Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра ____Автоматизації проектування обчислювальної техніки_____
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____
_____Програмно-апаратний комплекс аналізу швидкісних характеристик__
_____велосипеда на базі ESP8266_____
(тема)

Виконав:
здобувач 4 року навчання,
групи __КІУКІ-21-8_____
_____Чисятков Д. Ю. _____
(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
Освітня програма _____
_____Комп'ютерна інженерія_____
( повна назва освітньої програми)

Керівник _____проф. Хаханова Г.В._____
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

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

Тип програми Освітньо-професійна

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Чистякову Денису Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмно-апаратний комплекс аналізу швидкісних характеристик велосипеда на базі ESP8266
затверджена наказом університету від 21 травня 2025 р. № 403 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 06 червня 2025 р.
3. Вихідні дані до роботи Мікроконтролер ESP8266 з підключеним світлодіодом WS2813B та датчиком Холла, мобільний пристрій (смартфон) з операційною системою iOS або Android, бездротова мережа Wi-Fi, налаштування користувача
4. Перелік питань, що потрібно опрацювати в роботі Аналіз та обґрунтування вибору компонентів системи, розробка апаратної та програмної частини, реалізація комунікації між мікроконтролером та мобільним додатком, розробка функціональної схеми пристрою, тестування спроектованого пристрою

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____

Структурна схема КС, схема підключення компонентів пристрою, скріншоти з мобільного застосунку, 16 слайдів

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

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

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Видача теми проекту, узгодження і затвердження теми | 06.05.2025 – 09.05.2025 | |
| 2 | Аналіз та вибір методу вирішення поставленої задачі поставленої задачі | 10.05.2025 – 15.05.2025 | |
| 3 | Ознайомлення з літературними джерелами аналіз та вибір методу вирішення поставленої задачі вибір системних засобів вирішення | 16.05.2025 – 20.05.2025 | |
| 4 | Проектування апаратної програмної частини | 21.05.2025 – 25.05.2025 | |
| 5 | Налагодження та тестування комп'ютерної системи | 26.05.2025 – 28.05.2025 | |
| 6 | Оформлення пояснювальної записки | 29.05.2025 – 01.06.2025 | |
| 7 | Перевірка виконаного проекту керівником, допуск до захисту | 02.06.2025 – 05.06.2025 | |
| 8 | Захист проекту | 12.06.2025 – 24.06.2025 | |

Дата видачі завдання 06 травня 2025 р.

Здобувач _____ Чистяков Д.Ю.
(підпис)

Керівник роботи _____ проф. Хаханова Г.В.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка валіфікаційної роботи бакалавра містить: 46 с., 15 рис., 10 джерел.

ESP8266, WI-FI КОМУНІКАЦІЯ, ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС, ПРОГРАМНА ЧАСТИНА, ДАТЧИК ХОЛЛА.

Метою кваліфікаційної роботи є розробка програмно-апаратного комплексу аналізу швидкісних характеристик велосипеда на базі ESP8266, що забезпечує точне вимірювання, відображення та аналіз швидкісних параметрів під час велосипедних поїздок.

У ході виконання кваліфікаційної роботи було створено інтегрований програмно-апаратний комплекс, що складається з апаратної частини на базі ESP8266 та мобільного застосунку на React Native. Апаратна частина включає мікроконтролер ESP8266, датчик Холла для вимірювання швидкості та світлодіод WS2813B для індикації.

Програмне забезпечення мікроконтролера реалізує алгоритми аналізу швидкості та відстані, керування світлодіодом та створення Wi-Fi точки доступу.

Мобільний застосунок забезпечує інтерфейс для відображення та аналізу швидкісних параметрів, збереження історії поїздок та налаштування комплексу. Реалізовано бездротову комунікацію між компонентами програмно-апаратного комплексу через Wi-Fi. Особлива увага приділена оптимізації енергоспоживання та точності аналізу даних.

Розроблений програмно-апаратний комплекс демонструє практичне застосування знань з предметів спеціальності «Комп'ютерна інженерія».

# ABSTRACT

Course project: 46 pages, 15 figures, 10 sources.

ESP8266, WI-FI COMMUNICATION, HARDWARE AND SOFTWARE COMPLEX, SOFTWARE PART, HALL SENSOR.

The purpose of the qualification work is to develop a hardware-software complex for analyzing bicycle speed characteristics based on ESP8266, which provides accurate measurement, display and analysis of speed parameters during bicycle trips.

During the qualification work, an integrated hardware-software complex was created, consisting of hardware based on ESP8266 and a mobile application on React Native. The hardware part includes an ESP8266 microcontroller, a Hall sensor for measuring speed and a WS2813B LED for indication.

The microcontroller software implements algorithms for analyzing speed and distance, controlling the LED and creating a Wi-Fi access point.

The mobile application provides an interface for displaying and analyzing speed parameters, saving trip history and configuring the complex. Wireless communication between the components of the hardware-software complex via Wi-Fi is implemented. Special attention is paid to optimizing energy consumption and data analysis accuracy.

The developed software and hardware complex demonstrates the practical application of knowledge in the subjects of the Computer Engineering specialty.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GPIO (General Purpose Input/Output) - інтерфейс введення/виведення загального призначення, набір цифрових контактів мікроконтролера, які можна запрограмувати як входи або виходи.

TSX (TypeScript JSX) - розширення файлу, що використовується в React для компонентів, написаних на TypeScript з підтримкою JSX синтаксису.

Vin (Voltage input) - вхідна напруга, контакт для подачі живлення на електронний пристрій.

Wi-Fi (Wireless Fidelity) - технологія бездротової локальної мережі з пристроями на основі стандартів IEEE 802.11.

Км/г - кілометри за годину (одиниця вимірювання швидкості).

Км - кілометри (одиниця вимірювання відстані).

В - вольти (одиниця вимірювання напруги).

МА - міліампери (одиниця вимірювання струму).

Мс - мілісекунди (одиниця вимірювання часу).

# ВСТУП

У сучасному світі програмно-апаратні комплекси відіграють все більшу роль у різних сферах життя, включаючи спорт, фітнес та охорону здоров'я. Вони змінюють спосіб, яким ми збираємо, обробляємо та аналізуємо дані про фізичну активність, відкриваючи нові можливості для покращення спортивних результатів та загального стану здоров'я.

Одним з напрямків, що активно розвивається, є розробка програмно-апаратних комплексів для моніторингу та аналізу спортивних показників, зокрема у велоспорті. Від розробника вимагається створення інтегрованого рішення, що включає апаратну частину для збору даних, алгоритми аналізу інформації та програмне забезпечення з інтуїтивно зрозумілим інтерфейсом для відображення результатів.

Цікавим підходом до розробки таких комплексів є використання мікроконтролерів, таких як ESP8266, які виступають центральним елементом для збору та аналізу даних. ESP8266 - це потужний та економічний модуль, який дозволяє створювати компактні програмно-апаратні комплекси. Завдяки вбудованим можливостям зв'язку, він може ефективно взаємодіяти з іншими компонентами комплексу, включаючи мобільні застосунки для зберігання та поглибленого аналізу даних.

Програмно-апаратний комплекс аналізу швидкісних характеристик велосипеда на базі ESP8266, що поєднує мікроконтролер ESP8266, датчики для вимірювання швидкості та програмне забезпечення для аналізу даних, є прикладом сучасного підходу до створення спеціалізованих інтегрованих рішень.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Предметна область даної роботи охоплює сферу велоспорту та фітнесу, зокрема, системи моніторингу та аналізу показників велосипедної їзди. В останні роки спостерігається зростання популярності велоспорту як професійного, так і аматорського рівня. Це призвело до підвищення попиту на точні та функціональні інструменти для відстеження показників тренувань та змагань.

Аналіз існуючих рішень на ринку велокомп'ютерів виявив ряд проблем:

– обмежена функціональність базових моделей, які часто надають лише базову інформацію про швидкість та відстань;

– висока вартість моделей, що робить їх недоступними для широкого кола користувачів;

– відсутність інтеграції з сучасними смартфонами у багатьох моделях, що обмежує можливості аналізу та зберігання даних;

– недостатня увага до аспектів безпеки, таких як індикація гальмування [1].

Метою розробки даної системи є створення ефективного інструменту для обробки швидкісних характеристик велосипеду, що дозволить велосипедистам точно відстежувати свої показники під час тренувань та змагань. Система, як сукупність відношень між компонентами та зовнішнім середовищем [2], призначена для збору, обробки та відображення даних про швидкість, пройдену відстань та інші важливі параметри велосипедної їзди.

Розроблюваний комплекс має ряд унікальних особливостей, які відрізняють її від існуючих рішень:

– інтеграція з смартфоном: комплекс відображає дані на екрані смартфона, що забезпечує більш зручний та інформативний інтерфейс. Це дозволяє використовувати потужність та гнучкість сучасних мобільних пристроїв для аналізу даних;

– пряма передача даних з датчика: забезпечується безпосередня передача даних з датчика на смартфон, що підвищує точність та швидкість оновлення інформації. Це дозволяє отримувати актуальні дані в режимі реального часу;

– світлодіодна індикація з функцією сигналізації гальмування: унікальною особливістю є наявність світлодіода, який не лише привертає увагу миготінням, але й сигналізує про гальмування. Ця функція значно підвищує безпеку велосипедиста, особливо при їзді в умовах обмеженої видимості;

– зберігання історії поїздок: застосунок дозволяє зберігати дані про поїздки, що дає можливість аналізувати прогрес та планувати тренування більш ефективно. Ця функція забезпечує довгострокове відстеження результатів та допомагає в плануванні тренувального процесу;

– налаштування параметрів: система дозволяє легко налаштовувати параметри, такі як розмір колеса та режими роботи світлодіода, через зручний інтерфейс смартфона. Це забезпечує гнучкість та адаптивність системи до різних типів велосипедів та потреб користувачів.

Постановка задачі полягає у розробці програмно-апаратного комплексу для обробки швидкісних характеристик велосипеду, що включає:

– створення апаратної частини на базі мікроконтролера ESP8266 з датчиком Холла та світлодіодом WS2813B;

– розробку програмного забезпечення для мікроконтролера, яке забезпечить точне вимірювання швидкості та відстані, обробку даних з датчика, керування світлодіодною індикацією та передачу даних через Wi-Fi;

– створення мобільного застосункудля відображення даних в реальному часі, зберігання історії поїздок, аналізу даних та налаштування параметрів системи;

– реалізацію функції сигналізації гальмування.

# 2 РОЗРОБКА СТУРКТУРИ КОМПЛЕКСУ

Програмно-апаратний комплекс (рисунок 2.1) для аналізу швидкісних характеристик велосипеда на базі ESP8266 являє собою інтегроване рішення, що включає в себе апаратне та програмне забезпечення та забезпечує ефективний зв'язок між ними. Цей комплекс складається з двох основних частин, які взаємодіють між собою через Wi-Fi з'єднання, утворюючи єдину функціональну систему.


Рисунок 2.1 – Стурктурна організація компексу

Перша частина компексу базується на мікроконтролері ESP8266 та включає три рівні організації. На нижньому рівні знаходиться апаратне забезпечення, що складається з мікроконтролера ESP8266, датчика Холла KY-003 для вимірювання швидкості, світлодіода WS2813B для візуальної

індикації та системи живлення на базі акумулятора 18650. Системне програмне забезпечення цієї частини представлене Arduino Core та Wi-Fi стеком, які забезпечують базову функціональність мікроконтролера та можливість мережевої комунікації. На верхньому рівні знаходиться прикладне програмне забезпечення у вигляді спеціально розробленої прошивки, яка реалізує алгоритми обробки даних, керування світлодіодною індикацією та обмін інформацією з мобільним додатком.

Друга частина системи реалізована на базі смартфона і також має трирівневу структуру. Апаратною основою тут виступає сам смартфон з його процесором, пам'яттю та модулем Wi-Fi. Системне програмне забезпечення представлене операційною системою Android або iOS, яка надає необхідні системні сервіси та API для роботи з апаратним забезпеченням. Прикладний рівень реалізований у вигляді мобільного додатку, розробленого з використанням фреймворку React Native, який забезпечує зручний користувацький інтерфейс для відображення даних, керування налаштуваннями системи та аналізу історії поїздок.

Взаємодія між цими частинами здійснюється через Wi-Fi з'єднання, що забезпечує надійний та швидкий обмін даними. Мікроконтролер створює власну точку доступу, до якої підключається смартфон, після чого встановлюється безпосередній зв'язок між прикладним програмним забезпеченням обох частин системи. Це дозволяє передавати дані про швидкість та відстань в реальному часі [3], а також забезпечує можливість віддаленого керування налаштуваннями системи.

Така архітектура системи забезпечує оптимальний розподіл функцій між компонентами, де апаратна частина на базі ESP8266 відповідає за точні вимірювання та первинну обробку даних, а мобільний застосунок забезпечує зручний інтерфейс користувача та розширені можливості аналізу даних.

# 3 РЕАЛІЗАЦІЯ АПАРАТНОЇ ЧАСТИНИ

## 3.1 Основні компоненти системи та їх характеристики

### 3.1.1 ESP8266 Node MCU

ESP8266 Node MCU - це центральний елемент системи, який відповідає за обробку даних, керування іншими компонентами та комунікацію з мобільним додатком. Мікроконтролер ESP8266, встановлений на платі NodeMCU, представляє собою потужний та компактний модуль (рисунок 3.1) з вбудованим Wi-Fi, що робить його ідеальним вибором для IoT проектів та систем збору даних [4].

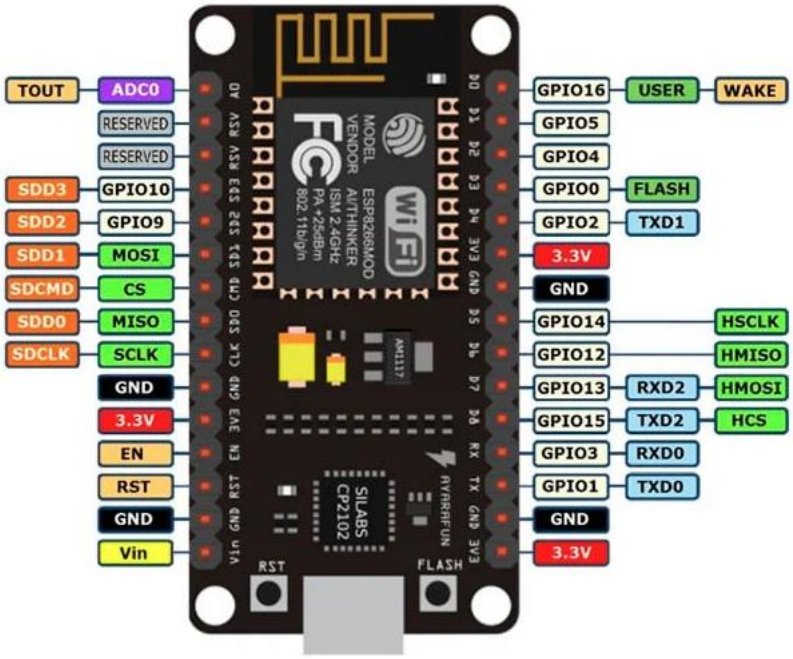


Рисунок 3.1 – Розташування пінів у модулі

Технічні характеристики [5]:

- процесор:
  - тип: Tensilica L106 32-bit RISC;
  - тактова частота: 80 МГц (можливість розгону до 160 МГц);
  - продуктивність: до 600 DMIPS;
- пам'ять:
  - RAM: 80 КБ оперативної пам'яті для даних;
  - ROM: 64 КБ для інструкцій;
  - зовнішня флеш-пам'ять: 4 МБ;
  - EEPROM: 512 байт для зберігання налаштувань;
- входи/виходи:
  - GPIO0-GPIO15: цифрові входи/виходи загального призначення;
  - GPIO1(TX) і GPIO3(RX): піни для UART інтерфейсу;
  - GPIO4, GPIO5: піни для I2C інтерфейсу (SDA, SCL);
  - GPIO12-GPIO14: піни для SPI інтерфейсу;
  - GPIO16: спеціальний пін для пробудження з режиму глибокого сну;
  - підтримка переривань на всіх GPIO пінах;
  - вбудований 10-бітний АЦП;
  - апаратна підтримка інтерфейсів I²C, SPI, UART та PWM;
- Wi-Fi можливості:
  - стандарт: IEEE 802.11 b/g/n;
  - підтримка режимів: Station / SoftAP / SoftAP+Station;
  - безпека: WPA/WPA2;
  - вбудована TCP/IP підтримка;
  - максимальна швидкість передачі: до 72.2 Mbps;
- електричні характеристики:
  - робоча напруга: 3.3В;
  - вхідна напруга (через VIN): 5-12В;
  - струм споживання в активному режимі: ~80мА;

- струм споживання в режимі глибокого сну: ~20мкА;
- вихідний струм GPIO: до 12мА на пін.

Обґрунтування вибору:

– вбудований Wi-Fi модуль дозволяє легко реалізувати бездротову передачу даних;

– достатня обчислювальна потужність для обробки даних в реальному часі;

– низьке енергоспоживання, що важливо для автономної роботи;

– велика кількість доступних бібліотек та широка підтримка спільноти розробників;

– наявність достатньої кількості GPIO для підключення всіх необхідних компонентів;

– підтримка переривань дозволяє точно вимірювати швидкість обертання колеса;

– вбудована флеш-пам'ять достатня для зберігання програми та налаштувань;

– апаратна підтримка різних інтерфейсів забезпечує гнучкість при розробці.

### 3.1.2 Датчик Холла KY-003

Датчик Холла KY-003 забезпечує вимірювання швидкості обертання колеса, що є основою для розрахунку швидкості та відстані. Датчик реагує на зміну магнітного поля, яке створюється магнітом, встановленим на спиці колеса велосипеда.

Рисунок 3.2 – Датчик Холла KY-003

Технічні характеристики :

– напруга живлення: 3.3В - 5В (сумісний з ESP8266);

– вихідний сигнал: Цифровий (HIGH/LOW) або аналоговий;

– розміри плати: 18.5мм х 15мм;

– чутливість: спрацьовує при наближенні магнітного поля;

– відстань спрацьовування: 2-20 мм (залежить від сили магніту);

– температурний діапазон роботи: -40°C до +85°C;

– струм споживання: приблизно 4-8 мА;

– інтерфейс: 4 виводи (VCC, GND, Digital Out, Analog Out);

– індикація: вбудований світлодіод, який загоряється при виявленні магнітного поля.

Обґрунтування вибору:

– цифровий вихід забезпечує простоту інтеграції з мікроконтролером без необхідності АЦП;

– висока швидкодія дозволяє точно вимірювати навіть при високих швидкостях обертання;

– стабільність сигналу завдяки чіткому розділенню логічних рівнів;

– можливість прямого підключення до GPIO ESP8266 без додаткових компонентів;

– низьке енергоспоживання в неактивному стані;

– захист від завад завдяки цифровому характеру сигналу.

### 3.1.3 Світлодіод WS2813B

Світлодіод WS2813B забезпечує сигналізацію про гальмування та візуальні ефекти, для покращення видимості у темноті.

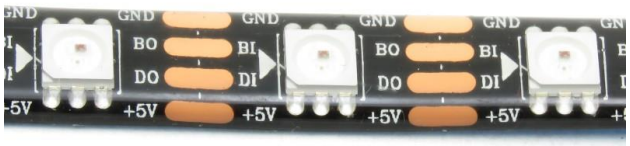


Рисунок 3.3 – Світлоідіодна стрічка WS2813B

Технічні характеристики :

– напруга живлення: 5В;

– розміри : 5050 SMD (5.0мм x 5.0мм);

– термін служби: >20,000 годин;

– температурний діапазон роботи: -40°C до +85°C;

– струм споживання: до 60 мА при повній яскравості (всі кольори).

Обґрунтування вибору:

– програмована RGB-світлодіодна стрічка дозволяє реалізувати різні режими індикації;

– висока яскравість забезпечує гарну видимість;

– низьке енергоспоживання в порівнянні з традиційними світлодіодами.

### 3.1.4 Контролер заряду TP4056

Контролер заряду TP4056 забезпечує безпечну зарядку акумулятора та захищає його від пошкоджень.


Рисунок 3.4 – Контролер заряду TP4056

Технічні характеристики :

– вхідна напруга: 5В;

– захист від: перезаряду, короткого замикання;

– розміри : 36мм х 17мм х 14мм;

– температурний діапазон роботи: -10°C до +85°C.

Обґрунтування вибору:

– вбудований захист від перезаряду та глибокого розряду подовжує термін служби акумулятора;

– можливість заряджання через стандартний micro-USB роз'єм;

– компактні розміри дозволяють легко інтегрувати в систему.

### 3.1.5 Перетворювач напруги MT3608

Перетворювач напруги MT3608 забезпечує стабільну напругу 5В для живлення мікроконтролера та світлодіоду.

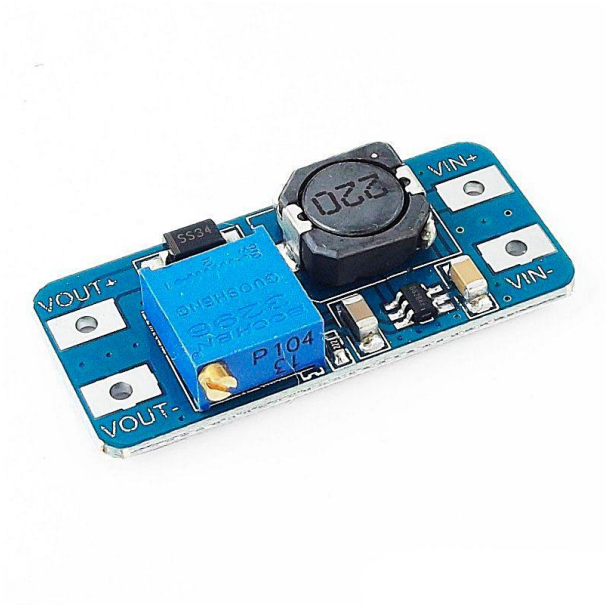


Рисунок 3.5 – Перетворювач напруги MT3608

Технічні характеристики :

– вхідна напруга: 2В - 24В;
– вихідна напруга: 5В - 28В (регульована);
– розміри : 36мм х 17мм х 14мм;
– струм спокою: 100 мкА;
– температурний діапазон роботи: -40°С до +85°С;
– максимальний вихідний струм: 2А (рекомендований до 1.5А для стабільної роботи);
– ефективність: до 97%;
– частота перемикання: 1.2 МГц.

## 3.2 Функціональна схема

Для створення апаратної частини комп'ютерної системи всі компоненти були об'єднані в єдину функціональну схему (рисунок 3.3). Це забезпечує ефективну взаємодію між усіма елементами та оптимальне використання ресурсів системи. Така інтеграція дозволяє створити компактну, енергоефективну та надійну апаратну частину комп'ютерної системи, яка здатна ефективно обробляти швидкісні характеристики велосипеду та взаємодіяти з мобільним додатком через Wi-Fi з'єднання.

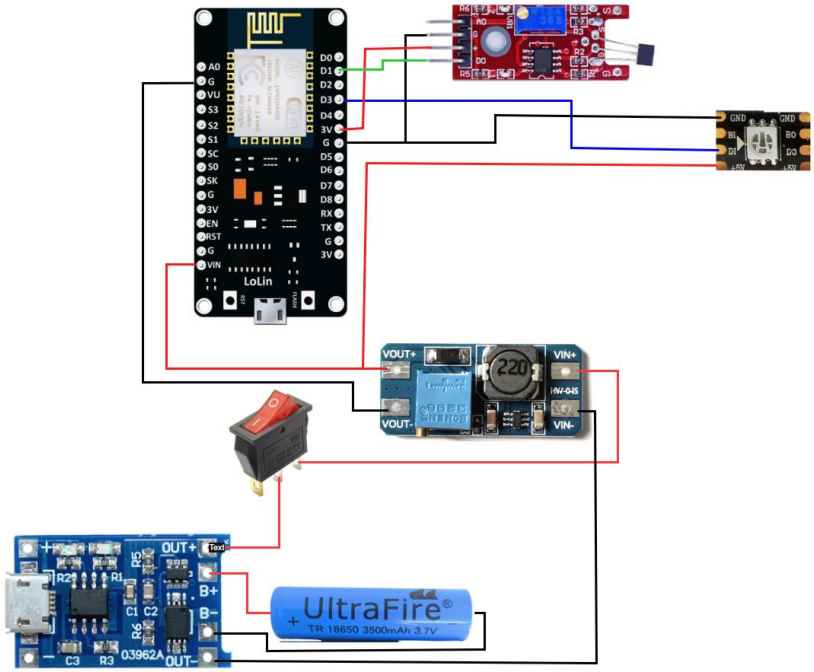


Рисунок 3.6 – Функціональна схема

# 4 ПРИКЛАДНЕ ПРОГАМНЕ ЗАБЕЗБЕЧЕННЯ ESP8266

## 4.1 Бібліотеки, макроси та глобальні змінні

У вихідному коду для мікроконтролера використовуються такі бібліотеки:

– ESP8266WiFi.h: забезпечує функціональність Wi-Fi для створення точки доступу та комунікації;

– ESP8266WebServer.h: дозволяє створити веб-сервер для обробки HTTP-запитів від мобільного додатку;

– Adafruit_NeoPixel.h: надає функції для керування RGB-світлодіодом WS2813B;

– EEPROM.h: Забезпечує доступ до енергонезалежної пам'яті для зберігання налаштувань.

Для правильного функціонування на початку файлу об’явленні глобальні змінні та макроси (лістинг 4.1).

Лістинг 4.1 – Макроси та глоабльні змінні

```
#define LED_PIN    D3     // Пін для підключення світлодіода
#define NUM_LEDS   1      // Кількість світлодіодів у стрічці
#define BRIGHTNESS 70     // Стандартна яскравість світлодіода
#define MAX_BRIGHTNESS 180 // Максимальна яскравість світлодіода
#define SENSOR_PIN D1     // Пін для підключення датчика Холла
#define TIMEOUT 3000       // Таймаут для оновлення швидкості (3 секунди)
#define BRAKE_DURATION 2000 // Тривалість індикації гальмування (2 секунди)
#define BRAKE_THRESHOLD 2.0 // Поріг швидкості для визначення гальмування (км/год)
#define EEPROM_ADDR_WHEEL_DIAMETER 0 // Адреса в EEPROM для зберігання діаметра колеса

// Ініціалізація об'єкта для керування світлодіодом
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, LED_PIN, NEO_GRB + NEO_KHZ800);
```

```
const char *ssid = "Speedometer";     // Ім'я Wi-Fi мережі
const char *password = "00000000";    // Пароль Wi-Fi мережі

ESP8266WebServer server(80);           // Створення об'єкта веб-
сервера на порту 80

volatile unsigned long lastTurn = 0;   // Час останнього оберту
колеса
volatile float speed = 0;              // Поточна швидкість
volatile float distance = 0;           // Пройдена відстань
unsigned long lastSpeedUpdate = 0;        // Час останнього
оновлення швидкості

float wheelDiameter = 29.0;               // Діаметр колеса за
замовчуванням (29 дюймів)
float wheelLength = 0.0;                  // Довжина окружності
колеса (розраховується пізніше)

// Визначення режимів роботи світлодіода
enum LedMode {
  NORMAL,          // Звичайний режим
  BLINK_INTERVAL, // Режим мигання
  FADE,            // Режим згасання
  STROBE           // Режим стробоскопа
};

LedMode currentMode = NORMAL;          // Поточний режим роботи
світлодіода
unsigned long lastLedUpdate = 0;          // Час останнього
оновлення стану світлодіода
int ledState = 0;                          // Поточний стан
світлодіода (увімкнено/вимкнено)
int fadeDirection = 1;                     // Напрямок зміни
яскравості в режимі FADE
int currentBrightness = BRIGHTNESS;     // Поточна яскравість
світлодіода
```

## 4.2 Головні функції

### 4.2.1 Функія «void ICACHE_RAM_ATTR sensorISR»

Лістинг 4.2 – Функія «void ICACHE_RAM_ATTR sensorISR»

```
// Обробник переривання для датчика Холла
void ICACHE_RAM_ATTR sensorISR() {
    unsigned long currentTime = millis();
    if (currentTime - lastTurn > 80) { // Захист від брязкоту
```

```
контактів
        float timeDiff = (float)(currentTime - lastTurn) / 
1000.0;
        speed = (wheelLength / timeDiff) * 3.6; // Розрахунок 
швидкості в км/год
        distance += wheelLength / 1000.0; // Оновлення пройденої 
відстані в км
        lastTurn = currentTime;
        lastSpeedUpdate = currentTime;
    }
}
```

### 4.2.2 Функія «void updateWheelLength»

Лістинг 4.3 – Функія «void updateWheelLength» (файл speedometer.ino)

```
void updateWheelLength() {
    // Розрахунок довжини окружності колеса
    wheelLength = wheelDiameter * 0.0254 * 3.14159; // 
Перетворення дюймів у метри і множення на пі
}
```

### 4.2.3 Функія «void updateSpeed»

Лістинг 4.4 – Функія «void updateSpeed»

```
// Функція оновлення швидкості
void updateSpeed() {
    unsigned long currentTime = millis();
    if (currentTime - lastSpeedUpdate > TIMEOUT) {
        speed = 0; // Скидання швидкості до нуля, якщо датчик не 
спрацьовував довгий час
    }
}
```

### 4.2.4 Функія «void handleSetMode»

Лістинг 4.5 – Функія «void handleSetMode» (файл speedometer.ino)

```
/ Функція для встановлення режиму роботи світлодіода
void handleSetMode() {
    if (server.hasArg("mode")) {
```

```
        int mode = server.arg("mode").toInt();
        if (mode >= 0 && mode <= 3) {
            // Встановлюємо новий режим, якщо він в допустимому
діапазоні
            currentMode = static_cast<LedMode>(mode);
            server.send(200,  "text/plain",  "Mode  set  to  "  +
String(currentMode));
        } else {
            // Відправляємо помилку, якщо режим недійсний
            server.send(400, "text/plain", "Invalid mode");
        }
    } else {
        // Відправляємо помилку, якщо параметр mode відсутній
        server.send(400, "text/plain", "Bad Request");
    }
}
```

### 4.2.5 Функія «void handleSetWheelDiameter»

Лістинг 4.6 – Функія «void handleSetWheelDiameter»

```
/ Функція для встановлення діаметра колеса
void handleSetWheelDiameter() {
    if (server.hasArg("diameter")) {
        float diameter = diameterStr.toFloat();
        if (diameter > 0) {
            // Встановлюємо новий діаметр колеса
            wheelDiameter = diameter;
            // Зберігаємо нове значення в EEPROM
            EEPROM.put(EEPROM_ADDR_WHEEL_DIAMETER,
wheelDiameter);
            EEPROM.commit();
            updateWheelLength();
            // Відправляємо підтвердження з новим діаметром
            server.send(200,                  "application/json",
"{\"diameter\":" + String(wheelDiameter, 2) + "}");
        } else {
            // Відправляємо помилку, якщо діаметр недійсний
            server.send(400,                  "application/json",
"{\"error\":\"Invalid diameter value\"}");
        }
    } else {
        //   Відправляємо   помилку,   якщо   параметр   diameter
відсутній
        server.send(400,   "application/json",   "{\"error\":\"Bad
Request\"}");
    }}
```

### 4.2.6 Функія «void updateLED»

Лістинг 4.7 – Функія «void updateLED»

```
/ Функція для оновлення стану світлодіода
void updateLED() {
    unsigned long currentTime = millis();
    static unsigned long brakeStartTime = 0;
    static float lastSpeed = 0;

    // Перевірка на гальмування
    if (speed < lastSpeed - BRAKE_THRESHOLD) {
        brakeStartTime = currentTime;
    }
    // Оновлення кольору світлодіода
    if (currentTime - brakeStartTime < BRAKE_DURATION) {
        // Яскравий червоний при гальмуванні
        strip.setPixelColor(0, strip.Color(MAX_BRIGHTNESS+70, 0,
0));
    } else {
        switch (currentMode) {
//…. Оброка режимів міготіння світлодіода
        }
    }
    // Оновлюємо стан світлодіода
    strip.show();
    lastSpeed = speed;}
```

### 4.2.7 Функія «void setup»

Функція setup() відповідає за ініціалізацію всіх компонентів системи велосипедного комп'ютера.

Лістинг 4.8 – Функія «void setup»

```
void setup() {
  // Ініціалізація EEPROM для зберігання налаштувань
  EEPROM.begin(512);

  // Зчитування діаметра колеса з EEPROM
  EEPROM.get(EEPROM_ADDR_WHEEL_DIAMETER, wheelDiameter);
  if (isnan(wheelDiameter) || wheelDiameter <= 0) {
    wheelDiameter = 29.0; // Значення за замовчуванням, якщо
EEPROM порожній або містить недійсне значення
```

```
    EEPROM.put(EEPROM_ADDR_WHEEL_DIAMETER, wheelDiameter);
    EEPROM.commit();
  }

  // Налаштування точки доступу Wi-Fi
  WiFi.softAP(ssid, password);

  // Налаштування маршрутів веб-сервера
  server.on("/speed", handleSpeed);
  server.on("/distance", handleDistance);
  server.on("/ledmode", handleLedMode);
  server.on("/setmode", HTTP_GET, handleSetMode);
  server.on("/setWheelDiameter",                          HTTP_GET,
handleSetWheelDiameter);
  server.begin();

  // Налаштування піна датчика та переривання
  pinMode(SENSOR_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(SENSOR_PIN),   sensorISR,
RISING);

  // Ініціалізація світлодіодної стрічки
  strip.begin();
  strip.setBrightness(MAX_BRIGHTNESS);
  strip.setPixelColor(0, strip.Color(MAX_BRIGHTNESS, 0, 0));
  strip.show();

  // Встановлення початкового режиму світлодіода
  currentMode = STROBE; // Починаємо з режиму стробоскопа

  // Ініціалізація довжини кола колеса
  updateWheelLength();
}
```

### 4.2.8 Функія «void loop»

Лістинг 4.9 – Функія «void loop»

```
void loop() {
  // Обробка запитів клієнтів веб-сервера
  server.handleClient();
  // Оновлення поточної швидкості
  updateSpeed();
  updateLED();  // Оновлення стану світлодіода
  delay(10);  // Коротка затримка для стабільності роботи
}
```

# 5 ПРИКЛАДНЕ ПРОГРАММЕ ЗАБЕЗПЕЧЕННЯ REACT NATIVE ЗАСТОСУНОК

## 5.1 Структура застосунку

Програмно-апаратний комплекс аналізу швидкісних характеристик велосипеда на базі ESP8266 включає мобільний застосунок, розроблений з використанням React Native. Застосунок має наступну структуру:

а) навігаційна система (_layout.tsx):

1) реалізована з використанням expo-router та компонента Tabs;

2) містить три основні вкладки: "Головна", "Поїздки" та "Налаштування";

3) забезпечує інтуїтивно зрозумілу навігацію з нижньою панеллю вкладок;

4) використовує іконки з бібліотеки Ionicons для візуального оформлення;

Лістинг 5.1 – Спрощена структура навігаційної системи застосунку (файл _layout.tsx)

```
export default function RootLayout() {
  return (
    <SafeAreaView style={styles.container}>
      <Tabs
        screenOptions={{
          tabBarStyle: {
            backgroundColor: '#234B6B',
          },
          tabBarActiveTintColor: '#7DD3E3',
          tabBarInactiveTintColor: '#ffffff',
          headerShown: false,
        }}
      >
        <Tabs.Screen
          name="index"
          options={{
            title: 'Головна',
            tabBarIcon: ({ color, size }) => (
```

```
              <Ionicons name="home" color={color} size={size} />
            ),
          }}
        />
        {/* Інші вкладки */}
      </Tabs>
    </SafeAreaView>
  );
}
```

б) головний екран спідометра(index.tsx):

1) відображає поточну швидкість у вигляді циферблата з індикатором прогресу;

2) показує статистику поточної поїздки (середня швидкість, відстань, максимальна швидкість);

3) дозволяє починати та закінчувати запис поїздки;

4) зберігає дані поїздок в AsyncStorage для подальшого аналізу;

Лістинг 5.2 – Компонент циферблата спідометра (файл index.tsx)

```
     const SpeedometerCircle: React.FC<SpeedometerCircleProps> =
({ speed, maxValue = 50 }) => {
  const progress = (speed / maxValue) * 100;

  return (
    <View style={styles.circleContainer}>
      <View style={styles.circle}>
        <Text style={styles.speedText}>{speed.toFixed(1)}</Text>
        <Text style={styles.unitText}>КМ/Г</Text>
      </View>
      <View style={styles.progressRing}>
        <View style={[styles.progressIndicator, { transform: [{
rotate: `${progress * 3.6}deg` }] }]} />
      </View>
    </View>
  );
};
```

в) екран історії поїздок (trips.tsx):

1) відображає список збережених поїздок з можливістю розгортання деталей;

2) показує дату, час початку та закінчення, відстань, середню та

максимальну швидкість;

3) дозволяє видаляти записи поїздок;

4) використовує useFocusEffect [6] для оновлення даних при переході на екран;

Лістинг 5.3 – Використання useFocusEffect для оновлення даних (файл trips.tsx)

```
useFocusEffect(
  useCallback(() => {
    loadTrips();
  }, [])
);
```

г) екран налаштувань (settings.tsx):

1) дозволяє налаштувати діаметр колеса через випадаючий список;

2) відображає інформацію про Wi-Fi підключення до пристрою ESP8266;

3) забезпечує вибір режиму роботи світлодіода (звичайний, миготіння, згасання, стробоскоп);

4) відправляє налаштування на пристрій ESP8266 через HTTP запити.

Лістинг 5.4 – Відображення інформації про Wi-Fi підключення (файл settings.tsx)

```
<View style={styles.infoContainer}>
  <Text style={styles.infoLabel}>Ім'я мережі(SSID):</Text>
  <Text style={styles.infoValue}>{wifiInfo.ssid}</Text>
</View>
<View style={styles.infoContainer}>
  <Text style={styles.infoLabel}>Пароль:</Text>
  <Text style={styles.infoValue}>{wifiInfo.password}</Text>
</View>
<View style={styles.infoContainer}>
  <Text style={styles.infoLabel}>IP адреса:</Text>
  <Text style={styles.infoValue}>{wifiInfo.ip}</Text>
```

```
</View>
```

## 5.2 Основні компоненти та функціональність

### 5.2.1 Екран спідометра

Екран спідометра (index.tsx) є центральним елементом застосунку та містить наступні компоненти:

а) SpeedometerCircle:

1) візуалізує поточну швидкість у вигляді кругового індикатора;

2) динамічно відображає прогрес швидкості відносно максимального значення;

3) показує числове значення швидкості та одиницю виміру (КМ/Г);

4) навколо основного кола є кільце прогресу, яке заповнюється відповідно до поточної швидкості;

б) статистика поїздки:

1) відображає середню швидкість, пройдену відстань та максимальну швидкість;

2) оновлюється в реальному часі під час активної поїздки;

Лістинг 5.6 – Відображення статистики поїздки (файл index.tsx)

```
<View style={styles.statsContainer}>
  <Text        style={styles.statsText}>Середня        швидкість:
{stats.avgSpeed.toFixed(2)} км/г</Text>
  <Text    style={styles.statsText}>Відстань:    {(distance    -
tripStartDistance).toFixed(2)} км</Text>
  <Text      style={styles.statsText}>Максимальна      швидкість:
{stats.maxSpeed.toFixed(2)} км/г</Text>
</View>
```

в) керування поїздкою:

1) кнопка "Почати/Закінчити поїздку" для керування записом даних;

2) функція startTrip() для ініціалізації нової поїздки;

3) функція endTrip() для завершення та збереження даних поїздки;

Лістинг 5.7 – Функція ініціалізації нової поїздки (файл index.tsx)

```
const startTrip = () => {
  const newTrip: TripData = {
    id: Date.now().toString(),
    startTime: new Date().toLocaleTimeString(),
    distance: 0,
    averageSpeed: 0,
    maxSpeed: 0
  };

  setIsTrip(true);
  setTripStartTime(new Date());
  speedHistoryRef.current = [];
  setStats({ avgSpeed: 0, maxSpeed: 0 });
  setCurrentTrip(newTrip);
  setTripStartDistance(distance);
};
```

Лістинг 5.8 – Функція завершення та збереження поїздки (файл index.tsx)

```
const endTrip = async () => {
  if (!tripStartTime || !currentTrip) return;

  const endTime = new Date();
  const tripDistance = distance - tripStartDistance;
  const updatedTrip: TripData = {
    ...currentTrip,
    date: tripStartTime.toLocaleDateString(),
    endTime: endTime.toLocaleTimeString(),
    distance: tripDistance,
    averageSpeed: stats.avgSpeed,
    maxSpeed: stats.maxSpeed,
  };

  try {
    const tripsJson = await AsyncStorage.getItem("trips");
    const trips: TripData[] = tripsJson ? JSON.parse(tripsJson)
: [];
    trips.unshift(updatedTrip);
    await AsyncStorage.setItem("trips", JSON.stringify(trips));
    console.log("Trip saved:", updatedTrip);
    setSaveMessage("Поїздку успішно збережено!");
    setTimeout(() => setSaveMessage(null), 3000);
```

```
  } catch (error) {
    console.error("Error saving trip:", error);
    setSaveMessage("Помилка при збереженні поїздки");
    setTimeout(() => setSaveMessage(null), 3000);
  }

  setIsTrip(false);
  setTripStartTime(null);
  setCurrentTrip(null);
};
```

г) отримання даних:

1) функція fetchData() для отримання даних від ESP8266 через HTTP запити;

2) періодичне оновлення даних з інтервалом в 1 секунду;

3) обробка помилок при втраті зв'язку з пристроєм.

Лістинг 5.9 – Функція отримання даних від ESP8266 (файл index.tsx)

```
const fetchData = async () => {
  try {
    const speedResponse = await
fetch("http://192.168.4.1/speed");
    const speedText = await speedResponse.text();
    const currentSpeed = Number.parseFloat(speedText);
    setSpeed(currentSpeed);

    const distanceResponse = await
fetch("http://192.168.4.1/distance");
    const distanceText = await distanceResponse.text();
    const currentDistance = Number.parseFloat(distanceText);
    setDistance(currentDistance);

    if (isTrip) {
      updateStats(currentSpeed);
    }
  } catch (error) {
    console.error("Error fetching data:", error);
  }
};
```

Лістинг 5.10 – Налаштування періодичного оновлення даних (файл index.tsx)

```
useEffect(() => {
  let intervalId: NodeJS.Timeout;

  if (isFocused) {
    fetchData();
    intervalId = setInterval(fetchData, 1000);
  }

  return () => {
    if (intervalId) {
      clearInterval(intervalId);
    }
  };
}, [isFocused, isTrip]);
```

### 5.2.2 Екран налаштувань

Екран налаштувань (settings.tsx) забезпечує конфігурацію програмно-апаратного комплексу та містить наступні компоненти:

а) налаштування діаметра колеса:

1) використовує компонент DropDownPicker для вибору розміру колеса;

2) відправляє вибраний діаметр на ESP8266 через HTTP запит;

3) обробляє відповідь від пристрою та відображає повідомлення про помилки;

Лістинг 5.11 – Компонент вибору діаметра колеса (файл settings.tsx)

```
<View style={styles.section}>
  <Text style={styles.sectionTitle}>Параметри колеса</Text>
  <Text style={styles.label}>Діаметр колеса:</Text>
  <View style={styles.dropdownWrapper}>
    <DropDownPicker
      open={open}
      value={null}
      items={items}
      setOpen={setOpen}
      setValue={(value) => {
```

```
        if (typeof value === 'function') {
          const newValue = value(wheelSize);
          if (typeof newValue === 'number') {
            setWheelDiameter(newValue);
          }
        } else if (typeof value === 'number') {
          setWheelDiameter(value);
        }
      }}
      setItems={setItems}
      placeholder={"Оберіть діаметр"}
      style={styles.picker}
      dropDownContainerStyle={styles.dropDownContainer}
      textStyle={styles.dropdownText}
      labelStyle={styles.dropdownLabel}
      listItemLabelStyle={styles.dropdownItemLabel}
      theme="DARK"
      zIndex={3000}
      zIndexInverse={1000}
    />
  </View>
</View>
```

Лістинг 5.12 – Функція встановлення діаметра колеса (файл settings.tsx)

```
const setWheelDiameter = async (diameter: number) => {
  try {
    const response = await
fetch(`http://${wifiInfo.ip}/setWheelDiameter?diameter=${diamete
r}`);
    if (response.ok) {
      const text = await response.text();
      const newDiameter = parseFloat(text);
      console.log(`Wheel diameter set to ${newDiameter}`);
      setWheelSize(newDiameter);
    } else {
      throw new Error('Failed to set wheel diameter');
    }
  } catch (error) {
    console.error('Error setting wheel diameter:', error);
    Alert.alert('Помилка', 'Не вдалося встановити діаметр колеса
(Приєднайтесь до пристрою)');
    setWheelSize(prevSize => prevSize);
  }
};
```

б) інформація про Wi-Fi підключення:

1) відображає SSID, пароль та IP-адресу для підключення до

пристрою;

2) забезпечує користувача необхідною інформацією для налаштування з'єднання;

Лістинг 5.13 – Конфігурація Wi-Fi підключення (файл settings.tsx)

```
const wifiInfo: WifiInfo = {
  ssid: "Speedometer",
  password: "00000000",
  ip: "192.168.4.1"
};
```

в) керування режимом світлодіода:

1) дозволяє вибрати один з чотирьох режимів роботи світлодіода;

2) відправляє вибраний режим на ESP8266 через HTTP запит;

3) візуально відображає активний режим через стилізацію кнопок.

Лістинг 5.14 – Компонент вибору режиму світлодіода (файл settings.tsx)

```
<View style={[styles.section, { zIndex: open ? -1 : 1 }]}>
  <Text style={styles.sectionTitle}> Режим світлодіоду</Text>
  <TouchableOpacity
    style={[styles.button,       ledMode       ===       0       &&
styles.buttonActive]}
    onPress={() => changeLedMode(0)}
  >
    <Text style={styles.buttonText}>Звичайний</Text>
  </TouchableOpacity>
  <TouchableOpacity
    style={[styles.button,       ledMode       ===       1       &&
styles.buttonActive]}
    onPress={() => changeLedMode(1)}
  >
    <Text style={styles.buttonText}>Миготіння</Text>
  </TouchableOpacity>
  <TouchableOpacity
    style={[styles.button,       ledMode       ===       2       &&
styles.buttonActive]}
    onPress={() => changeLedMode(2)}
  >
    <Text style={styles.buttonText}>Згасання</Text>
  </TouchableOpacity>
```

```
  <TouchableOpacity
    style={[styles.button,       ledMode       ===       3       &&
styles.buttonActive]}
    onPress={() => changeLedMode(3)}
  >
    <Text style={styles.buttonText}>Стробоскоп</Text>
  </TouchableOpacity>
</View>
```

Лістинг 5.15 – Функція зміни режиму світлодіода (файл settings.tsx)

```
const changeLedMode = async (mode: number) => {
  try {
    const              response              =              await
fetch(`http://${wifiInfo.ip}/setmode?mode=${mode}`);
    if (response.ok) {
      setLedMode(mode);
    } else {
      throw new Error('Failed to change LED mode');}
  } catch (error) {
    console.error('Error changing LED mode:', error);
    Alert.alert('Помилка', 'Не вдалося змінити режим світлодіоду
(Приєднайтесь до пристрою)');
  }
};
```

### 5.2.3 Екран історії поїздок

Екран історії поїздок (trips.tsx) відповідає за відображення списку збережених поїздок користувача. Компонент використовує хуки React (useState, useCallback) для управління станом та ефектами. Основна функціональність включає завантаження поїздок з AsyncStorage при фокусуванні на екрані, відображення списку поїздок з можливістю розгортання деталей кожної поїздки, та видалення окремих поїздок.

Лістинг 5.16 – Основна структура компонента TripsScreen (файл trips.tsx)

```
const TripsScreen: React.FC = () => {
  const [trips, setTrips] = useState<Trip[]>([]);
  const  [expandedTrip,  setExpandedTrip]  =  useState<string  |
null>(null);
```

```
  useFocusEffect(
    useCallback(() => {
      loadTrips();
    }, [])
  );

  // Інші функції компонента...
};
```

Інтерфейс компонента побудований з використанням FlatList для ефективного рендерингу списку поїздок. Кожен елемент списку представлений як розгортаний блок, який показує дату та час початку поїздки в згорнутому стані, а при розгортанні відображає додаткову інформацію, таку як час закінчення, пройдена відстань, середня та максимальна швидкість. Компонент також включає функціональність видалення поїздок з можливістю оновлення AsyncStorage після видалення.

Лістинг 5.17 – Відображення елемента поїздки у списку (файл trips.tsx)

```
const renderTripItem = ({ item }: { item: Trip }) => (
  <View style={styles.tripItem}>
    <TouchableOpacity
      style={styles.tripHeader}
      onPress={() => setExpandedTrip(expandedTrip === item.id ?
null : item.id)}
    >
      <Text                    style={styles.tripDate}>{item.date}
{item.startTime}</Text>
      <Ionicons
        name={expandedTrip   ===   item.id   ?   "chevron-up"   :
"chevron-down"}
        color="white"
        size={24}
      />
    </TouchableOpacity>
    {expandedTrip === item.id && (
      <View style={styles.tripDetails}>
        <Text       style={styles.tripInfo}>Час       закінчення:
{item.endTime}</Text>
        <Text                    style={styles.tripInfo}>Відстань:
{item.distance.toFixed(2)} км</Text>
        <Text      style={styles.tripInfo}>Середня      швидкість:
{item.averageSpeed.toFixed(2)} км/г</Text>
        <Text    style={styles.tripInfo}>Максимальна    швидкість:
{item.maxSpeed.toFixed(2)} км/г</Text>
```

```
        <TouchableOpacity             style={styles.deleteButton}
onPress={() => deleteTrip(item.id)}>
          <Ionicons name="trash-outline" color="white" size={24}
/>
        </TouchableOpacity>
      </View>
    )}
  </View>
);
```

Лістинг 5.18 – Функція завантаження збережених поїздок (файл trips.tsx)

```
const loadTrips = async () => {
  try {
    const tripsJson = await AsyncStorage.getItem('trips');
    if (tripsJson) {
      const parsedTrips = JSON.parse(tripsJson);
      setTrips(parsedTrips);
      console.log('Trips loaded:', parsedTrips);
    } else {
      console.log('No trips found');
    }
  } catch (error) {
    console.error('Error loading trips:', error);
  }
};
```

Лістинг 5.19 – Функція видалення поїздки (файл trips.tsx)

```
const deleteTrip = async (id: string) => {
  const updatedTrips = trips.filter(trip => trip.id !== id);
  setTrips(updatedTrips);
  try {
    await                          AsyncStorage.setItem('trips',
JSON.stringify(updatedTrips));
  } catch (error) {
    console.error('Error saving trips:', error);
  }
};
```

# 6 ІНСТРУКЦІЯ КОРИСТУВАЧА

Для початку потрібно закріпити пристрій та датчик на велосипед, і встановити мобільними застосунок «BicycleSpeedometer.apk» (рисунок 6.1).

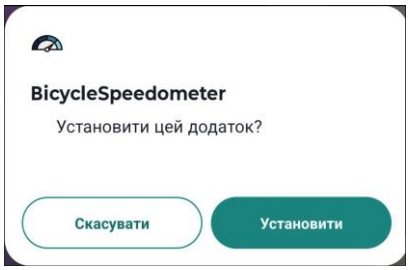


Рисунок 6.1 – Встановлення додатку

Наступний крок полгає у влюченні живлення пристрою (рисунок 6.2). та підкючення до мережі «Speedometer» (рисунок 6.3).

Рисунок 6.2 – Влюченнія живлення пристрою

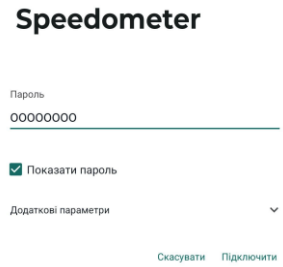


Рисунок 6.3 – Підключення до мережі

На далі, потрібно зайти в застосунок та налаштувати під себе діаметр колеса та обрати режим роботи світлодіода (рисунок 6.4).

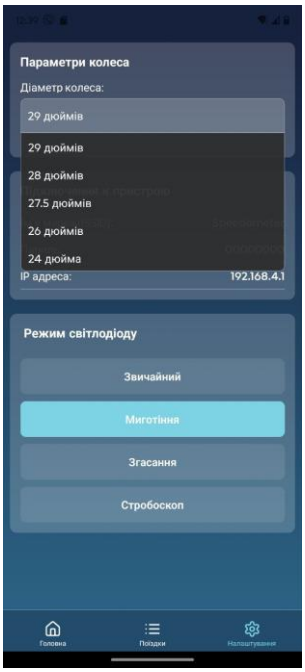


Рисунок 6.4 – Екран налаштування

Перейти на головний екран, та натиснути на кнопку «Почати поїздку» (рисунок 6.5), та почати рух на велосипеді. На екрані буде зображенні показники швидкості, та доступна функція закінчити поїдку (рисунок 6.6).

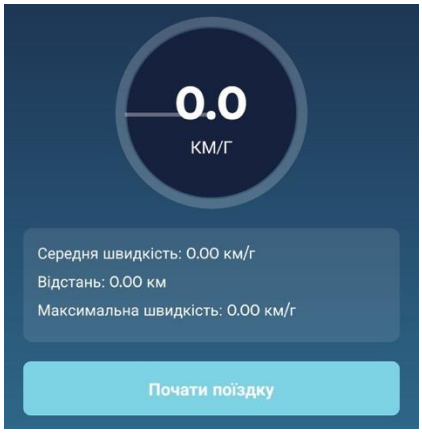


Рисунок 6.5 – Кнопка «Почати поїздку»

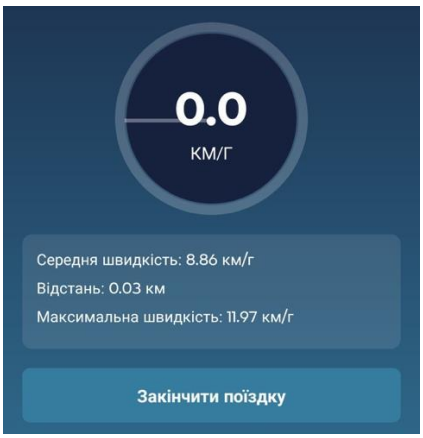


Рисунок 6.6 – Кнопка «Закінчити поїздку»

Після закінчення поїздки з’явиться повідомлення про збереження поїдки (рисунок 6.7), та можна переглянути дані на екрані «Поїдки» (рисунок 6.8).

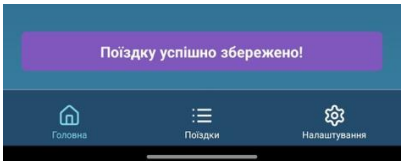


Рисунок 6.7 – Повідомлення про збереження поїдки

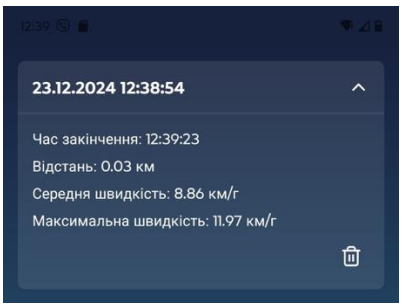


Рисунок 6.8 – Збережена поїдка на екрані «Поїдки»

Після користування потрібно не забути вимкнути живлення, та якщо є необхідність , то поставити заряджатись.

# 7 ТЕХНІЧНІ ВИМОГИ ТА ТЕСТУВАННЯ

В ході реалізації кваліфікаційної роботи з розробки програмно-апаратного комплексу аналізу швидкісних характеристик велосипеда на базі ESP8266 код зазнав значних змін та удосконалень. Це пов'язано з тим, що на різних етапах проектування і тестування виникали помилки та неочікувані ситуації, які потребували вирішення.

При початковому тестуванні на макетній платі виникли проблеми з стабільністю Wi-Fi з'єднання та точністю аналізу швидкісних характеристик. Ці проблеми були вирішені шляхом оптимізації коду та покращення алгоритму аналізу сигналів від датчика Холла.

Таким чином, в процесі реалізації програмно-апаратного комплексу виникли певні труднощі, але завдяки ітеративному підходу до розробки, тестуванню в різних умовах та постійному вдосконаленню програмної та апаратної частин вдалося отримати надійний та ефективний програмно-апаратний комплекс аналізу швидкісних характеристик велосипеда на базі ESP8266.

Технічні вимоги:

а) для мобільного додатку: операційна система: Android 6.0+ або iOS 11+; оперативна пам'ять: від 2 ГБ; вільне місце для встановлення: від 80 МБ; наявність Wi-Fi модуля для з'єднання з комплексом;

б) до з'єднання: стабільне з'єднання в радіусі до 10 метрів між велосипедним комп'ютером та смартфоном;

в) умови експлуатації: здатність витримувати типові вібрації при їзді на велосипеді.

# ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено програмно-апаратний комплекс аналізу швидкісних характеристик велосипеда на базі ESP8266 з мобільним додатком. Поставлена задача була реалізована в повному обсязі. Створено функціональний програмно-апаратний комплекс, який аналізує швидкість, відстань та інші параметри поїздки, а також розроблено мобільний застосунок для взаємодії з пристроєм та відображення даних. Розроблено апаратну частину на базі ESP8266 з датчиком Холла та світлодіодним індикатором, створено прошивку для ESP8266 та мобільний застосунок для відображення та аналізу даних поїздок.

Незважаючи на досягнуті результати, робота має певні недоліки. Відсутність належного корпусу та недостатній рівень вологозахищеності обмежують використання комплексу в різних погодних умовах. Відсутність індикації заряду акумулятора ускладнює контроль за енергоспоживанням пристрою.

Для вдосконалення програмно-апаратного комплексу можна запропонувати розробку водонепроникного корпусу, впровадження системи моніторингу заряду акумулятора, розширення функціоналу мобільного додатку, включаючи можливість ділитися результатами аналізу поїздок. Додавання нових режимів роботи світлодіодного індикатора, інтеграція з GPS та реалізація функції автоматичного калібрування для різних розмірів коліс також підвищать функціональність та універсальність комплексу.

Загалом, розроблений програмно-апаратний комплекс аналізу швидкісних характеристик велосипеда на базі ESP8266 демонструє високу функціональність та відповідає поставленим вимогам. Комплекс має значний потенціал для подальшого вдосконалення, що дозволить зробити його більш універсальним та привабливим для широкого кола користувачів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Данилюк, О. В. Безпека велосипедистів: технічні рішення та інноваційні підходи [Текст] : монографія / О. В. Данилюк ; Одеський нац. політехн. ун-т. – Одеса : Транспортні технології, 2022. – 198 с. – Бібліогр.: с. 185–198.

2) Hahanova A. Vector-Deductive Faults-as-Address Simulation / A. Hahanova // International Journal of Computing. – 2023. –22(3). – P. 328–334.

3) Шевченко, В. Л. Розробка мобільних додатків для спортивних систем моніторингу [Текст] / В. Л. Шевченко, Т. П. Іванова. – Львів : Мобільні технології, 2023. – 284 с. – (Серія "Інформаційні технології").

4) ESP8266 Documentation [Електронний ресурс] : технічна документація / Espressif Systems. – Режим доступу: [https://www.espressif.com/en/products/socs/esp8266](https://www.espressif.com/en/products/socs/esp8266)

5) Schwartz, M. Internet of Things with ESP8266 [Text] / M. Schwartz. – Birmingham : Packt Publishing, 2019. – 246 p.

6) Brown, E. React Native: Building Mobile Apps with JavaScript [Text] / E. Brown. – Birmingham : Packt Publishing, 2018. – 372 p.

7) Vector-logic fault simulation / V. I. Hahanov, S. V. Chumachenko, E. I. Lytvynova, H. V. Khakhanova, I. V. Hahanov, T. G. Rozhnova, V. I. Obrizan // Радіоелектроніка, інформатика, управління. – Запоріжжя: ЗНТУ. – 2024. – №4 (71). – С.185-194.

8) Faults-as-address simulation / [V. Hahanov, S. Chumachenko, E. Litvinova et al.] // IAES International Journal of Robotics and Automation (IJRA). – 2024. – Vol. 13, № 4. – P. 452–468.

9) Бондаренко, І. В. Сучасні системи моніторингу спортивних показників: технології та перспективи розвитку [Текст] / І. В. Бондаренко, О. С. Петров. – Київ : Технічна література, 2023. – 256 с.

10) Коваленко, М. Ю. Мікроконтролери ESP8266/ESP32 у проектах

Інтернету речей [Текст] : навч. посіб. / М. Ю. Коваленко. – Харків : Електронні системи, 2022. – 320 с. – Бібліогр.: с. 315–320.