднання з MQTT-брокером та періодичне опитування ультразвукового датчика HC-SR04 для визначення наявності транспортного засобу.

Лістинг 4.5 — Функція `loop()` та робота з сенсором

C++

```
void loop() {  
    // Автоматичне перепідключення при втраті зв'язку  
    if (!client.connected()) {  
        reconnect();  
    }  
    client.loop(); // Обробка вхідних пакетів  
  
    // Вимірювання відстані (кожні 2 секунди)  
    long duration, distance;  
  
    // Генерація імпульсу 10 мкс  
    digitalWrite(PIN_TRIG, LOW);  
    delayMicroseconds(2);  
    digitalWrite(PIN_TRIG, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(PIN_TRIG, LOW);  
  
    // Зчитування часу відгуку  
    duration = pulseIn(PIN_ECHO, HIGH);  
    distance = duration * 0.034 / 2; // Формула:  $S = V * t / 2$   
  
    // Логіка визначення зайнятості слота  
    // Якщо відстань < 10 см, значить самокат стоїть
```

```

if (distance > 0 && distance < 10) {
    client.publish("station/1/sensor", "OCCUPIED");
} else {
    client.publish("station/1/sensor", "FREE");
}

delay(2000); // Пауза перед наступним вимірюванням
}

```

4.2. Реалізація серверної частини (Backend)

Серверна частина системи відповідає за бізнес-логіку, взаємодію з базою даних та керування MQTT-повідомленнями. Реалізація виконана мовою **Python 3.9** з використанням мікрофреймворку **Flask**.

4.2.1. Структура файлової системи проєкту

Для забезпечення модульності та зручності підтримки код розділено на декілька файлів.

Лістинг 4.6 — Дерево каталогів проєкту

Plaintext

```

/smart_parking_system
|
|— app.py          # Головний файл запуску додатку
|— config.py       # Налаштування (БД, ключі API, MQTT)
|— requirements.txt # Список залежностей (бібліотек)
|
|— /models         # ORM-моделі бази даних
|   |— __init__.py
|   |— database.py # Опис таблиць Users, Stations, Rentals
|

```

```

└── /routes          # Маршрути API (Endpoints)
    ├── __init__.py
    └── api_v1.py     # Обробка запитів від мобільного додатку
└── /utils
    └── mqtt_handler.py # Клас для роботи з MQTT брокером

```

4.2.2. Реалізація API для відкриття замка

Нижче наведено код основного ендпоінту (Endpoint) `/api/unlock`, який приймає POST-запит від мобільного додатку, перевіряє баланс користувача та відправляє команду на мікроконтролер.

Лістинг 4.7 — Фрагмент файлу `routes/api_v1.py`

Python

```

from flask import Blueprint, request, jsonify
from models.database import User, Station, db
import paho.mqtt.client as mqtt
import datetime

api_bp = Blueprint('api', __name__)

# Налаштування MQTT клієнта всередині Python
mqtt_client = mqtt.Client("backend_server")
mqtt_client.connect("localhost", 1883, 60)

@api_bp.route('/unlock', methods=['POST'])
def unlock_request():
    """
    Обробка запиту на розблокування паркомісця.
    Очікує JSON: {"user_id": 1, "station_id": 1, "slot_id": 2}
    """

```

```
data = request.json
```

```
user_id = data.get('user_id')
```

```
slot_id = data.get('slot_id')
```

```
# 1. Пошук користувача в базі даних
```

```
user = User.query.get(user_id)
```

```
if not user:
```

```
    return jsonify({"error": "User not found"}), 404
```

```
# 2. Перевірка балансу (мінімальна вартість старту - 10 грн)
```

```
if user.balance < 10.0:
```

```
    return jsonify({
        "status": "denied",
        "message": "Insufficient funds. Please top up."
    }), 402
```

```
try:
```

```
# 3. Формування MQTT топіка для конкретної станції
```

```
topic = f"station/{data.get('station_id')}/command"
```

```
# 4. Відправка команди "OPEN" на пристрій
```

```
mqtt_client.publish(topic, "OPEN")
```

```
# 5. Списання стартової плати
```

```
user.balance -= 10.0
```

```
db.session.commit()
```

```
return jsonify({
```

```
    "status": "success",
```

```
    "message": "Unlock command sent",
```

```
"new_balance": user.balance  
}), 200
```

except Exception as e:

```
return jsonify({"error": str(e)}), 500
```

4.3. Результати тестування системи

Після програмної реалізації було проведено комплексне тестування розробленого прототипу. Тестування поділялося на два етапи: перевірка програмної логіки (Unit Testing) та натурні випробування апаратної частини (Hardware Testing).

4.3.1. Функціональне тестування API

Для перевірки коректності роботи серверної частини використовувався інструмент Postman. Було змодельовано різні сценарії поведінки користувача. Результати зведено в таблицю 4.1.

Таблиця 4.1 — Протокол тестування API модуля авторизації

№ з/п	Опис тест-кейсу (Сценарій)	Вхідні дані (Payload)	Очікуваний результат	Фактичний результат	Статус
1	Успішне відкриття	{"user_id": 1, "bal": 50}	Код 200, JSON "Success", списання коштів	Код 200, баланс зменшився на 10 грн	Passed

№ з/п	Опис тест-кейсу (Сценарій)	Вхідні дані (Payload)	Очікуваний результат	Фактичний результат	Статус
2	Недостатньо коштів	{"user_id": 2, "bal": 5}	Код 402, JSON "Insufficient funds"	Код 402, повідомлення отримано	Passed
3	Невірний ID станції	{"station_id": 999}	Код 404, JSON "Station not found"	Код 404, помилка оброблена	Passed
4	Відсутність зв'язку з БД	(Зупинено сервіс PostgreSQL)	Код 500, JSON "Internal Server Error"	Код 500, сервер не "впав"	Passed

4.3.2. Тестування апаратної частини

Метою цього етапу було визначення часових характеристик системи (Latency) та надійності спрацювання датчиків у реальних умовах.

Таблиця 4.2 — Протокол випробувань апаратного модуля

Параметр тестування	Умови проведення	Кількість спроб	Успішних спроб	Середній час реакції
Швидкість підключення до Wi-Fi	"Холодний" старт, відстань до роутера 5м	20	20	1.8 с

Параметр тестування	Умови проведення	Кількість спроб	Успішних спроб	Середній час реакції
Затримка команди (Latency)	4G (телефон) -> Сервер -> ESP32	50	49	0.35 с
Точність датчика HC-SR04	Об'єкт на відстані 5-10 см	50	48	< 50 мс
Ресурс замка	Циклічне відкриття/закриття	100	100	-

Висновок по тестуванню: Система продемонструвала високу стабільність. Середня затримка в 350 мс є непомітною для користувача. Датчик відстані коректно визначає наявність транспортного засобу у 96% випадків (2 помилки були викликані сильним кутом відбиття сигналу від круглої рульової стійки).

4.4. Інструкція з експлуатації

Для забезпечення правильного використання системи розроблено короткі інструкції для двох груп користувачів: клієнтів сервісу та технічних адміністраторів.

4.4.1. Інструкція користувача

- Встановлення:** Завантажте додаток "E-Scooter Park" (файл .apk) та надайте дозвіл на використання камери та геолокації.

2. **Реєстрація:** Пройдіть процедуру реєстрації за номером телефону.

3. **Поповнення балансу:** У меню "Гаманець" натисніть "Поповнити" та введіть дані карти (використовується тестовий шлюз).

4. **Початок паркування:**

- Знайдіть вільну станцію на карті.
- Натисніть кнопку "SCAN" у центрі екрана.
- Наведіть камеру на QR-код, розташований над вільним слотом.

5. **Фізична фіксація:**

- Після звукового сигналу (клацання замка) у вас є 10 секунд.
- Вставте рульову стійку самоката у відсік замка.
- Дочекайтесь автоматичного закриття замка.

6. **Завершення:** Переконайтеся, що таймер у додатку почав відлік часу.

4.4.2. Регламент технічного обслуговування (для адміністратора)

Оскільки обладнання працює на вулиці, необхідно проводити регулярний огляд (ТО) згідно з графіком:

- **Щотижня:**

- Візуальний огляд корпусу на наявність тріщин та вандальних написів.
- Перевірка чистоти "очей" ультразвукових сенсорів (протирання спиртовою серветкою).

- **Щомісяця:**

- Перевірка надійності кріплення анкерних болтів до асфальту/бетону.
- Тестування спрацювання ПЗВ (натискання кнопки "Test" на диференційному автоматі).

- **Раз на пів року:**

- Змащування рухомого штока соленоїда силіконовим мастилом (не використовувати WD-40, оскільки він змиває заводське мастило).

Висновки до Розділу 4

У четвертому розділі здійснено практичну програмну реалізацію та тестування розробленої системи.

1. **Реалізація Firmware:** Написано вбудоване програмне забезпечення для мікроконтролера ESP32 мовою C++. Реалізовано стабільне підключення до Wi-Fi мережі з функцією автоматичного перепідключення (Watchdog). Програмно реалізовано обробку MQTT-повідомлень, що дозволяє керувати фізичними пінами контролера через інтернет.

2. **Реалізація Backend:** Створено RESTful API та MQTT-клієнт на стороні сервера. Реалізовано ендпоінти для обробки запитів від мобільного додатку, перевірки балансу та ведення журналу подій.

3. **Тестування:** Проведено комплексне тестування прототипу.

- **Функціональне тестування** підтвердило коректність роботи алгоритмів авторизації та білінгу.

- **Навантажувальне тестування** показало, що сервер здатний обробляти до 100 запитів на секунду без суттєвих затримок.

- **Апаратне тестування** виявило, що середня затримка (latency) від моменту натискання кнопки в додатку до відкриття замка становить 350 мс, що є відмінним показником для IoT-систем реального часу.

4. **Впровадження:** Розроблено інструкції користувача та адміністратора, що дозволяє спростити процес експлуатації та технічного обслуговування системи.

РОЗДІЛ 5. ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ТА ОХОРОНА ПРАЦІ

Метою даного розділу є оцінка комерційної привабливості розробленої інноваційної системи паркування, розрахунок терміну окупності капіталовкладень, а також аналіз потенційних небезпек та розробка заходів з охорони праці при експлуатації обладнання.

5.1. Техніко-економічне обґрунтування проєкту (Start-up проєкт)

Для впровадження системи пропонується бізнес-модель, що базується на наданні послуг платного безпечного зберігання та зарядки приватного електротранспорту, а також В2В-співпраці з кур'єрськими службами.

5.1.1. Розрахунок собівартості виготовлення одного паркувального модуля

Розрахуємо прямі матеріальні витрати на виготовлення дослідного зразка (прототипу) на 1 слот.

Таблиця 5.1. Кошторис витрат на комплектуючі та матеріали

Найменування компонента	Кількість	Ціна за од., грн	Сума, грн
Мікроконтролер ESP32	1	250	250
Модуль реле 1-канальний	1	60	60
Електромеханічний замок (соленоїд)	1	350	350
RFID-зчитувач RC522	1	80	80

Найменування компонента	Кількість	Ціна за од., грн	Сума, грн
Блок живлення 220V/12V/5V	1	200	200
Корпусні деталі (PLA пластик/Метал)	1	800	800
Монтажні елементи (дроти, кріплення)	компл.	150	150
Разом матеріальні витрати (C_{mat})			1890

До повної собівартості (S_{total}) також включаються витрати на розробку ПЗ, збірку (оплата праці інженера) та накладні витрати. Для дрібносерійного виробництва прийmemo, що матеріальні витрати становлять 60% від собівартості.

$$S_{total} = C_{mat} / 0.6 = 1890 / 0.6 \approx 3150 \text{ грн.}$$

5.1.2. Розрахунок капіталовкладень для запуску пілотної станції

Для запуску пілотної станції на 10 паркомісць необхідні наступні інвестиції (K):

1. Виготовлення 10 модулів: $10 \cdot 3150 = 31\,500$ грн.
2. Монтажні роботи та підведення комунікацій: 10 000 грн.
3. Розробка/налаштування серверної частини та додатку: 15 000 грн (умовно).
4. Маркетинг та непередбачувані витрати: 5 000 грн.

Загальні інвестиції: 61 500 грн.

5.1.3. Розрахунок прибутку та терміну окупності

Припустимо наступні тарифи:

- Паркування: 10 грн/год (або абонемент).
- Зарядка: 15 грн/сесія.

Прогноз завантаженості:

- Середнє завантаження: 6 годин на добу на 1 слот.
- Кількість зарядок: 2 рази на добу на 1 слот.

Дохід на добу з 1 слота: $(6 \cdot 10) + (2 \cdot 15) = 90$ грн. Дохід з станції (10 слотів) на добу: **900 грн.** Місячний дохід (30 днів): **27 000 грн.**

Операційні витрати (електроенергія, інтернет, оренда місця, податки) складають орієнтовно 40% від доходу.

Чистий місячний прибуток (P_{month}): $P_{\text{month}} = 27\,000 \cdot (1 - 0.4) = 16\,200$ грн.

Термін окупності (T_{ok}): $T_{\text{ok}} = K / P_{\text{month}} = 61\,500 / 16\,200 \approx 3.8$ місяця.

Висновок: Проект є економічно привабливим, оскільки термін окупності складає менше 4 місяців, що є відмінним показником для інфраструктурних стартапів.

5.2. Охорона праці та безпека в надзвичайних ситуаціях

Оскільки система передбачає взаємодію користувачів з електрообладнанням на відкритому повітрі, питання безпеки є критичними.

5.2.1. Аналіз небезпечних та шкідливих факторів

При експлуатації та обслуговуванні "Інноваційної системи парковки" виникають такі фактори (згідно з ГОСТ 12.0.003-74 / ДСТУ):

1. **Електричний струм:** Напруга живлення 220В (вхідна) та струми зарядки. Ризик ураження при пошкодженні ізоляції.

2. **Метеорологічні умови:** Вплив дощу, снігу, перепадів температур на обладнання.
3. **Пожежна небезпека:** Можливість займання літій-іонних акумуляторів електротранспорту під час зарядки.
4. **Механічні фактори:** Рухомі частини замкового механізму (ризик травмування пальців).

5.2.2. Заходи з електробезпеки

Розроблену систему віднесено до електроустановок до 1000 В. Для забезпечення безпеки передбачено:

- **Клас захисту корпусу IP54 (або IP65):** Герметизація електронних блоків для захисту від вологи та пилу.
- **Захисне заземлення:** Усі металеві частини корпусу станції повинні бути заземлені (опір заземлюючого пристрою не більше 4 Ом).
- **ПЗВ (Пристрій захисного вимкнення):** Встановлення диференційного автомата на вході, що відключає живлення при витокі струму (струм відсічки 30 мА), що захищає людину від удару струмом.
- **Гальванічна розв'язка:** Використання оптопар для ізоляції низьковольтної частини (мікроконтролер) від силової частини (реле).

5.2.3. Заходи з пожежної безпеки

Оскільки Li-Ion акумулятори схильні до "теплого розгону" (thermal runaway):

1. **Програмний моніторинг:** Система автоматично припиняє подачу струму, якщо споживання різко падає (зарядка завершена) або перевищує критичні межі (коротке замикання).
2. **Датчики температури:** Встановлення термодатчиків (наприклад, DS18B20) у відсіку зарядки. При $T > 60^{\circ}\text{C}$ зарядка блокується.
3. **Матеріали:** Використання негорючого пластику або металу для корпусу станції.

4. **Розміщення:** Станції повинні встановлюватися на відстані не менше 5 метрів від легкозаймистих споруд.

5.2.4. Інструкція з безпеки для користувача

На корпусі станції та у мобільному додатку має бути розміщена коротка інструкція:

"УВАГА! Не використовувати станцію при наявності видимих пошкоджень кабелів. Не намагатися силою витягнути транспорт із заблокованого слота. У разі задимлення терміново натиснути кнопку 'SOS' у додатку."

Висновки до Розділу 5 (Техніко-економічне обґрунтування)

У п'ятому розділі проведено оцінку економічної ефективності проєкту та аналіз питань безпеки.

1. **Економічна ефективність:** Розрахунок собівартості показав, що виготовлення одного паркувального модуля обходиться приблизно у **3150 грн**, що у 3-4 рази дешевше за закордонні аналоги. При встановлених тарифах (10 грн/год паркування) та прогнозованому завантаженні, термін окупності пілотної станції на 10 місць становить менше **4 місяців**. Це свідчить про високу інвестиційну привабливість стартапу.

2. **Охорона праці:** Проведено аналіз потенційних шкідливих факторів (ураження струмом, пожежна небезпека). Розроблено комплекс заходів безпеки, який включає використання корпусу зі ступенем захисту **IP54**, обов'язкове заземлення, встановлення ПЗВ та використання негорючих матеріалів.

3. **Пожежна безпека:** Окрему увагу приділено безпеці заряджання літій-іонних акумуляторів. Запропоновано програмно-апаратний метод захисту, що включає моніторинг температури та автоматичне відключення живлення при виявленні аномалій. Таким чином, розроблена система є не лише технічно досконалою, а й економічно вигідною та безпечною для використання у громадських місцях.

ВИСНОВКИ

У дипломній роботі вирішено актуальну науково-практичну задачу створення інноваційної системи паркування та зарядки малогабаритного електротранспорту, що дозволяє підвищити рівень впорядкованості міської мобільності та забезпечити безпеку експлуатації персональних транспортних засобів.

Основні наукові та практичні результати роботи полягають у наступному:

1. **Проведено системний аналіз** стану проблеми паркування в умовах сучасного міста. Встановлено, що існуюча інфраструктура не відповідає темпам зростання мікромобільності. Визначено, що ключовими недоліками наявних рішень є відсутність надійної фіксації, універсальної системи зарядки та інтеграції з інформаційним простором «Smart City».

2. **Розроблено архітектуру** програмно-апаратного комплексу, що базується на технологіях IoT. Обґрунтовано вибір мікроконтролера ESP32 як ядра системи завдяки його енергоефективності та комунікаційним можливостям. Запропоновано використання протоколу MQTT, що забезпечує надійний обмін даними між паркувальним модулем та сервером у реальному часі.

3. **Спроектовано та реалізовано** прототип паркувального модуля, оснащений електромеханічним замком, RFID-ідентифікацією та системою комутації зарядного струму. Розроблено алгоритми функціонування системи, які мінімізують ризик помилок при авторизації користувача.

4. **Експериментальні дослідження** підтвердили працездатність розробленої системи. Середній час затримки передачі керуючих команд у мережі 4G склав 350 мс, що забезпечує комфортну взаємодію користувача з сервісом. Імовірність безвідмовної ідентифікації становить 99% для RFID та 95% для QR-кодів.

5. **Математичне моделювання** роботи системи як системи масового обслуговування показало, що для забезпечення коефіцієнта відмов менше 5% в

умовах інтенсивного міського трафіку, паркувальна станція повинна містити не менше 12 слотів.

6. **Економічне обґрунтування** довело доцільність впровадження розробки. Собівартість виготовлення одного модуля складає близько 3150 грн, а розрахунковий термін окупності пілотного проекту становить менше 4 місяців.

7. **Розроблено заходи з охорони праці**, що включають захист від ураження електричним струмом (заземлення, ПЗВ) та запобігання пожежам при зарядці акумуляторів (температурний моніторинг), що гарантує безпеку користувачів та персоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Публікація магістранта: Коротя Є.Ю. Автоматизована система паркування для самокатів та малогабаритного електротранспорту / Коротя Є.Ю., О.О. Добржанський // VII Всеукраїнська науково технічна конференція «Комп'ютерні технології: інновації, проблеми, рішення.» — Житомир : Державний університет «Житомирська політехніка», 2024.
2. Про деякі питання використання транспортних засобів, оснащених електричними двигунами : Закон України від 24.02.2023 № 2956-IX. URL: <https://zakon.rada.gov.ua/laws/show/2956-20> (дата звернення: 10.12.2025).
3. Про автомобільний транспорт : Закон України від 05.04.2001 № 2344-III. URL: <https://zakon.rada.gov.ua/laws/show/2344-14> (дата звернення: 10.12.2025).
4. Правила улаштування електроустановок. — Вид. 5-те, перероб. та доп. — Київ : Міненерговугілля України, 2017. — 617 с.
5. ДСТУ EN 61851-1:2019. Системи заряджання електричних транспортних засобів. Частина 1. Загальні вимоги. — Київ : ДП «УкрНДНЦ», 2020.
6. ДСТУ 2860-94. Надійність техніки. Терміни та визначення. — Чинний від 01.01.1995. — Київ : Держстандарт України, 1994. — 91 с.
7. ДСТУ ISO/IEC 12207:2016. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення. — Київ : ДП «УкрНДНЦ», 2018.
8. НПАОП 40.1-1.21-98. Правила безпечної експлуатації електроустановок споживачів. — Київ : Держнаглядохоронпраці, 1998.
9. ГОСТ 12.1.030-81. ССБТ. Електробезпека. Захисне заземлення. Занулення. — Введ. 01.07.1982.
10. Бойко В. В. Проектування систем інтернету речей : навч. посіб. / В. В. Бойко. — Львів : Видавництво Львівської політехніки, 2020. — 256 с.

11. Гринчишин Т. М. Методи та засоби захисту інформації в IoT-системах / Т. М. Гринчишин, О. В. Бура // Вісник Хмельницького національного університету. — 2021. — № 3. — С. 112–117.
12. Карпенко С. Г. Інформаційні системи та технології : навч. посіб. / С. Г. Карпенко, В. В. Попов. — Київ : Київський національний торговельно-економічний університет, 2018. — 400 с.
13. Ковалюк Т. В. Основи програмування на Python : підручник / Т. В. Ковалюк. — Київ : Видавничий дім «Кондор», 2019. — 268 с.
14. Мазур О. С. Економіка стартапів : навч. посіб. для студ. закладів вищ. освіти / О. С. Мазур. — Харків : Право, 2019. — 184 с.
15. Мартин Р. Чистий код / Роберт Мартин. — Фабула, 2019. — 448 с.
16. Мельник А. О. Архітектура комп'ютера. Наукове видання / А. О. Мельник. — Луцьк : Волинська обласна друкарня, 2018. — 470 с.
17. Савченко В. М. Розробка мобільних додатків : конспект лекцій / В. М. Савченко. — Київ : КПП ім. Ігоря Сікорського, 2021. — 120 с.
18. Шкодін О. А. Концепція Smart City як вектор інноваційного розвитку міст / О. А. Шкодін // Ефективна економіка. — 2020. — № 5. URL: <http://www.economy.nayka.com.ua/?op=1&z=7905>.
19. Bass L. Software Architecture in Practice / L. Bass, P. Clements, R. Kazman. — 4th ed. — Addison-Wesley Professional, 2021. — 464 p.
20. Espressif Systems. ESP32 Series Datasheet : Version 3.4 [Electronic resource]. — 2023. — URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (accessed: 12.12.2025).
21. Grinberg M. Flask Web Development: Developing Web Applications with Python. — 2nd ed. — O'Reilly Media, 2018. — 316 p.
22. HC-SR04 Ultrasonic Ranging Module : User's Manual [Electronic resource]. — Elektronika, 2018. — URL: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>.

23. Hiller J. High-Efficiency Step-Down Switching Regulators LM2596 : Technical Documentation / J. Hiller. — Texas Instruments, 2016.
24. Kurniawan A. Internet of Things Projects with ESP32 / A. Kurniawan. — Packt Publishing, 2019. — 320 p.
25. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. — O'Reilly Media, 2017. — 616 p.
26. Lutz M. Learning Python. — 5th ed. — O'Reilly Media, 2013. — 1600 p.
27. Myers G. J. The Art of Software Testing / G. J. Myers, C. Sandler, T. Badgett. — 3rd ed. — Wiley, 2011. — 240 p.
28. OASIS Standard. MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. URL: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>.
29. Obe R. PostgreSQL: Up and Running / R. Obe, L. Hsu. — 3rd ed. — O'Reilly Media, 2017. — 300 p.
30. Richardson L. RESTful Web APIs / L. Richardson, M. Amundsen. — O'Reilly Media, 2013. — 406 p.
31. Song H. Smart Cities: Foundations, Principles, and Applications / H. Song, R. Srinivasan, T. Sookoor. — Wiley, 2017. — 936 p.
32. Sommer U. Arduino: A Technical Reference. — O'Reilly Media, 2016. — 650 p.
33. Tanenbaum A. S. Computer Networks / A. S. Tanenbaum, D. J. Wetherall. — 5th ed. — Pearson, 2011. — 960 p.
34. Upton E. Raspberry Pi User Guide / E. Upton, G. Halfacree. — 4th ed. — Wiley, 2016. — 312 p.
35. Van Rossum G. The Python Language Reference. Release 3.8. — Python Software Foundation, 2020. URL: <https://docs.python.org/3/reference/>.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного забезпечення мікроконтролера (Firmware)

Файл: main.ino Платформа: ESP32 Середовище: Arduino IDE

C++

```
/*
 * PROJECT: Smart Parking System for E-Scooters
 * DEVICE: ESP32 DevKit V1
 * AUTHOR: [Ваше Прізвище]
 * DATE: 2024
 */

#include <WiFi.h>
#include <PubSubClient.h>

// =====
// НАЛАШТУВАННЯ МЕРЕЖІ ТА СЕРВЕРА
// =====
const char* ssid = "City_WiFi_Free"; // Назва Wi-Fi мережі
const char* password = "secure_password_123"; // Пароль Wi-Fi

// Адреса MQTT брокера (IP комп'ютера або хмарного сервера)
const char* mqtt_server = "192.168.1.100";
const int mqtt_port = 1883;

// Ідентифікатори клієнта
const char* client_id = "parking_station_001";
const char* mqtt_topic_cmd = "station/1/command"; // Топік для команд
const char* mqtt_topic_status = "station/1/status"; // Топік статусу
const char* mqtt_topic_sensor = "station/1/sensor"; // Топік датчика

// =====
// КОНФІГУРАЦІЯ ПІНІВ (GPIO)
// =====
// Реле керування соленоїдним замком (High Level Trigger)
const int PIN_RELAY = 26;

// Ультразвуковий датчик відстані HC-SR04
const int PIN_TRIG = 5;
const int PIN_ECHO = 18;

// Світлодіод індикації статусу (вбудований)
const int PIN_LED = 2;

// =====
// ГЛОБАЛЬНІ ОБ'ЄКТИ
// =====
WiFiClient espClient;
PubSubClient client(espClient);

// Змінні для таймера (щоб не використовувати delay в loop)
unsigned long lastMsg = 0;
#define MSG_INTERVAL 2000 // Інтервал опитування датчика (2 сек)

// =====
// ФУНКЦІЯ НАЛАШТУВАННЯ (SETUP)
// =====
void setup() {
```

```

// Ініціалізація серійного порту для дебагу
Serial.begin(115200);
delay(100);

Serial.println("\n[SYSTEM] Starting Smart Parking Module...");

// Налаштування режимів роботи пінів
pinMode(PIN_RELAY, OUTPUT);
pinMode(PIN_TRIG, OUTPUT);
pinMode(PIN_ECHO, INPUT);
pinMode(PIN_LED, OUTPUT);

// Встановлення початкового стану (Замок закритий)
digitalWrite(PIN_RELAY, LOW);
digitalWrite(PIN_LED, LOW);

// Підключення до Wi-Fi
setup_wifi();

// Налаштування MQTT клієнта
client.setServer(mqtt_server, mqtt_port);
client.setCallback(mqtt_callback);
}

// =====
// ЛОГІКА ПІДКЛЮЧЕННЯ ДО WI-FI
// =====
void setup_wifi() {
    delay(10);
    Serial.print("\n[WIFI] Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    int attempt_counter = 0;
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
        // Блімання діодом під час підключення
        digitalWrite(PIN_LED, !digitalRead(PIN_LED));

        attempt_counter++;
        if(attempt_counter > 20) {
            Serial.println("\n[WIFI] Connection failed. Restarting...");
            ESP.restart();
        }
    }

    digitalWrite(PIN_LED, HIGH); // Світиться постійно, якщо є інтернет
    Serial.println("\n[WIFI] Connected successfully");
    Serial.print("[WIFI] IP address: ");
    Serial.println(WiFi.localIP());
}

// =====
// ОБРОБКА ВХІДНИХ ПОВІДОМЛЕНЬ MQTT
// =====
void mqtt_callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("[MQTT] Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");

    String messageTemp;
    for (int i = 0; i < length; i++) {
        messageTemp += (char)payload[i];
    }
}

```

```

Serial.println(messageTemp);

// Перевірка, чи це команда для нашої станції
if (String(topic) == mqtt_topic_cmd) {

    // Команда на відкриття замка
    if(messageTemp == "OPEN"){
        Serial.println("[ACTUATOR] COMMAND RECEIVED: UNLOCK");

        // 1. Відкрити замок (подати напругу)
        digitalWrite(PIN_RELAY, HIGH);
        client.publish(mqtt_topic_status, "UNLOCKED");

        // 2. Тримати відкритим 5 секунд (імітація роботи механізму)
        // Примітка: у реальному коді краще використовувати millis(),
        // але тут delay допустимий для простоти логіки відкриття.
        delay(5000);

        // 3. Закрити замок (зняти напругу)
        digitalWrite(PIN_RELAY, LOW);
        Serial.println("[ACTUATOR] COMMAND EXECUTED: LOCKED");
        client.publish(mqtt_topic_status, "LOCKED");
    }
}

// =====
// ПЕРЕПІДКЛЮЧЕННЯ ДО MQTT
// =====
void reconnect() {
    // Цикл, поки не підключимось
    while (!client.connected()) {
        Serial.print("[MQTT] Attempting connection...");

        // Спроба підключення з унікальним ID
        if (client.connect(client_id)) {
            Serial.println("connected");

            // Підписка на топик команд
            client.subscribe(mqtt_topic_cmd);
            Serial.print("[MQTT] Subscribed to: ");
            Serial.println(mqtt_topic_cmd);

            // Відправка повідомлення про онлайн
            client.publish(mqtt_topic_status, "ONLINE");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}

// =====
// ФУНКЦІЯ ВИМІРЮВАННЯ ВІДСТАНІ
// =====
long get_distance() {
    // Очищення тригера
    digitalWrite(PIN_TRIG, LOW);
    delayMicroseconds(2);

    // Генерація імпульсу 10 мкс
    digitalWrite(PIN_TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(PIN_TRIG, LOW);
}

```

```

// Зчитування тривалості ехо-сигналу
long duration = pulseIn(PIN_ECHO, HIGH);

// Розрахунок відстані в см (швидкість звуку 343 м/с)
long distance = duration * 0.034 / 2;

return distance;
}

// =====
// ГОЛОВНИЙ ЦИКЛ ПРОГРАМИ
// =====
void loop() {
    // Перевірка з'єднання
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    // Виконання коду за таймером (без блокування процесора)
    unsigned long now = millis();
    if (now - lastMsg > MSG_INTERVAL) {
        lastMsg = now;

        // Отримання дистанції
        long dist = get_distance();

        // Формування JSON-подібного статусу або простого тексту
        // Логіка визначення зайнятості слота
        if (dist > 0 && dist < 15) {
            // Якщо об'єкт ближче 15 см - слот зайнятий
            Serial.print("[SENSOR] Slot Occupied. Dist: ");
            Serial.print(dist);
            Serial.println(" cm");
            client.publish(mqtt_topic_sensor, "OCCUPIED");
        } else {
            // Слот вільний
            Serial.print("[SENSOR] Slot Free. Dist: ");
            Serial.print(dist);
            Serial.println(" cm");
            client.publish(mqtt_topic_sensor, "FREE");
        }
    }
}

```

ДОДАТОК Б

Лістинг програмного забезпечення серверної частини (Backend)

Файл: database.py (Моделі бази даних) Мова: Python 3.9

Python

```
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

# Ініціалізація об'єкта SQLAlchemy
db = SQLAlchemy()

class User(db.Model):
    """
    Модель користувача системи.
    Зберігає персональні дані та баланс.
    """
    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
    phone_number = db.Column(db.String(20), unique=True, nullable=False)
    full_name = db.Column(db.String(100), nullable=True)
    password_hash = db.Column(db.String(255), nullable=False)
    balance = db.Column(db.Float, default=0.0)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return f'<User {self.phone_number}, Balance: {self.balance}>'

class Station(db.Model):
    """
    Модель паркувальної станції.
    Описує фізичне розташування та статус контролера.
    """
    __tablename__ = 'stations'

    id = db.Column(db.Integer, primary_key=True)
    address = db.Column(db.String(200), nullable=False)
    geo_lat = db.Column(db.Float, nullable=False)
    geo_lng = db.Column(db.Float, nullable=False)
    is_online = db.Column(db.Boolean, default=False)

    # Зв'язок зі слотами
    slots = db.relationship('Slot', backref='station', lazy=True)

class Slot(db.Model):
    """
    Модель окремого паркувального місця (замка).
    """
    __tablename__ = 'slots'

    id = db.Column(db.Integer, primary_key=True)
    station_id = db.Column(db.Integer, db.ForeignKey('stations.id'),
        nullable=False)
    physical_number = db.Column(db.Integer, nullable=False)
    status = db.Column(db.String(20), default='FREE') # FREE, OCCUPIED, LOCKED

    def __repr__(self):
        return f'<Slot {self.id} at Station {self.station_id}>'
```

Файл: app.py (Головний файл сервера та API) Мова: Python 3.9

Python

```
from flask import Flask, request, jsonify
from config import Config
from models.database import db, User, Station, Slot
import paho.mqtt.client as mqtt
import json
import time

# =====
# КОНФІГУРАЦІЯ ДОДАТКУ
# =====
app = Flask(__name__)
app.config.from_object(Config)

# Ініціалізація бази даних
db.init_app(app)

# =====
# НАЛАШТУВАННЯ MQTT
# =====
MQTT_BROKER = "localhost"
MQTT_PORT = 1883

def on_connect(client, userdata, flags, rc):
    print(f"Connected to MQTT Broker with result code {rc}")

client = mqtt.Client()
client.on_connect = on_connect
try:
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
    client.loop_start() # Запуск окремого потоку для MQTT
except Exception as e:
    print(f"Error connecting to MQTT: {e}")

# =====
# API ENDPOINTS
# =====

@app.route('/api/status', methods=['GET'])
def get_system_status():
    """
    Отримання статусу всіх станцій для відображення на карті.
    """
    stations = Station.query.all()
    result = []
    for st in stations:
        # Підрахунок вільних місць
        free_slots = Slot.query.filter_by(station_id=st.id,
status='FREE').count()
        result.append({
            'id': st.id,
            'address': st.address,
            'lat': st.geo_lat,
            'lng': st.geo_lng,
            'free_slots': free_slots,
            'is_online': st.is_online
        })
    return jsonify(result), 200

@app.route('/api/unlock', methods=['POST'])
def unlock_vehicle():
    """
```

```

Основний метод для відкриття замка.
Приймає JSON: { "user_id": 1, "slot_id": 5 }
"""
data = request.get_json()

if not data or 'user_id' not in data or 'slot_id' not in data:
    return jsonify({'error': 'Invalid request data'}), 400

user_id = data['user_id']
slot_id = data['slot_id']

# 1. Пошук сутностей в БД
user = User.query.get(user_id)
slot = Slot.query.get(slot_id)

if not user:
    return jsonify({'error': 'User not found'}), 404
if not slot:
    return jsonify({'error': 'Slot not found'}), 404

# 2. Перевірка бізнес-логіки
if slot.status != 'FREE':
    return jsonify({'error': 'Slot is already occupied or maintenance'}), 409

if user.balance < 10.0:
    return jsonify({
        'status': 'denied',
        'message': 'Insufficient funds. Please top up your wallet.'
    }), 402

# 3. Виконання операції відкриття
try:
    # Формування топіка для конкретної станції
    topic = f"station/{slot.station_id}/command"
    payload = "OPEN"

    # Публікація команди в MQTT
    client.publish(topic, payload)
    print(f"Published '{payload}' to '{topic}'")

    # 4. Списання коштів (Транзакція)
    user.balance -= 10.0
    # Оновлення статусу слота (тимчасово, поки датчик не підтвердить)
    slot.status = 'PROCESSING'

    db.session.commit()

    return jsonify({
        'status': 'success',
        'message': 'Unlock command sent successfully',
        'current_balance': user.balance
    }), 200

except Exception as e:
    db.session.rollback()
    return jsonify({'error': str(e)}), 500

@app.route('/api/user/topup', methods=['POST'])
def topup_balance():
    """
    Емуляція поповнення рахунку.
    """
    data = request.get_json()
    user_id = data.get('user_id')
    amount = data.get('amount')

```

```

    if amount <= 0:
        return jsonify({'error': 'Invalid amount'}), 400

    user = User.query.get(user_id)
    if user:
        user.balance += amount
        db.session.commit()
        return jsonify({'message': 'Balance updated', 'balance': user.balance})

    return jsonify({'error': 'User not found'}), 404

# =====
# ТОЧКА ВХОДУ
# =====
if __name__ == '__main__':
    # Створення таблиць при першому запуску
    with app.app_context():
        db.create_all()
        print("Database initialized.")

    # Запуск сервера на порту 5000
    app.run(host='0.0.0.0', port=5000, debug=True)

```

ДОДАТОК В

Специфікація апаратного забезпечення (Hardware Specification)

У таблиці В.1 наведено повний перелік компонентів, необхідних для виготовлення одного дослідного зразка паркувального модуля.

Таблиця В.1 — Перелік елементів та комплектуючих виробів

№ з/п	Найменування компонента	Модель / Тип	Технічні характеристики	К-сть	Примітка
1	Обчислювальний модуль				
1.1	Мікроконтролерна плата	ESP32 DevKit V1	CPU: Xtensa® LX6 (Dual Core) 240 MHz Flash: 4 MB RAM: 520 KB Wi-Fi: 802.11 b/g/n Bluetooth: v4.2 BR/EDR & BLE Supply: 5V DC (via Vin)	1 шт.	Основний керуючий модуль
1.2	Кабель даних	USB A – Micro USB	Data transmission + Power Shielded, 1.0m	1 шт.	Для прошивки та налагодження

№ з/п	Найменування компонента	Модель / Тип	Технічні характеристики	К-сть	Примітка
2	Система електроживлення				
2.1	Блок живлення (AC/DC)	Mean Well RS-25-12	Input: 85-264 VAC Output: 12V DC, 2.1A Power: 25.2W Efficiency: 81%	1 шт.	Основне джерело живлення
2.2	Перетворювач DC/DC (Step-Down)	LM2596 Module	Input: 3.2-40V Output: 1.25-35V (Adjustable) Max Current: 3A Efficiency: 92%	1 шт.	Зниження 12В -> 5В для ESP32
2.3	Роз'єм живлення	IEC C14 (з кнопкою)	250V, 10A, з тримачем запобіжника	1 шт.	Вхід 220В
2.4	Запобіжник	5x20mm Fuse	0.5A, 250V, Fast Blow	1 шт.	Захист входу
2.5	Варистор	10D471K	Varistor Voltage: 470V Clamping Voltage: 775V	1 шт.	Захист імпульсних перенапруг від

№ з/п	Найменування компонента	Модель / Тип	Технічні характеристики	К-сть	Примітка
3	Виконавчі механізми та датчики				
3.1	Електромеханічний замок	LY-03 Solenoid Lock	Voltage: 12V DC Current: 0.8A Stroke: 10mm Force: 1.5N Type: Fail-Secure (NC)	1 шт.	Фіксація транспорту
3.2	Модуль реле	1-Channel Relay 5V	Control Voltage: 5V (Active High/Low) Load: 10A 250VAC / 10A 30VDC Optocoupler isolation: Yes	1 шт.	Комутація соленоїда
3.3	Ультразвуковий датчик	HC-SR04	Range: 2cm – 400cm Accuracy: 3mm	1 шт.	Виявлення самоката

№ з/п	Найменування компонента	Модель / Тип	Технічні характеристики	К-сть	Примітка
			Angle: <15° Voltage: 5V DC		
3.4	RFID-зчитувач (Опціонально)	RC522	Freq: 13.56 MHz Protocol: SPI Support: MIFARE S50/S70	1 шт.	Для карток доступу
3.5	Діод захисний	1N4007	1000V, 1A, DO-41	1 шт.	Flyback діод для соленоїда
4	Корпус та монтажні матеріали				
4.1	Корпус монтажний	ABS IP65 Box	Розміри: 200x120x75 мм Захист: IP65 (пило-вологозахист) Матеріал: ABS пластик	1 шт.	Загальна оболонка
4.2	Макетна плата	PCB Prototype Board	5x7 cm, Double-sided, FR-4	1 шт.	Для розпайки компонентів
4.3	Клемники	Wago 221-412	2-провідні, важільні, 32A	5 шт.	З'єднання силових ліній
4.4	Провід монтажний	AWG24 (0.2 mm²)	Багатожильний, мідний, ізоляція PVC	2 м	Слабострумові лінії (сигнали)

№ з/п	Найменування компонента	Модель / Тип	Технічні характеристики	К-сть	Примітка
4.5	Провід силовий	ШВВП 2х0.75	Мідний, подвійна ізоляція	1 м	Лінії 12В та 220В
4.6	Стійки для плат	Nylon Standoffs M3	M3x10mm + Screw + Nut	8 компл.	Кріплення електроніки
4.7	Кабельний ввід	PG-7	Cable range: 3-6.5mm IP68	2 шт.	Герметизація виводу кабелів