Галицький фаховий коледж імені В'ячеслава Чорновола
відділення комп'ютерних технологій
циклова комісія інформатики та комп'ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач відділення
комп'ютерних технологій
Наталія СТЕФУРАК / _____/
підпис
«__»_____2024р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи
освітньо-професійного ступеня «фаховий молодший бакалавр»
зі спеціальності 123 «Комп'ютерна інженерія»
на тему:
«Система дистанційного моніторингу показників стану автомобіля»

Студент групи КІ-41	Богдан ГРИЦАЮК	_____ (підпис)
Керівник роботи	Оксана СИРОТЮК	_____ (підпис)
Консультанти: з техніко-економічного обґрунтування	Любов МЕЛЕНЧУК	_____ (підпис)
Нормоконтролер	Ольга СЛЄПЦОВА	_____ (підпис)

Тернопіль – 2024

Галицький фаховий коледж імені В'ячеслава Чорновола

відділення комп'ютерних технологій

циклова комісія інформатики та комп'ютерних дисциплін

ЗАТВЕРДЖУЮ

Завідувач відділенням

комп'ютерних технологій

Наталія СТЕФУРАК /_____/

підпис

«____»_____2024р.

ЗАВДАННЯ

на кваліфікаційну роботу

на здобуття освітньо-кваліфікаційного рівня «фаховий молодший бакалавр»

студенту Грицаюку Богдану Леонідовичу

(прізвище, ім'я та по-батькові студента)

1. Тема кваліфікаційної роботи: Система дистанційного моніторингу показників стану автомобіля затверджено наказом по коледжу Від "__" ____2024р., №_____

2. Термін здачі студентом завершеної роботи "__" _____2024р.

3. Вихідні дані до роботи *Інформація про показники стану автомобіля та технологія їх обробки*

4. Перелік питань, які повинні бути розроблені в роботі:

а) основна частина *1 Аналіз існуючих рішень та постановка завдання; 2 Проєктування системи; 3 Реалізація та тестування системи*

б) техніко-економічне обґрунтування *1 Аналіз ринку збуту продукту чи послуги. 2 Розрахунок витрат на проектування. 3 Обґрунтування необхідності розробки*

5. Перелік графічного матеріалу *1 Структурна схема. 2 Діаграма послідовності. 3 Схема взаємодії користувача з системою. 4.Функціональна схема. 5. Електрична схема.*

6. Консультанти роботи:

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	_____ _____		

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1	Вибір теми, ознайомлення з вимогами до кваліфікаційної роботи.	23.11.2023	01.12.2023
2	Дослідження предметної області, огляд типових рішень.	02.12.2023	05.02.2024
3	Дослідження технологій реалізації.	29.01.2024	07.02.2024
4	Розробка функціональних вимог до проекту та робота над структурою системи.	08.02.2024	01.03.2024
5	Проектування системи та підготовка відповідного графічного матеріалу.	03.03.2024	05.04.2024
6	Встановлення та налаштування середовища реалізації.	18.03.2024	08.04.2024
7	Реалізація системи та написання відповідного розділу.	09.04.2024	09.05.2024
8	Доопрацювання апаратної складової.	10.05.2024	16.05.2024
9	Опрацювання економічного розділу кваліфікаційної роботи та оформлення спеціального розділу.	11.03.2024	03.05.2024
10	Тестування системи та усунення недоліків.	17.05.2024	31.05.2024
11	Робота над оформленням пояснюваної записки.	01.06.2024	18.06.2024
12	Попередній захист кваліфікаційної роботи, доопрацювання.	18.06.2024	18.06.2024
13	Підготовка до захисту кваліфікаційної роботи.	19.06.2024	27.06.2024
14	Захист кваліфікаційної роботи.	28.06.2024	28.06.2024

Дата видачі “27” листопада 2024 р. Керівник _____/ *Оксана СИРОТЮК*
Завдання прийняв до виконання _____/ *Богдан ГРИЦАЮК*

Реферат

Кваліфікаційна робота. Система дистанційного моніторингу показників стану автомобіля. 54с., 20 рисунків, 10 додатків, 6 джерел.

Об'єкт дослідження – система дистанційного моніторингу показників стану автомобіля.

Метою роботи є розробка системи дистанційного моніторингу, яка дозволить відстежувати основні показники стану автомобіля в реальному часі, використовуючи мікроконтролер Arduino Mega та модулі MCP251 і ESP8266.

Система повинна забезпечити збір даних з CAN шини, таких як швидкість, оберти двигуна, рівень палива, температура двигуна, та інші параметри. Дані передаватимуться на веб сторінку, де вони будуть зберігатися та відображатися у графічному вигляді для подальшого аналізу.

Для реалізації поставленої задачі було використано мікроконтролер Arduino Mega та модулі MCP251 і ESP8266

Результатом розробки стала завершена система, яка готова до впровадження та використання для покращення обслуговування та моніторингу автомобілів.

СИСТЕМА ДИСТАНЦІЙНОГО МОНІТОРИНГУ показників стану автомобіля, Arduino Mega, MCP251, ESP8266.

Abstract

Qualification work. Remote Vehicle Condition Monitoring System. 54 Pages, 20 figures, 10 appendices, 6 sourse.

Object of research – remote vehicle condition monitoring system.

The purpose of the work is to develop a remote monitoring system that will allow real-time tracking of the main indicators of vehicle condition using an Arduino Mega microcontroller and MCP251 and ESP8266 modules.

The system should provide data collection from the CAN bus, such as speed, engine RPM, fuel level, engine temperature, and other parameters. The data will be transmitted to a web page, where they will be stored and displayed in graphical form for further analysis.

To achieve the set task, an Arduino Mega microcontroller and MCP251 and ESP8266 modules were used.

The result of the development is a complete system ready for implementation and use to improve vehicle maintenance and monitoring.

REMOTE VEHICLE CONDITION MONITORING SYSTEM, Arduino Mega, MCP251, ESP8266.

ЗМІСТ

Зм.	Арк.	№ докум.	Підпис	Дата	КР.КІ 24.528.02.000 ПЗ		
Розроб.		Грицаюк Б.Л.			Система дистанційного моніторингу показників стану автомобіля	Літ.	Арк. 6 / Акрушів 53
Перевір.		Сиротюк О.Б					
Реценз.		Посвятовська О.Б				ГФК.ВКТ. КІ - 41	
Н. Контр.		Слєпцова О.Я.					
Затверд.		Стефурак Н.А.					

ВСТУП

У сучасному світі автомобільний транспорт є важливою складовою повсякденного життя людей та економічної діяльності суспільства. Зростаюча кількість транспортних засобів на дорогах потребує ефективних рішень для моніторингу їхнього стану та забезпечення безпеки. Одним з таких рішень є система дистанційного моніторингу показників стану автомобіля.

Розробка та впровадження систем дистанційного моніторингу дозволяє своєчасно виявляти технічні несправності, контролювати рівень палива, стежити за параметрами роботи двигуна, гальмівної системи, системи охолодження та інших важливих елементів автомобіля. Такі системи забезпечують підвищення безпеки водія та пасажирів, зниження ризику аварійних ситуацій, а також економію коштів на ремонт та технічне обслуговування.

Мета цієї роботи – розробка та впровадження системи дистанційного моніторингу показників стану автомобіля, яка забезпечить надійний та зручний контроль за технічним станом транспортного засобу. У ході виконання роботи буде проведено аналіз існуючих рішень, визначено основні технічні вимоги до системи, розроблено програмне забезпечення та апаратну частину системи, а також проведено тестування та оцінку її ефективності.

Дана робота спрямована на підвищення рівня безпеки та надійності автомобільного транспорту, що є важливим кроком у розвитку сучасних технологій у сфері автотранспорту та покращенні якості життя суспільства.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної області

Система дистанційного моніторингу показників стану автомобіля є важливим компонентом сучасних технологій, спрямованих на підвищення безпеки, ефективності та зручності у використанні автотранспорту. Аналізуючи цю предметну область, можна розглянути кілька ключових аспектів:

– Технічні аспекти включають в себе апаратне забезпечення (сенсори, пристрої зчитування, модулі зв'язку), програмне забезпечення (програми аналізу даних, платформи для віддаленого доступу), а також методи збору, передачі, зберігання та обробки інформації про стан автомобіля.

– Функціональність включають в себе специфікації та можливості системи, такі як відстеження місцезнаходження, моніторинг рівня палива, діагностика автомобільних систем (двигун, гальмівна система, система охолодження тощо), сповіщення про неполадки та потенційні аварійні ситуації.

– Інтеграція та взаємодія з іншими системами управління транспортним потоком, системами GPS, мобільними додатками тощо. Важливо мати здатність взаємодіяти з цими системами для забезпечення більшої ефективності та функціональності.

– Економічні переваги, аналізуючи цю область, також важливо оцінити економічні переваги використання таких систем для автовласників, компаній з управління автопарками та інших учасників автомобільного ринку.

Загалом, аналіз предметної області системи дистанційного моніторингу показників стану автомобіля включає в себе широкий спектр технічних, функціональних, економічних, правових та етичних аспектів.

1.2 Огляд існуючих рішень

Огляд існуючих рішень у галузі систем дистанційного моніторингу показників стану автомобіля дасть можливість оцінити різноманітність

доступних продуктів та їхніх характеристик. Нижче представлено декілька прикладів існуючих рішень на ринку:

- OnStar.
- Vyncs.
- Mojio.
- Automatic.
- Bosch Connected Car.

OnStar (рис. 1.1) є системою дистанційного моніторингу та аварійного реагування, яка надається компанією General Motors (GM). Запущена в 1996 році, OnStar стала однією з перших у світі служб автомобільної безпеки, що надає підтримку в режимі реального часу для водіїв автомобілів GM по всьому світу. Система використовує GPS технології та мобільний зв'язок для надання широкого спектру послуг, від автоматичного реагування на аварії до навігації та діагностики автомобіля [3].

Основні функції та послуги OnStar включають:

- Автоматичне реагування на аварії. У разі серйозної аварії, система автоматично зв'язується з оператором OnStar, який може викликати аварійні служби до місця події, навіть якщо водій не в змозі самостійно звернутися по допомогу.
- Кнопка SOS. Водії можуть натиснути кнопку SOS в своєму автомобілі для негайного з'єднання з оператором OnStar у разі будь-якої надзвичайної ситуації.
- Діагностика автомобіля. OnStar може дистанційно моніторити стан різних систем автомобіля і надавати водію звіти про потребу в обслуговуванні або ремонті.
- Відстеження вкрадених автомобілів. У разі крадіжки автомобіля, OnStar може допомогти в слідкуванні за його місцезнаходженням і сприяти його швидкому поверненню.
- Мобільний додаток. OnStar пропонує мобільний додаток, що дозволяє

користувачам віддалено керувати різними функціями свого автомобіля, включаючи замки дверей, запуск двигуна та перегляд діагностичної інформації.

– Wi-Fi точка доступу. Деякі пакети OnStar включають можливість створення Wi-Fi точки доступу в автомобілі, що дозволяє підключення до інтернету на ходу для декількох пристроїв.

OnStar є платною службою, з різними планами абонентської плати, що забезпечують доступ до різних рівнів послуг. Система доступна не тільки для власників нових автомобілів GM, але і для деяких інших марок автомобілів через післямонтажні рішення.

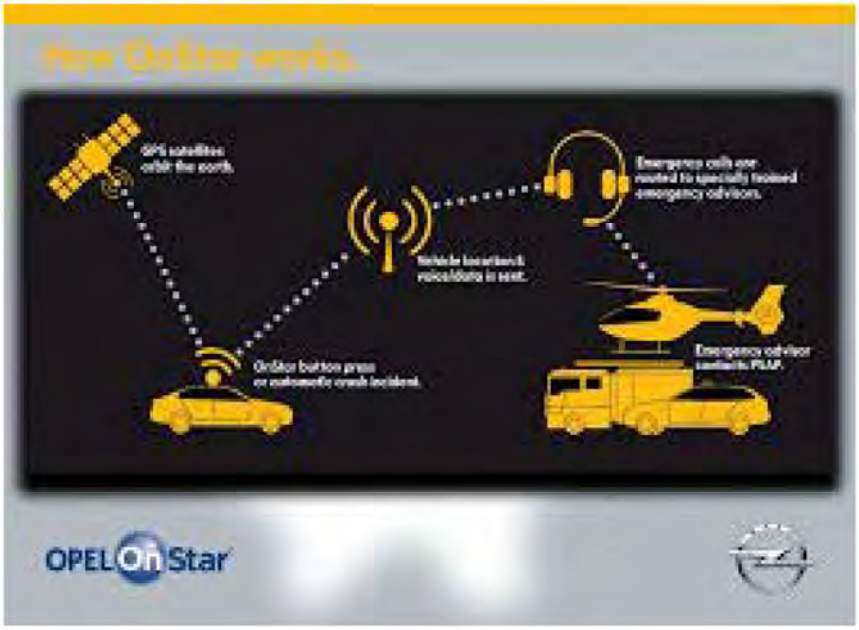

Рисунок 1.1 – Система OnStar

Vyncs (рис. 1.2) є розумною системою моніторингу автомобілів, яка використовує підключення до OBD-II (On-Board Diagnostics II) порту автомобіля для збору та передачі даних про стан і поведінку автомобіля в реальному часі. Ця система є продуктом компанії Agnik, яка спеціалізується на аналітиці великих даних, Інтернеті речей (IoT) та мобільних технологіях для автомобільної індустрії [4].

Основні функції та характеристики Vyncs включають:

– GPS-відстеження. Vyncs дозволяє користувачам відслідковувати місцезнаходження свого автомобіля в реальному часі, що є корисним для моніторингу автомобілів у корпоративному парку або для особистого використання.

– Діагностика автомобіля. Система забезпечує детальну інформацію про стан автомобіля, включаючи коди помилок діагностики, які допомагають виявити проблеми до візиту до автосервісу.

– Повідомлення та сповіщення. Користувачі можуть налаштувати сповіщення про різні події, такі як перевищення швидкості, використання автомобіля поза встановленими годинами або в'їзд/виїзд з геозон.

– Аналітика поїздок та патерни водіння. Vyncs аналізує дані про поїздки, включаючи швидкість, пройдену відстань та використання пального, допомагаючи оптимізувати маршрути та покращити ефективність використання пального.

– Оцінка стилю водіння. Система може оцінювати стиль водіння, вказуючи на агресивні або ризиковані маневри, що може сприяти покращенню безпеки на дорозі.

– Wi-Fi точка доступу. Деякі версії Vyncs також пропонують функціонал мобільної Wi-Fi точки доступу, що дозволяє пасажирам підключатися до інтернету під час поїздки.

– Без щомісячної абонплати. Vyncs вирізняється на ринку тим, що пропонує свої послуги без щомісячної абонплати, замість цього користувачі оплачують щорічний збір за використання послуг.

Vyncs пропонує різні пакети послуг, включаючи базовий, преміум, і для підприємств, кожен з яких має свій набір функцій, що дозволяє користувачам вибрати оптимальний варіант відповідно до своїх потреб. Це робить Vyncs універсальним рішенням для широкого спектра користувачів, від приватних власників автомобілів до великих корпоративних автопарків.

Рисунок 1.2 – Сисема Vyncs

Mojio (рис. 1.3) – це інноваційна платформа для підключених автомобілів, яка пропонує послуги моніторингу та діагностики автомобілів у реальному часі. Заснована в 2012 році, компанія Mojio розробляє рішення, які використовують обладнання, підключене до порту OBD-II (On-Board Diagnostics II) автомобіля, для збору даних про його стан і поведінку. Ці дані передаються через мобільний зв'язок і обробляються на платформі Mojio, надаючи користувачам доступ до цінної інформації через мобільний додаток або веб-інтерфейс [5]. Основні характеристики та можливості Mojio включають:

– Відстеження місцезнаходження автомобіля в реальному часі. Mojio дозволяє користувачам знати точне розташування їхнього автомобіля, що є корисним для моніторингу місцезнаходження дітей за кермом або відстеження вкраденого транспортного засобу.

– Діагностика та сповіщення про проблеми з автомобілем. Платформа може виявляти і повідомляти про механічні проблеми та помилки, виявлені через систему OBD-II, допомагаючи запобігти серйозним поломкам.

– Аналіз стилю водіння. Mojio аналізує стиль водіння і надає зворотний зв'язок, який може допомогти покращити безпеку водіння та знизити витрати на пальне.

– Повідомлення про аварії. В деяких випадках Mojio може автоматично

відправляти сповіщення в разі ДТП або інших надзвичайних ситуацій, що підвищує безпеку водія та пасажирів.

– Управління поїздками. Користувачі можуть переглядати історію поїздок, включаючи маршрути, тривалість та витрати на пальне, що може допомогти у плануванні та аналізі використання автомобіля.

– Інтеграція з третіми сторонами. Mojio пропонує API та SDK для розробників, що дозволяє інтегрувати дані з платформи у сторонні додатки та сервіси.

Mojio співпрацює з автомобільними виробниками, мобільними операторами та страховими компаніями для надання своїх послуг широкому спектру користувачів. Завдяки своїм інноваційним рішенням, компанія допомагає зробити водіння безпечнішим, економічнішим та зручнішим, надаючи цінну інформацію про стан і використання автомобіля

Рисунок 1.3 – Система Mojio

Automatic (рис. 1.4) була інноваційною системою дистанційного моніторингу автомобілів, яка з'єднувала водіїв з їхніми автомобілями через адаптер, підключений до порту OBD-II (On-Board Diagnostics II), та мобільний додаток. Заснована у 2011 році, компанія Automatic Labs пропонувала рішення, яке дозволяло власникам автомобілів отримувати детальну інформацію про

роботу свого транспортного засобу, стиль водіння, історію поїздок, а також різні сповіщення та рекомендації для покращення ефективності водіння [6]. Основні функції Automatic включали:

– Діагностика та сповіщення про помилки. Automatic могла ідентифікувати та пояснювати коди помилок двигуна, надаючи водіям змогу розуміти причини попереджувальних ламп на панелі приладів без необхідності відвідування сервісного центру.

– Аналіз стилю водіння та рекомендації. Програмне забезпечення аналізувало стиль водіння користувача, надаючи поради щодо покращення ефективності використання пального та зниження зносу автомобіля.

– Автоматичне викликання допомоги при аваріях. У випадку серйозної аварії, Automatic могла автоматично надсилати запит на екстрену допомогу, надаючи точне місцезнаходження автомобіля.

– Моніторинг поїздок. Користувачі могли відстежувати свої поїздки, включаючи маршрути, час в дорозі та витрати пального, що було корисно для особистого бюджетування або ведення податкового обліку.

– Підтримка смарт-дому. Automatic надавала інтеграцію з додатками та пристроями смарт-дому, дозволяючи, наприклад, автоматично вмикати освітлення вдома при наближенні автомобіля.

На жаль, незважаючи на свою інноваційність та користь для користувачів, Automatic Labs оголосила про закриття своєї діяльності у травні 2020 року, посилаючись на економічні виклики, пов'язані з пандемією COVID-19. Тим не менш, технології та ідеї, реалізовані Automatic, продовжують впливати на розвиток ринку підключених автомобілів, надихаючи інших розробників на створення нових інноваційних продуктів у цій сфері

Рисунок 1.4 – Система Automatic Labs

Bosch Connected Car (рис. 1.5) – це ініціатива компанії Bosch, яка має на меті інтегрувати передові цифрові технології в автомобілі, роблячи їх більш зв'язаними, інтелектуальними та ефективними. Bosch є одним із лідерів у сфері автомобільних технологій та компонентів, і їхня пропозиція в області зв'язаних автомобілів охоплює широкий спектр продуктів та послуг для особистих та комерційних транспортних засобів.[7]

Основні аспекти та можливості Bosch Connected Car включають:

– Діагностика та моніторинг стану автомобіля: Системи Bosch надають докладну інформацію про стан автомобіля, включаючи діагностику помилок, стан батареї, тиск у шинах та інше, допомагаючи водіям запобігати несподіваним поломкам та підтримувати автомобіль у належному стані.

– Екстрені та асистуючі сервіси: Bosch Connected Car пропонує різноманітні асистуючі сервіси, такі як автоматичний виклик екстреної допомоги у випадку ДТП, віддалений доступ до автомобіля через смартфон, а також функції контролю та управління для батьківського контролю.

– Оптимізація поїздок та управління пальним: Функції моніторингу дозволяють аналізувати стиль водіння та надають рекомендації для покращення ефективності використання пального, допомагаючи водіям знизити витрати та вплив на довкілля.

– Безпека та зручність: Системи безпеки, такі як автоматичне

гальмування в екстрених ситуаціях, виявлення "сліпих" зон, а також інтелектуальні паркувальні асистенти, роблять водіння безпечнішим та зручнішим.

– Інтеграція з розумним домом та іншими сервісами: Bosch пропонує інтеграцію автомобіля з розумним домом, дозволяючи, наприклад, управляти опаленням або освітленням вдома безпосередньо з автомобіля.

Розробка та підтримка екосистеми: Bosch активно працює над розширенням екосистеми партнерів та розробників для створення нових інноваційних рішень у сфері зв'язаних автомобілів.

Bosch Connected Car (рис. 1.5) використовує найновіші технології, включаючи штучний інтелект, машинне навчання, інтернет речей (IoT), для надання авангардних рішень, які покращують досвід водіння, підвищують безпеку та ефективність транспортних засобів. Bosch є одним із ключових гравців на ринку зв'язаних автомобілів, і їхня продукція та сервіси відіграють важливу роль у формуванні майбутнього автомобільної мобільності.

Ці рішення можуть відрізнятися за своїми можливостями, ціною, інтеграцією з іншими системами та іншими параметрами. При виборі конкретної системи важливо врахувати потреби та вимоги користувача, а також можливості та характеристики доступних рішень.

Рисунок 1.5 – Система Bosch Connected Car

Врахуємо результати огляду полярних систем в сфері моніторингу показників автомобіля в проектуванні власної системи моніторингу.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Виділення апаратних або програмних підсистем

Виділення апаратних або програмних підсистем та відображення на них зовнішніх функцій програмно-технічного засобу для забезпечення ефективної роботи системи необхідно розділити систему на окремі підсистеми, кожна з яких буде виконувати свої специфічні функції.

Визначимо основні апаратні підсистеми необхідні для функціонування дистанційної системи моніторингу показників стану автомобіля:

- Контролер. Дана підсистема забезпечує передачу даних від комунікаційного модуля до центральної системи через мережу та може виконувати попередню обробку даних.
- Комунікаційний модуль. Ця підсистема виконує локальну обробку даних, взаємодіє з контролером і бортовим комп'ютером
- Бортовий комп'ютер. Це підсистема, яка здійснює збір даних з сенсорів та передає їх у комунікаційний модуль.

Також визначимо основні програмні підсистеми необхідні для функціонування дистанційної системи моніторингу показників стану автомобіля:

- Система збору та обробки даних. Програмне забезпечення, яке працює на контролері та бортовому комп'ютері, обробляє дані з сенсорів та готує їх для передачі.
- Веб-інтерфейс. Підсистема дозволяє користувачам отримувати доступ до даних про стан автомобіля у реальному часі, переглядати історію показників та отримувати повідомлення про критичні стани.

Структурна схема системи дистанційного моніторингу показників стану автомобіля показана рисунку 2.1.

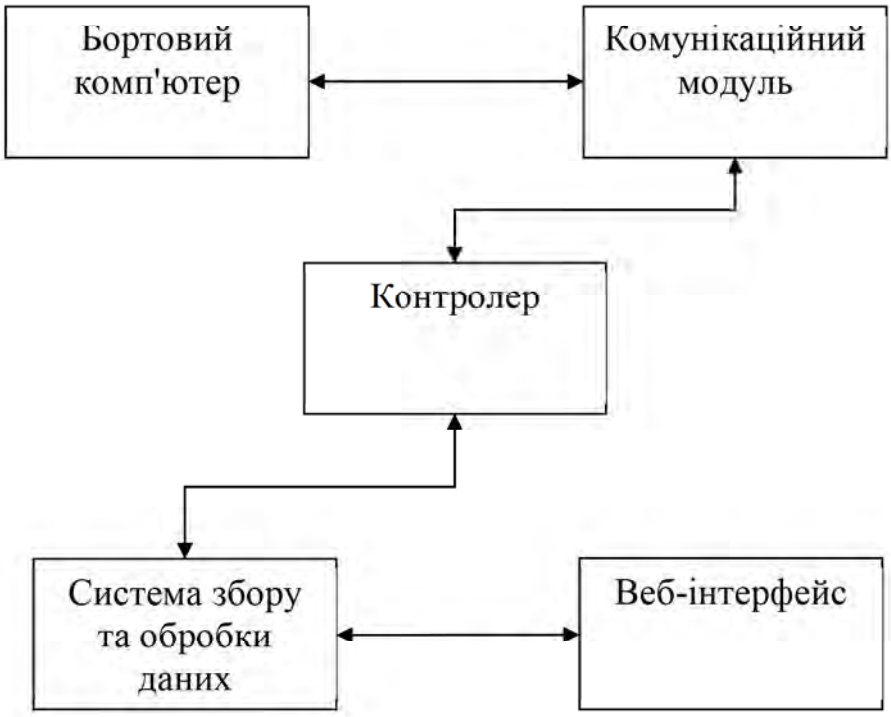

Рисунок 2.1 – Структурна схема

Відображення зовнішніх функцій на підсистеми дистанційної системи моніторингу показників стану автомобіля:

Контролер повинен забезпечити виконання таких функцій:

– Попередня обробка даних.
– Отримання даних від комунікаційного модуля.
– Забезпечення зв'язку з центральною системою через мережу (Wi-Fi, Bluetooth).

Комунікаційний модуль повинен забезпечити виконання таких функцій:

– Надсилання даних до контролера.
– Отримання даних від бортового комп'ютера

Бортовий комп'ютер повинен забезпечити виконання таких функцій:

– Отримання даних від сенсорів.
– Передача інформації до комунікаційного модуля.

Веб-інтерфейс повинен забезпечити виконання таких функцій:

– Відображення даних про стан автомобіля в реальному часі.
– Надання доступу до історії показників.
– Відправка повідомлень про критичні стани.

Продемонструємо взаємодію між підсистемами на наступній діаграмі (рис. 2.2).

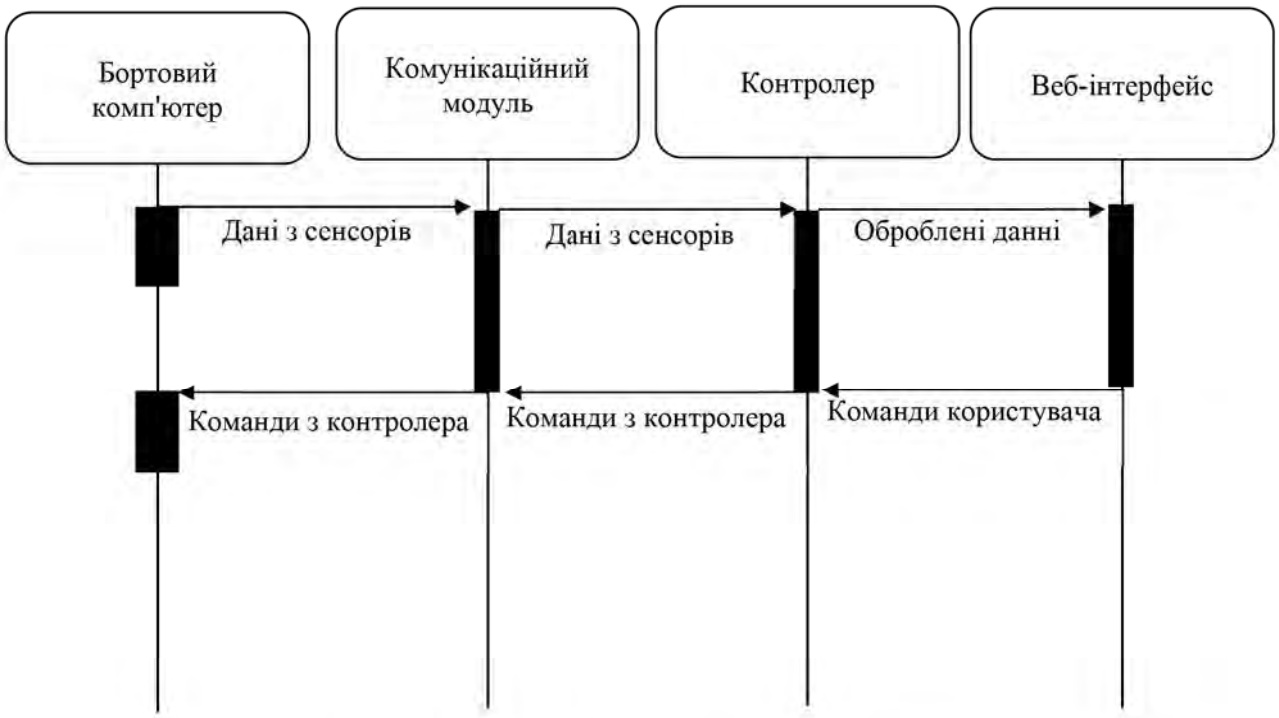

Рисунок 2.2 – Діаграма послідовності

Таким чином, виділення апаратних та програмних підсистем та відображення на них зовнішніх функцій забезпечує чітке розмежування обов'язків між компонентами системи та забезпечує їх ефективну взаємодію для виконання задач.

2.2 Взаємодія користувача з системою

Система моніторингу показників стану автомобіля дозволяє кінцевому користувачеві ефективно взаємодіяти з автомобілем і отримувати інформацію

про його технічний стан. Ця система забезпечує збір, обробку та представлення даних у зручній для користувача формі через веб-інтерфейс. Взаємодія користувача з системою охоплює кілька основних функцій, кожна з яких виконує специфічні завдання, спрямовані на підвищення зручності та ефективності експлуатації автомобіля.

Перед початком роботи з системою користувач повинен віти в браузер і війти на веб інтерфейс

Після успішного входу користувач може переглядати поточні показники стану автомобіля в режимі реального часу. Система збирає дані з різних сенсорів автомобіля, таких як температура двигуна, рівень палива, швидкість та інші технічні параметри, і відображає їх у зручному форматі. Це дозволяє користувачу оперативно отримувати інформацію про стан автомобіля і вчасно реагувати на будь-які відхилення від норми.

Система моніторингу також передбачає функцію отримання сповіщень про критичні стани автомобіля. Користувач отримує повідомлення у реальному часі про виявлені проблеми або небезпечні ситуації, такі як перегрів двигуна або низький рівень масла. Сповіщення можуть надходити через веб-інтерфейс користувача, що забезпечує своєчасне реагування на потенційні загрози.

Для забезпечення індивідуальних потреб користувача система дозволяє налаштовувати параметри моніторингу. Користувач може визначати частоту оновлення даних, встановлювати критичні значення для різних показників і налаштовувати інші параметри системи. Це забезпечує гнучкість і адаптивність системи до різних умов експлуатації автомобіля.

Взаємодія кінцевого користувача з системою моніторингу показників стану автомобіля наглядно показано на рисунку 2.3.

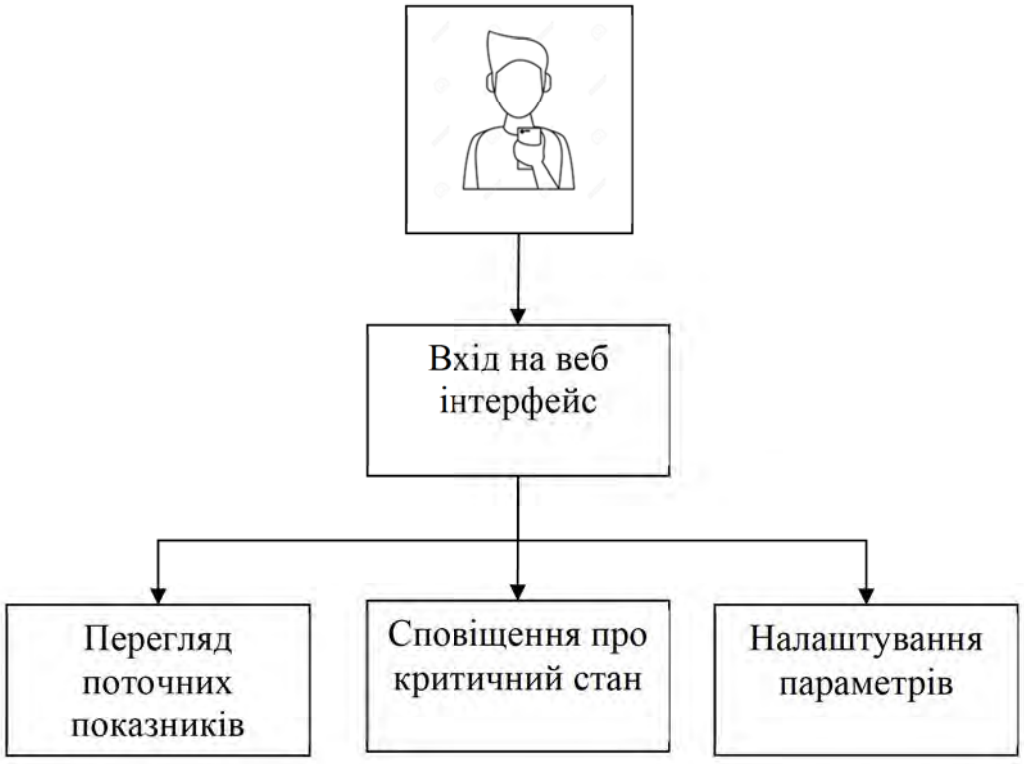

Рисунок 2.3 – Схема взаємодії користувача з системою

Результати проектування системи, а саме врахуємо при реалізація системи дистанційного моніторингу показників стану автомобіля

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Опис компонентів

Для реалізації системи моніторингу стану автомобіля обрано такі компоненти:

- Arduino Mega.
- Модуль CAN шини на MCP2515.
- Wi-Fi модуль ESP8266.

Arduino Mega (Рис. 2.1) – це потужна платформа для розробки електронних проектів, яка надає великі можливості для створення складних систем. Вона базується на мікроконтролері ATmega2560 і має значно більше виводів і пам'яті у порівнянні зі стандартною Arduino Uno, що робить її ідеальною для проектів, які вимагають великої кількості підключень і ресурсів.

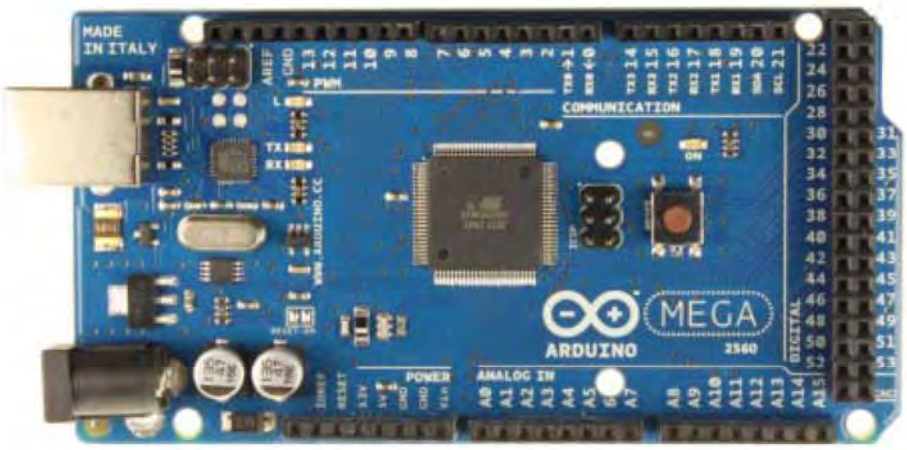

Рисунок 2.1 Arduino Mega

Arduino Mega оснащена 54 цифровими виводами вводу/виводу (з яких 15 можуть використовуватися як PWM-виводи), 16 аналоговими входами, 4 апаратними послідовними портами (UART), що дозволяє легко підключати кілька послідовних пристроїв одночасно. Також на платі є 256 KB флеш-пам'яті для зберігання програм, 8 KB SRAM і 4 KB EEPROM для збереження даних.

Arduino Mega підтримує живлення як від USB-порту, так і від зовнішнього джерела живлення (напруга живлення від 7 до 12 вольт). Вона також має кнопку перезавантаження, що полегшує процес налагодження проекту.

Швидкість передачі даних (тактова частота) становить 16 МГц, що дозволяє мікроконтролеру виконувати команди з високою швидкістю. Завдяки цьому Arduino Mega підходить для обробки складних алгоритмів та управління великою кількістю периферійних пристроїв.

Для програмування Arduino Mega використовується середовище розробки Arduino IDE, яке дозволяє писати коди на мові програмування C/C++. Програмування та завантаження коду на плату здійснюється через USB-кабель, що робить процес розробки інтуїтивно зрозумілим навіть для новачків.

Завдяки своїй гнучкості, потужності та зручності використання, Arduino Mega часто використовується в проектах автоматизації, робототехніки, створення інтерактивних пристроїв, а також в освітніх проектах для навчання електроніці та програмуванню.

Модуль CAN шини на базі мікросхеми MCP2515 (Рис. 2.2) є важливим компонентом для організації зв'язку в системах, які потребують надійної передачі даних в умовах сильних електромагнітних завад. CAN (Controller Area Network) протокол, на якому базується цей модуль, широко використовується в автомобільній промисловості, а також у промислових системах і інших високонадійних застосуваннях. MCP2515 є повноцінним контролером CAN, який реалізує всі функції обміну даними через CAN шину, що робить його універсальним рішенням для багатьох проектів.

Модуль складається з кількох основних компонентів. Центральним є мікроконтролер MCP2515, який відповідає за всі операції, пов'язані з CAN протоколом. Він підтримує стандарти CAN2.0A і CAN2.0B, що забезпечує сумісність з більшістю існуючих систем. Швидкість передачі даних може досягати 1 Мбіт/с, що дозволяє використовувати його в системах з високими вимогами до швидкості і надійності обміну даними.

Для фізичного рівня зв'язку модуль використовує трансівер TJA1050, який

конвертує сигнали між рівнями MCP2515 і CAN шиною. Це забезпечує надійний і стабільний зв'язок, навіть у складних умовах з високим рівнем електромагнітних завад.

Інтерфейс SPI використовується для зв'язку між MCP2515 і мікроконтролером, таким як Arduino. Це дозволяє легко інтегрувати модуль у більшість проектів. Під час ініціалізації через SPI інтерфейс встановлюються параметри роботи CAN шини, такі як швидкість передачі даних та інші налаштування.

Процес передачі даних включає надсилання даних з мікроконтролера на MCP2515 через SPI інтерфейс, після чого MCP2515 обробляє ці дані і передає їх на CAN шину через трансівер. У зворотному напрямку, дані, отримані через CAN шину, приймаються трансівером, обробляються MCP2515 і передаються на мікроконтролер для подальшої обробки.

Цей модуль широко використовується в різних застосуваннях. В автомобільних системах він застосовується для діагностики, моніторингу стану транспортного засобу та управління компонентами автомобіля. У промислових системах він використовується для управління та моніторингу на заводах і фабриках. Крім того, модуль може використовуватися у будь-яких інших системах, де необхідний надійний зв'язок між багатьма пристроями в умовах високих електромагнітних завад.

Завдяки своїй надійності, гнучкості і простоті інтеграції, модуль CAN шини на MCP2515 є потужним інструментом для реалізації ефективного і стабільного зв'язку в різних високонадійних системах

Рисунок 2.2 Модуль CAN шини на MCP2515

Wi-Fi модуль ESP8266 (рис. 2.3) – це популярний модуль для бездротового зв'язку, який надає можливість легко підключати електронні пристрої до мережі Інтернет. Він містить у собі мікроконтролер і вбудований Wi-Fi чіп, що робить його універсальним рішенням для розробки IoT (Internet of Things) проектів.

ESP8266 підтримує стандарт Wi-Fi 802.11 b/g/n і працює на частоті 2.4 ГГц, забезпечуючи швидкість передачі даних до 150 Мбіт/с. Модуль має вбудований TCP/IP стек, що дозволяє здійснювати мережеві з'єднання без потреби в додаткових мережевих процесорах. Він може працювати в режимі точки доступу (AP), клієнта (station) або одночасно в обох режимах (AP+station).

Модуль оснащений 32-бітовим мікроконтролером Tensilica Xtensa LX106, який працює на частоті до 160 МГц. Він має 64 KB оперативної пам'яті (SRAM) та 96 KB інструкційної RAM, що дозволяє виконувати досить складні обчислювальні завдання. Крім того, ESP8266 має флеш-пам'ять об'ємом від 512 KB до 4 MB, залежно від моделі, для зберігання програм і даних.

ESP8266 підтримує різні інтерфейси для підключення до інших пристроїв, такі як UART, SPI, I2C, PWM, ADC і GPIO. Це робить модуль дуже гнучким і дозволяє використовувати його у великій кількості застосувань, від простих датчиків до складних систем управління.

Програмування ESP8266 може здійснюватися за допомогою середовища розробки Arduino IDE, яка надає зручний інтерфейс і велику кількість бібліотек для роботи з Wi-Fi, HTTP, MQTT та іншими мережевими протоколами. Крім того, для розробки можна використовувати інші платформи, такі як NodeMCU (на основі мови Lua) або Espressif IDF (офіційний SDK від виробника).

Завдяки своїй невисокій вартості, компактним розмірам та широким можливостям, ESP8266 широко використовується в різних IoT проектах, домашніх автоматизаціях, системах віддаленого моніторингу, розумних датчиках та інших додатках, де потрібен бездротовий зв'язок.

Рисунок 2.3 – Wi-Fi модуль ESP8266

3.2 Інтерфейси передачі даних

В даній підсистемі для передачі даних використовується 4 інтерфейса передачі даних таких як:

- CAN шина;
- SPI інтерфейс;
- UART інтерфейс;
- HTTP протокол;

Інтерфейс CAN (Controller Area Network) – це високошвидкісна серійна шина передачі даних, яка була розроблена для забезпечення надійного зв'язку між електронними компонентами автомобілів. CAN використовується для обміну даними між різними мікроконтролерами та пристроями без необхідності центрального комп'ютера. Завдяки своїй надійності та ефективності, CAN знайшов застосування не лише в автомобільній промисловості, але й у багатьох інших галузях, включаючи промислову автоматизацію, медичне обладнання, побутову техніку та інші вбудовані системи.

CAN шина працює за принципом мультимастера, що означає, що будь-який вузол на шині може ініціювати передачу даних, коли шина вільна. Це дозволяє реалізувати децентралізовану систему зв'язку, де кожен вузол може безпосередньо взаємодіяти з іншими вузлами без посередництва центрального контролера.

Швидкість передачі даних CAN шини підтримує передачі даних до

1 Мбіт/с на відстанях до 40 метрів. На більших відстанях швидкість передачі знижується, наприклад, при довжині шини до 1 км швидкість знижується до 50 кбіт/с.

Метод доступу до середовища CAN шини використовує метод CSMA/CR (Carrier Sense Multiple Access with Collision Resolution), що дозволяє декільком вузлам передавати дані одночасно, а при виникненні колізії (одночасної передачі даних кількома вузлами) автоматично вирішувати її без втрати даних.

Формат кадру CAN шини складається з декількох полів, включаючи поле ідентифікатора (адреси), поле управління, поле даних (до 8 байт), поле контрольної суми CRC і поле кінця кадру. Ідентифікатор використовується для визначення пріоритету повідомлення, де нижче значення має вищий пріоритет.

Ідентифікація повідомлень адресації конкретних вузлів, CAN використовує ідентифікатори повідомлень. Кожне повідомлення має унікальний ідентифікатор, що визначає його зміст і пріоритет. Це дозволяє вузлам слухати тільки ті повідомлення, які їм необхідні.

Механізм виявлення помилок CAN шини включає кілька механізмів виявлення помилок, таких як перевірка на рівні бітів, перевірка значущості полів, контроль циклічним надлишковим кодом (CRC) і перевірка вікна підтвердження. Якщо помилка виявлена, повідомлення повторно передається.

CAN шина зазвичай використовує топологію шини, де всі вузли підключені до спільної лінії передачі. Це дозволяє легко додавати або видаляти вузли без значного впливу на роботу системи.

Простота інтеграції CAN інтерфейсу легко інтегрується з іншими протоколами і мікроконтролерами, що дозволяє створювати складні системи з багатьма різними типами пристроїв.

CAN інтерфейс забезпечує надійний і ефективний спосіб передачі даних в реальному часі між електронними компонентами, що робить його незамінним у багатьох сучасних вбудованих системах.

Інтерфейс SPI (Serial Peripheral Interface) – це високошвидкісна синхронна серійна шина передачі даних, яка широко використовується для підключення

периферійних пристроїв до мікроконтролерів. SPI був розроблений компанією Motorola і є дуже популярним завдяки своїй простоті та ефективності.

Синхронність передачі SPI інтерфейсу, це означає, що дані передаються разом з тактовим сигналом (SCK – Serial Clock). Тактовий сигнал синхронізує передачу даних між майстром (master) і веденими (slaves) пристроями.

Чотири лінії зв'язку:

– MOSI (Master Out Slave In): Лінія, по якій майстер передає дані веденому.

– MISO (Master In Slave Out): Лінія, по якій ведений передає дані майстру.

– SCK (Serial Clock): Тактовий сигнал, що генерується майстром для синхронізації передачі даних.

– SS/CS (Slave Select/Chip Select): Лінія вибору веденого пристрою. Коли ця лінія активна (зазвичай низький рівень сигналу), обраний ведений пристрій готовий до обміну даними з майстром.

SPI підтримує чотири режими роботи, які визначають полярність і фазу тактового сигналу. Ці режими конфігуруються за допомогою двох параметрів:

– CPOL (Clock Polarity) визначає, на якому рівні (низькому чи високому) знаходиться тактовий сигнал у стані спокою.

– CPHA (Clock Phase) визначає, на якому фронті тактового сигналу (на зростаючому чи спадному) зчитуються дані.

SPI підтримує повнодуплексний зв'язок, що означає, що дані можуть передаватися і прийматися одночасно. Це досягається через використання окремих ліній для передачі (MOSI) та прийому (MISO) даних.

Швидкість передачі SPI є високим, здатним передавати дані на швидкостях до десятків мегагерц. Швидкість передачі визначається тактовою частотою, яку генерує майстер.

Підтримка кількох ведених пристроїв, один основний пристрій може

керувати кількома веденими пристроями за допомогою окремих ліній вибору веденого (SS/CS). Це дозволяє підключати декілька периферійних пристроїв до одного мікроконтролера.

Реалізація інтерфейсу SPI має просту апаратну реалізацію і не потребує складних схем керування. Це робить його зручним для використання в багатьох мікроконтролерах та периферійних пристроях.

SPI є простим, ефективним і універсальним інтерфейсом, який забезпечує швидку та надійну передачу даних між мікроконтролером і периферійними пристроями. Завдяки своїй високій швидкості та низькій складності реалізації, він став стандартом для багатьох вбудованих систем.

Інтерфейс UART (Universal Asynchronous Receiver-Transmitter) – це універсальний асинхронний приймач-передавач, який широко використовується для серійної передачі даних між комп'ютерами та периферійними пристроями. UART є одним з найпопулярніших інтерфейсів завдяки своїй простоті та універсальності.

Асинхронність передачі UART інтерфейсу, це означає, що передача даних не синхронізується з тактовим сигналом. Кожен байт даних передається разом із стартовим і стоповим бітом, які сигналізують початок і кінець передачі байту.

Формат кадру даних в UART зазвичай включає:

– Стартовий біт – це один біт, який сигналізує початок передачі байту. Це завжди низький рівень (логічний 0).

– Біти даних – це біти від 5 до 9, які містять власне інформацію.

– Парність (необов'язково), передбачає, що один біт для перевірки парності (парний або непарний) для забезпечення цілісності даних.

– Стоповий біт(и) – це один або два біти високого рівня (логічний 1), які сигналізують кінець передачі байту.

Швидкість передачі даних в UART вимірюється в бодах (baud) і визначає кількість біт, які передаються за секунду. Типові швидкості передачі включають 9600, 19200, 38400, 57600, 115200 бод.

UART має просту апаратну реалізацію, що робить його зручним для використання в мікроконтролерах та інших цифрових системах. Більшість мікроконтролерів мають вбудовані UART модулі.

Повнодуплексний UART зв'язок, що означає можливість одночасної передачі і прийому даних. Для цього використовуються дві окремі лінії:

– Лінія передачі даних від передавача TX (Transmit).

– Лінія прийому даних приймачем RX (Receive).

Для забезпечення цілісності даних UART може використовувати біти парності. Однак, для більш надійного захисту від помилок зазвичай використовуються додаткові протоколи верхнього рівня.

UART широко використовується в різних застосуваннях, таких як:

– Підключення до комп'ютерів через COM-порти.

– Зв'язок між мікроконтролерами та периферійними пристроями (датчики, модулі пам'яті).

– Зв'язок між мікроконтролерами та модемами або іншими комунікаційними модулями (наприклад, Wi-Fi, Bluetooth).

Основні обмеження UART включають відносно низьку швидкість передачі порівняно з іншими інтерфейсами (наприклад, SPI або I2C) і залежність від зовнішніх факторів (наприклад, якість кабелів, електромагнітні перешкоди).

UART перетворює паралельні дані в серійний формат для передачі по одній лінії і назад з серійного в паралельний формат для обробки. Під час передачі даних UART додає до кожного байту даних стартовий і стоповий біти, а також, за необхідності, біт парності. Під час прийому UART перевіряє стартовий біт, зчитує задану кількість біт даних, перевіряє біт парності і чекає на стоповий біт. Це дозволяє приймачеві відновити передані дані в оригінальному вигляді.

UART є одним з найпростіших і найбільш надійних інтерфейсів для серійної передачі даних, завдяки чому він знайшов широке застосування в багатьох електронних пристроях і системах.

HTTP (HyperText Transfer Protocol) – це протокол прикладного рівня для передачі гіпертекстових документів, таких як HTML. Він є основою для будь-якого обміну даними в Інтернеті та підтримує комунікацію між веб-браузерами та веб-серверами. HTTP розроблено для забезпечення взаємодії між клієнтами та серверами, використовуючи запити та відповіді.

HTTP дотримується клієнт-серверної моделі, де клієнт (наприклад, веб-браузер) надсилає запити до сервера, а сервер відповідає відповідними даними. Це дозволяє розділити функціональність між пристроями та оптимізувати обробку запитів.

HTTP визначає кілька методів запитів, основними з яких є:

– GET, запитує ресурс із сервера. Використовується для отримання даних без впливу на стан сервера.

– POST, відправляє дані на сервер для створення або оновлення ресурсу. Використовується для передачі великих обсягів даних.

– PUT, замінює всі поточні представлення ресурсу даними із запиту.

– DELETE, видаляє вказаний ресурс.

– HEAD, запитує ресурс без тіла відповіді, що використовується для отримання метаданих.

– OPTIONS, запитує параметри комунікації для вказаного ресурсу.

– PATCH, вносить часткові зміни до ресурсу.

Запит HTTP складається з:

– Стартового рядка, що вказує на метод запиту, URI (універсальний ідентифікатор ресурсу) та версію HTTP (наприклад, `GET /index.html HTTP/1.1`).

– Заголовки запиту містять метадані про запит, такі як тип вмісту, авторизацію та інформацію про клієнта.

– Тіло запиту вміщує дані, що відправляються на сервер (для методів, які передають дані, таких як POST).

Відповідь HTTP складається з:

– Стартовий рядок вказує версію HTTP, код стану відповіді та опис стану (наприклад, `HTTP/1.1 200 OK`).

– Заголовки відповіді містять метадані про відповідь, такі як тип вмісту та тривалість кешування.

– Тіло відповіді вміщує дані, що повертаються клієнту (наприклад, HTML-код веб-сторінки).

HTTP використовує коди стану для вказівки результату запиту. Основні групи кодів стану:

– 1xx (Інформаційні) запити отримано і обробляється.

– 2xx (Успішні) запити успішно оброблено (наприклад, `200 OK`).

– 3xx (Перенаправлення) потрібні додаткові дії для завершення запиту (наприклад, `301 Moved Permanently`).

– 4xx (Помилки клієнта) запит містить помилку (наприклад, `404 Not Found`).

– 5xx (Помилки сервера) сервер не зміг виконати запит (наприклад, `500 Internal Server Error`).

HTTP є безстанним протоколом, що означає, що кожен запит обробляється незалежно від інших. Для збереження стану між запитами використовуються механізми, такі як cookies, сесії та токени.

Для забезпечення безпеки даних у мережі використовується HTTPS (HTTP Secure), який додає шифрування через SSL/TLS поверх стандартного HTTP. Це захищає дані від перехоплення та несанкціонованого доступу.

HTTP підтримує кешування для зменшення навантаження на сервери та підвищення швидкості доступу до ресурсів. Заголовки кешування (наприклад, `Cache-Control`) керують поведінкою кеша.

HTTP є фундаментальним протоколом для сучасного Інтернету, який забезпечує ефективну та надійну передачу даних між клієнтами та серверами, підтримуючи широкий спектр веб-додатків та сервісів.

3.3 Опис реалізації модулів апаратного та програмного забезпечення програмно-технічного засобу

Апаратне забезпечення системи дистанційного моніторингу показників стану автомобіля складається з таких основних апаратних модулів [2].

Arduino Mega 2560:

– Центральний контролер системи, який відповідає за обробку даних та взаємодію з іншими модулями.

– Володіє достатньою кількістю входів/виходів для підключення додаткових модулів і сенсорів.

– Підключення через SPI інтерфейс до MCP2515 та через UART інтерфейс до ESP8266.

Модуль CAN шини MCP2515 відповідає за прийом і передачу даних через CAN шину автомобіля використовує SPI інтерфейс для комунікації з Arduino Mega.

Wi-Fi модуль ESP8266 відповідає за бездротову передачу даних через Wi-Fi та використовує UART інтерфейс для комунікації з Arduino Mega.

Програмне забезпечення складається з двох основних частин прошивки для Arduino Mega та прошивки для ESP8266.

Прошивка для Arduino Mega написана на мові C++ з використанням бібліотек для роботи з CAN шиною (MCP_CAN) та SoftwareSerial. Основними функціями є ініціалізація SPI і UART інтерфейсів, обробка вхідних/вихідних повідомлень CAN шини, обмін даними з ESP8266 через UART. Основний код для системи моніторингу стану автомобіля наведено в додатку А.

Прошивка для ESP8266 написана на мові C++ з використанням бібліотек ESP8266WiFi і ESP8266WebServer. основними функціями є встановлення Wi-Fi з'єднання, створення HTTP сервера, обробка запитів від користувачів для перегляду даних. Основний код для системи моніторингу стану автомобіля наведено в додатку Б.

Ці модулі забезпечують повноцінне функціонування системи дистанційного моніторингу показників стану автомобіля, обмін даними між CAN

шиною, Arduino Mega та ESP8266, а також доступ до даних через Wi-Fi мережу та веб-інтерфейс.

3.4 Опис схем системи

Функціональна схема демонструє основні компоненти системи і їх взаємозв'язки в системі дистанційного моніторингу показників авто.

Основний контролер системи, який керує обміном даними між CAN шиною та Wi-Fi модулем, взаємодіє з MCP2515 через SPI інтерфейс та взаємодіє з ESP8266 через UART інтерфейс.

Модуль CAN шини MCP2515 виконує прийом і передачу даних через CAN шину підключений до Arduino Mega через SPI інтерфейс.

Wi-Fi модуль ESP8266 виконує передачу даних через Wi-Fi мережу, підключений до Arduino Mega через UART інтерфейс.

На рисунку 3.4 зображено функціональну схему системи дистанційного моніторингу стану авто

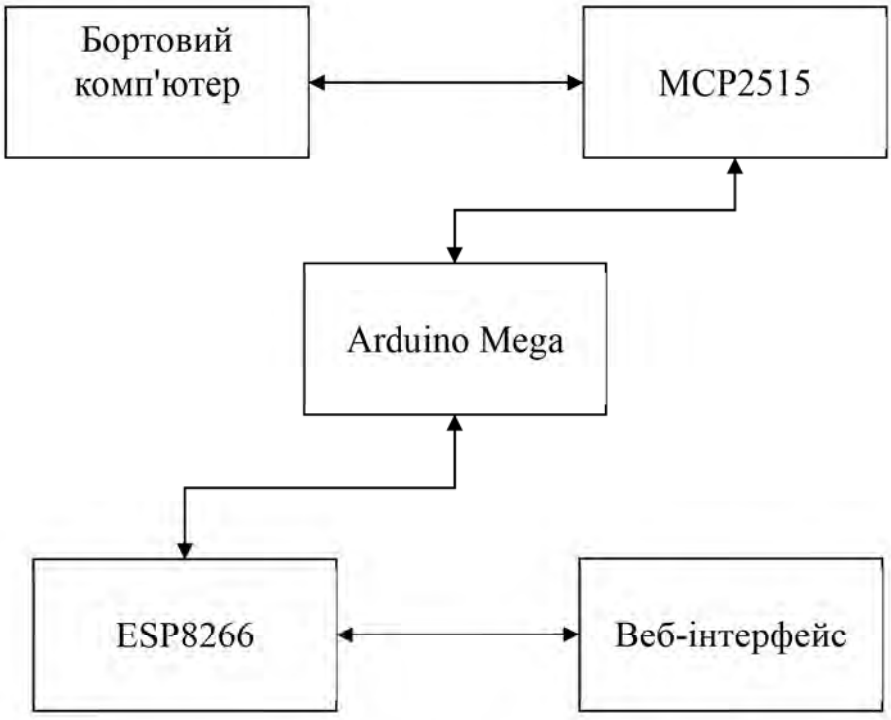

Рисунок 3.4 – Функціональна схема

Електрична схема описує детальні підключення між компонентами системи.

Підключення джерела живлення:

– VCC MCP2515 -> 5V Arduino Mega;

– GND MCP2515 -> GND Arduino Mega;

– VCC ESP8266 -> 3.3V Arduino Mega (або зовнішній стабілізатор напруги на 3.3V);

– GND ESP8266 -> GND Arduino Mega.

Підключення MCP2515 до Arduino Mega через SPI інтерфейс:

– CS MCP2515 -> Пін 10 Arduino Mega;

– SCK MCP2515 -> Пін 52 Arduino Mega;

– MOSI MCP2515 -> Пін 51 Arduino Mega;

– MISO MCP2515 -> Пін 50 Arduino Mega;

– INT MCP2515 -> Пін 2 Arduino Mega (опціонально).

Підключення ESP8266 до Arduino Mega через UART інтерфейс

– RX ESP8266 -> Пін 3 Arduino Mega (TX SoftwareSerial);

– TX ESP8266 -> Пін 2 Arduino Mega (RX SoftwareSerial).

Принципова схема демонструє основні електричні компоненти і їх підключення в системі дистанційного моніторингу стану показників авто

На рисунку 3.5 продемонструємо електричну схему яка описує детальні підключення між компонентами системи.

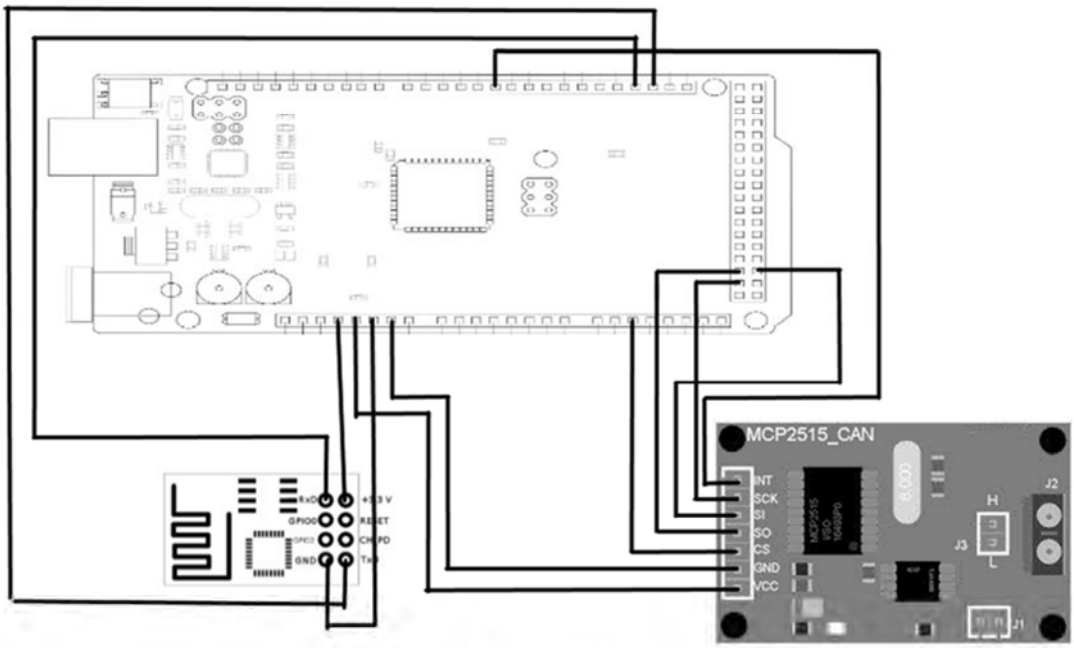

Рисунок 3.5 – Електрична схема

Arduino Mega 2560 виступає центральним контролером і обробляє всі вхідні та вихідні дані взаємодіє з MCP2515 через SPI інтерфейс та з ESP8266 через UART інтерфейс.

Модуль CAN шини MCP2515 Використовує SPI інтерфейс для обміну даними з Arduino Mega.

Схема підключення:

- VCC до 5V
- GND до GND
- CS до пін 10

- SCK до пін 52
- SI (MOSI) до пін 51
- SO (MISO) до пін 50
- INT (опціонально) до пін 2

Wi-Fi модуль ESP8266 використовує UART інтерфейс для обміну даними з Arduino Mega.

Схема підключення:

– VCC до 3.3V
– GND до GND
– RX до пін 3
– TX до пін 2

На рисунку 3.6 зображено Принципову схему системи дистанційного моніторингу стану авто.

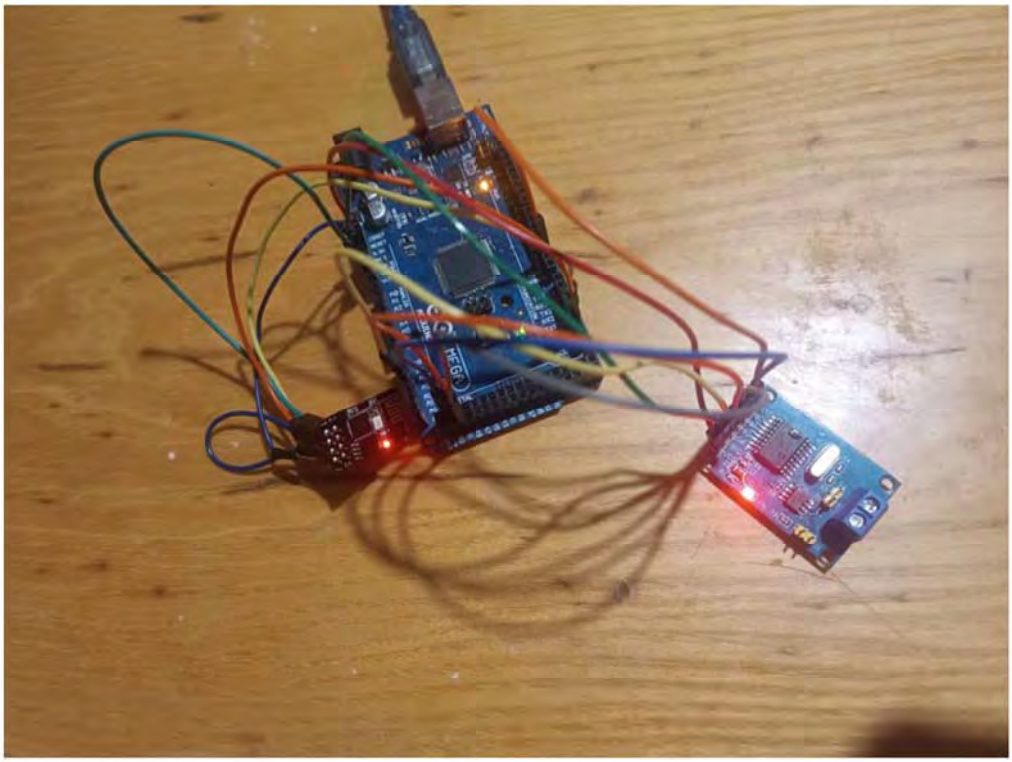

Рисунок 3.6 – Принципова схема

Цей підрозділ описує функціональну, електричну і принципову схему підключення основних компонентів системи. Це забезпечує повне розуміння роботи системи та її компонентів для забезпечення дистанційного моніторингу показників стану автомобіля.

3.5 Реалізація людино-машинного інтерфейсу

Реалізація людино-машинного інтерфейсу для системи дистанційного моніторингу автомобілів вимагає створення зручного та інтуїтивно зрозумілого інтерфейсу, який дозволить користувачам легко взаємодіяти з системою та

отримувати необхідну інформацію про стан своїх транспортних засобів.

Проектування інтерфейсу, який включає елементи управління, такі як кнопки, меню, графіки та карти для візуалізації даних. Розробка програмного забезпечення програмування логіки інтерфейсу, що включає збір та обробку даних з датчиків

Налаштування зв'язку між іншими підсистемами та обладнанням системи моніторингу, забезпечуючи синхронізацію даних та команд між інтерфейсом користувача та датчиками в автомобілі.

Регулярне оновлення програмного забезпечення для вдосконалення функціоналу HMI та врахування змін у потребах користувачів або технологічних стандартах.

Цей процес дозволяє створити ефективний інтерфейс, який спрощує управління системою дистанційного моніторингу автомобілем і робить цей процес більш інтуїтивно зрозумілим для користувача.

3.6 Інструкція користувача

Система дистанційного моніторингу автомобілів призначена для відслідковування показників стану автомобіля. Користувач має дотримуватися законодавства про захист даних.

Інтерфейс Користувача включає головний екран з меню управління та інформаційними панелями. Користувач може переглядати дані в реальному часі та історичні дані. Сповіщення про події виводяться у вигляді спливаючих повідомлень.

Послідовність дій користувача:

- Підключення до точки доступу системи.
- Вхід на веб сторінку з допомогою браузера.
- Перегляд інформації про стан автомобіля.
- Використання функцій управління, таких як встановлення час до наступного технічного обслуговування або чистка помилок.
- Вихід з системи після завершення сеансу роботи.

Ця інструкція допоможе користувачам ефективно використовувати систему дистанційного моніторингу автомобілей, забезпечуючи безпеку та конфіденційність даних.

3.7 Тестування системи

Встановлення цілей тестування для перевірки функціональності та продуктивності. Вибір методів тестування, які відповідають специфікаціям та обмеженням системи.

Метод «чорної скриньки» Тестування без знання внутрішньої структури системи.

Метод «білої скриньки» Тестування з урахуванням внутрішньої структури системи.

Функціональне тестування виконання конкретних функцій.

Нефункціональне тестування оцінюючи атрибутів системи, таких як продуктивність, безпека, зручність використання.

Тестування продуктивності перевіркою швидкості реакції системи та стабільності під навантаженням.

Тестування зручності використання оцінюючи інтерфейс користувача на інтуїтивну зручність.

Створення контрольованого тестового середовища.

Проведемо функціональне тестування системи моніторингу показників стану автомобіля згідно плану:

1) З'єднання доточки доступу.
2) Підключення до веб сторінки.
3) Вивід даних на панелі.
4) Вивід помилок системи.

Для реалізації цього плану тесту розробимо тест-кейси, які відповідають усім функціональним специфікаціям. Визначення критеріїв успіху та невдачі для кожного тест-кейсу є створення звіту про результати тестування, який включає:

– Ідентифікатор тест-кейсу.

– Короткий опис тест-кейсу.

– Перерахування вхідних та очікуваних вихідних значень.

– Перерахування реальних вихідних значень.

– Інформація про збіг або розбіжності між очікуваними та реальними вихідними значеннями.

– Висновок про успішність проходження тест-кейсу.

Звіт про результати тестування та акт тестування наведено в додатку В.

Ця стратегія дозволяє систематично перевірити працездатність програмно-технічного засобу, забезпечити його функціональну придатність і виявити потенційні проблеми перед впровадженням у виробництво.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Аналіз ринку збуту продукту чи послуги.

Основні техніко–економічні та експлуатаційні характеристики виробу:

– Платформа Arduino Mega.

– Комунікаційний модуль MCP2515 CAN шина.

– Бездротовий модуль ESP8266 Wi-Fi.

Цей виріб є модифікацією існуючих рішень на ринку, але пропонує покращену інтеграцію з використанням популярних і доступних компонентів, таких як Arduino Mega, MCP2515 і ESP8266. Він поєднує в собі переваги доступності, простоти у використанні та широких можливостей налаштування.

Потенційний замовник (покупець) виробу:

– Власники автомобілів, які бажають мати додатковий контроль за станом свого автомобіля.

– Автосервіси, що надають послуги з діагностики та моніторингу.

– Компанії з автопарками, що потребують моніторингу та управління транспортними засобами.

Ринки реалізації виробу:

– Глобальний ринок автомобільної електроніки та аксесуарів.

– Ринок автомобільних діагностичних інструментів.

– Онлайн платформи, такі як Amazon, eBay, Aliexpress.

– Спеціалізовані магазини автоелектроніки.

Методи продажу виробу:

– Онлайн продажі через власний вебсайт та популярні онлайн-платформи.

– Співпраця з роздрібними магазинами автоелектроніки.

– Участь у виставках та автомобільних шоу.

– Пряма реклама в автомобільних журналах та на вебсайтах.

Організація сервісного обслуговування:

– Допомога у встановленні та налаштуванні через онлайн-ресурси (відеоуроки, FAQ, технічна підтримка).

– Гарантійне обслуговування та заміна у випадку дефектів.

– Післяпродажне обслуговування та оновлення програмного забезпечення.

Залежно від маркетингових зусиль та партнерських відносин, можна очікувати продажі від декількох сотень до тисяч одиниць на місяць.

Головні конкуренти на ринку аналогічної продукції:

– OnStar.

– Vyncs.

– Mojio.

– Automatic.

– Bosch Connected Car.

OnStar пропонує широкий спектр функцій, включаючи діагностику автомобіля, відстеження місця розташування, екстрені виклики та дистанційне керування. Система відома своєю високою надійністю та інтеграцією з багатьма автомобільними брендами, але має високу ціну та потребує підписки на послуги. Ціна: близько 1400 грн.

Vyncs забезпечує відстеження місця розташування в режимі реального часу, діагностику автомобіля, моніторинг водіння та оповіщення про технічне обслуговування. Цей продукт має доступну ціну та не потребує щомісячних підписок, але має обмежену функціональність у порівнянні з більш дорогими рішеннями. Ціна: близько 1800 грн.

Mojio пропонує інтелектуальну телематику з можливістю підключення до різних додатків, включаючи відстеження місця розташування, діагностику та аналітику водіння. Висока точність та зручність використання роблять його привабливим для широкого кола користувачів, але система має високу ціну. Ціна: близько 3200 грн.

Automatic забезпечує простоту у використанні та інтеграцію з мобільними

пристроями для відстеження місця розташування, діагностики автомобіля та аналітики водіння. Цей продукт спрямований на кінцевого споживача, але має обмежену функціональність у порівнянні з професійними рішеннями та високу ціну. Ціна: близько 2800 грн.

Bosch Connected Car надає професійний рівень діагностики та аналітики, що робить його ідеальним для автосервісів. Система відома своєю високою надійністю та якістю, але потребує певного досвіду у використанні та має високу ціну. Ціна: близько 4000 грн.

Загалом, здійснюючи аналіз ринку та оцінюючи можливий життєвий цикл, можна прийти до висновку, що виготовлення та реалізація даного продукту є доцільним та перспективним напрямком.

4.2 Розрахунок витрат на проектування

Розрахунок витрат на проектування системи дистанційного моніторингу показників стану автомобіля виконується наступним чином.

Заробітна плата визначається виходячи з кількості виконавців, діючих посадових окладів та кількості місяців їх участі в розробці проекту.

Таблиця 4.1 – Відсоток нарахування відносно розряду

Посада	Розряд	Катр
Старший науковий співробітник	15	2.58
Науковий співробітник	14	2.42
Інженер І категорії	12	2.12

Ставка першого розряду 6000 грн.

Місячна заробітна плата:

- Старший науковий співробітник: 6000 × 2.58 = 15480 грн.
- Науковий співробітник: 6000 × 2.42 = 14520 грн.
- Інженер І категорії: 6000 × 2.12 = 12720 грн.

Розмір зарплати працівника за повністю виконану місячну (годинну) норму праці (ч. 1 ст. 31 Закону України "Про оплату праці" від 24.03.1995 р.) не може бути нижчим за розмір мінімальної зарплати, тобто із 1 січня 2024 р. не менш ніж 8000 грн. у погодинному розмірі з 1 січня – 46 грн. прожитковий мінімум для працездатних осіб з 1 січня 2024 року – 3000 грн. Граничний розмір зарплати, до якої застосовується податкова соціальна пільга, у 2024 році дорівнює 4240 грн.

Податок на доходи фізичних осіб:

- Старший науковий співробітник 15480 * 18% = 2786 грн.
- Науковий співробітник 14520 * 18% = 2613 грн.
- Інженер І категорії 12720 * 18% = 2289 грн.

Військовий збір:

- Старший науковий співробітник 15480 * 1.5% = 232 грн.
- Науковий співробітник 14520 * 1.5% = 217 грн.
- Інженер І категорії 12720 * 1.5% = 190 грн.

Єдиний внесок:

- Старший науковий співробітник 15480 * 22% = 3405 грн.
- Науковий співробітник 14520 * 22% = 3194 грн.
- Інженер І категорії 12720 * 22% = 2798 грн.

До виплати працівникові:

- Старший науковий співробітник 15480 – 2786 – 232 – 3405 = 9057 грн.
- Науковий співробітник 14520 – 2613 – 217 – 3194 = 8496 грн.
- Інженер І категорії 12720 – 2289 – 190 – 2798 = 7443 грн.

Зарплата за 5 місяців:

– Старший науковий співробітник 9057 грн/місяць × 5 місяців = 45285 грн.

– Науковий співробітни: 8496 грн/місяць × 5 місяців = 42480 грн.

– Інженер І категорії 7443 грн/місяць × 5 місяців = 37215 грн.

Подаємо розрахунки зарплати працівників в таблицю 4.2

Таблиця 4.2 – Нарахування зарплати робітникам

N п/п	Посада виконавця	Оклад грн/міс	Відрахуваня грн/міс	Кількість чол.	Кількість місяців	Сума з/п, грн
1.	Старший науковий співробітник	15480	6 423	1	5	45285
2.	Науковий співробітник	14520	6 024	1	5	42480
3.	Інженер І категорії	14520	5 277	1	5	37215

Загальну зарплату за 5 місяців розрахуємо сумою зарплат працівників 45285 грн + 42480 грн + 37 215 грн = 125 980 грн.

Відрахування на соціальні потреби розраховуються за ставкою 22% від суми заробітної плати 125 980 грн × 22% = 27 715 грн.

Вартість контрагентських робіт і послуг може складати 10-20% від суми зарплати основних виконавців 125 980 грн. × 15% = 18 897 грн.

Витрати на відрядження зазвичай визначаються приблизній сумі 20 000 грн.

Інші прямі витрати включають вартість спеціального обладнання, витратних матеріалів та канцелярських товарів і можуть складати 40-50% від видатків на заробітну плату 125 980 грн. × 45% = 56 741 грн.

Усього прямих витрат 125 980 грн. + 27 715 грн. + 18 897 грн. + +20 000 грн. + 56 741 грн. = 227 333 грн.

Обрахуємо вартість деталей.

Вартість пристроїв за одиницю:

– Arduino Mega 500 грн.

– CAN-шина на MCP2515 200 грн.

– Wi-Fi модуль ESP8266 150 грн.

Загальна вартість на одиницю 500 грн. + 200 грн. + 150 грн. = 850 грн.

Вартість за 5 місяців (20 робочих днів на місяць)

Щоденна вартість деталей 850 грн./одиниця × 50 одиниць/день = =42 500 грн./день

Місячна вартість деталей 42 500 грн./день × 20 днів/місяць = =850 000 грн./місяць

Загальна вартість деталей за 5 місяців = 850 000 грн./місяць × 5 місяців = =4 250 000 грн.

Накладні витрати враховують загально господарчі витрати по забезпеченню проведення роботи. Вони визначаються в відсотках (30-40%) від суми прямих витрат по даній роботі 227 333 грн. * 35% = 79 517 грн.

Планові накопичення визначається у відсотках (20-30%) від суми прямих, накладних витрат та вартості деталей (227333 грн. + 4250000 грн. + 79517 грн.)* * 20% = 1 128 712 грн.

Уся кошторисна вартість підраховується сумою прямих і накладних витрат та планових накопичень 4514850 грн. + 1 128 712 грн. = 5 643 562 грн.

Податок на додану вартість рахується множенням кошторисної вартості на 20% 5 643 562 грн. × 20% = 1 128 712 грн.

Договірна ціна включає ПДВ і кошторису вартість 5 643 562 грн. + +1 128 712 грн. = 6 772 274 грн.

Запишемо результати підрахунків в таблицю 4.3 для зручнішого перегляду результату.

Таблиця 4.3 – Розрахунок вартості проекту

Найменування статей витрат	Сума грн
Зарплата проектувальників.	125 980 грн.
Відрахування на соціальні потреби.	27 715 грн.
Контрагентські роботи і послуги.	18 897 грн.
Витрати на відрядження.	20 000 грн.
Інші прямі витрати.	56 741 грн.
Усього прямих витрат.	227 333 грн.
Вартість деталей	4 250 000 грн.
Накладні витрати.	79 517 грн.
Планові накопичення.	1 128 712 грн.
Усього, кошторисна вартість проекту.	5 643 562 грн.
Податок на додану вартість.	1 128 712 грн.
Загалом, договірна ціна розробки Зп	6 772 274 грн.

Отже, орієнтовна вартість проекту, включаючи всі компоненти та податки, складає 6 772 274 грн. а ціна однієї одиниці продукції вартує 1 355 грн.

4.3 Обґрунтування необхідності розробки

Пропонована система дистанційного моніторингу покликана задовольнити і поліпшити такі потреби замовників:

– Водії отримають можливість стежити за важливими показниками стану автомобіля в режимі реального часу через мобільний додаток або веб-інтерфейс.

– Автопарки зможуть контролювати стан своїх транспортних засобів,

що дозволить оперативно реагувати на будь-які несправності та планувати технічне обслуговування.

– Моніторинг критичних параметрів, таких як температура двигуна, рівень масла та наявність діагностичних кодів помилок, сприятиме запобіганню поломок і аварійних ситуацій.

– Власники автомобілів та автопарків зможуть знизити витрати на ремонт завдяки своєчасному виявленню та усуненню несправностей.

– Автосервіси зможуть надавати клієнтам більш точні та обґрунтовані рекомендації щодо обслуговування автомобіля.

Запровадження системи дистанційного моніторингу вплине на поліпшення економічних показників наступним чином:

– Своєчасне виявлення несправностей дозволить уникнути дорогих ремонтів.

– Підвищення ефективності використання палива завдяки оптимізації роботи автомобіля.

– Автопарки зможуть мінімізувати час простою транспортних засобів, плануючи технічне обслуговування на основі реальних даних про стан автомобілів.

– Завдяки оперативному доступу до даних про стан автомобіля, власники та автосервіси зможуть більш ефективно керувати своїми ресурсами та покращити обслуговування клієнтів.

Соціальний ефект:

– Підвищення безпеки водіїв та пасажирів завдяки своєчасному виявленню та усуненню несправностей.

– Покращення умов праці працівників автопарків та автосервісів завдяки доступу до актуальної інформації про стан автомобілів.

Екологічний ефект:

– Оптимізація роботи автомобілів сприятиме зниженню викидів шкідливих речовин у навколишнє середовище.

– Підвищення ефективності використання палива сприятиме зменшенню негативного впливу на екологію.

Запровадження даної системи дистанційного моніторингу показників стану автомобіля є обґрунтованим та перспективним проектом, який має значні економічні, соціальні та екологічні переваги. Це дозволить замовникам підвищити ефективність експлуатації транспортних засобів, знизити витрати та покращити безпеку і якість обслуговування.

ВИСНОВКИ

У результаті виконання роботи з розробки системи дистанційного моніторингу показників стану автомобіля на основі Arduino Mega, CAN шини на MCP2515 та Wi-Fi модуля ESP8266 вдалося досягти вагомих результатів, що свідчать про успішність і доцільність впровадження даної технології.

Перш за все, був проведений детальний аналіз існуючих рішень на ринку систем дистанційного моніторингу автомобілів. Вивчення наявних технологій, їхніх технічних характеристик та можливостей інтеграції з іншими системами дозволило сформувати чітке уявлення про вимоги до розроблюваної системи. Це забезпечило можливість уникнути повторення наявних недоліків та оптимально реалізувати функціонал.

Розробка системи включала створення апаратної та програмної частин. Використання платформи Arduino Mega у поєднанні з CAN шиною на MCP2515 забезпечило надійний збір даних з автомобіля, тоді як Wi-Fi модуль ESP8266 дозволив організувати передачу цих даних на сервер для подальшого аналізу та відображення в реальному часі. Це рішення забезпечило ефективний і стабільний зв'язок між автомобілем та віддаленим сервером.

Тестування розробленої системи проводилося у різних умовах експлуатації, що підтвердило високу точність і надійність зчитування даних. Система показала стабільну роботу в процесі передачі інформації та коректне відображення даних на сервері. Результати тестування свідчать про відповідність системи заявленим вимогам і її готовність до практичного використання.

Економічне обґрунтування проєкту включало детальний розрахунок витрат на розробку, впровадження та обслуговування системи. Виявилося, що розроблена система є економічно вигідною для впровадження як у приватних автомобілях, так і у великих автопарках. Це робить її привабливою для різних категорій споживачів.

На основі отриманих результатів були визначені перспективи подальшого розвитку системи. Серед можливих напрямків розширення функціональних

можливостей, інтеграція з іншими бортовими системами автомобіля, покращення інтерфейсу користувача та підвищення рівня безпеки даних. Ці кроки сприятимуть підвищенню ефективності та зручності використання системи.

Таким чином, розроблена система дистанційного моніторингу показників стану автомобіля на основі Arduino Mega, CAN шини на MCP2515 та Wi-Fi модуля ESP8266 відповідає сучасним вимогам і стандартам. Вона забезпечує високий рівень контролю за станом транспортного засобу, сприяючи підвищенню безпеки і ефективності його експлуатації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Галицький інститут імені В’ячеслава Чорновола: вебсайт. URL: https://gi.edu.ua/ (дата звернення: 15.06.2024).
2. Arduino.ua: вебсайт. URL: https://arduino.ua/ (дата звернення: 15.06.2024).
3. OnStar: вебсайт. URL: https://www.onstar.com/ (дата звернення: 15.06.2024).
4. Vyncs: вебсайт. URL: https://vyncs.com/vyncs2.0/welcome.aspx (дата звернення: 15.06.2024).
5. Moj.io: вебсайт. URL: https://www.moj.io/ (дата звернення: 15.06.2024).
6. Automatic Labs. вебсайт . URL: https://www.theverge.com/2020/5/1/21244295/automatic-labs-vehicle-tracking-service-covid-19-pandemic-rebate-may-28th (дата звернення: 15.06.2024).
7. Bosch Mobility. вебсайт. URL: https://www.bosch-mobility.com/en/mobility-topics/connected-services/ (дата звернення: 15.06.2024).

ДОДАТКИ
Додаток А

```
#include <SPI.h>
#include "mcp_can.h"

const int SPI_CS_PIN = 10;
MCP_CAN CAN(SPI_CS_PIN); // Set CS pin


int rawSpeed;
int rawRPM;
int rawFuel;
int oilStatus;
int rawTemperature;
int errors;
int speed;
int rpm;
int fuel;
int temperature;

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
};

void can () {
  unsigned char len = 0;
  unsigned char buf[8];
  if(CAN_MSGAVAIL == CAN.checkReceive()){
  CAN.readMsgBuf(&len, buf); // read data, len: data length, buf:
data buf
  unsigned long canId = CAN.getCanId();
  };

  if (CAN.getCanID() == "7E0"){
    int rawSpeed = CAN.getCan(1);          // Зчитування значення
швидкості
    int rawRPM = CAN.getCan(2);            // Зчитування значення
обертів
    int rawFuel = CAN.getCan(3);         // Зчитування рівня палива
    int oilStatus = CAN.getCan(4);       // Зчитування стану масла
    int rawTemperature = CAN.getCan(5);  // Зчитування температури
    int errors = CAN.getCan(6);          // Зчитування стану CHECK

    // Перетворення сигналів у відповідні одиниці
    int speed = map(rawSpeed, 0, 1023, 0, 240);       // Максимальна
швидкість 240 км/год
    int rpm = map(rawRPM, 0, 1023, 0, 8000);          // Максимальні
оберти 8000 RPM
    int fuel = map(rawFuel, 0, 1023, 0, 100);        // Максимальний
рівень палива 100%
```

```
    int temperature = map(rawTemperature, 0, 1023, 0, 120); // Максимальна температура 120°C
  };
}

void arduino (){
  int rawSpeed = analogRead(A0);           // Зчитування значення швидкості
  int rawRPM = analogRead(A1);             // Зчитування значення обертів
  int rawFuel = analogRead(A2);          // Зчитування рівня палива
  int oilStatus = digitalRead(3);        // Зчитування стану масла
  int rawTemperature = analogRead(A3);   // Зчитування температури
  int errors = digitalRead(4);           // Зчитування стану CHECK

  // Перетворення сигналів у відповідні одиниці
  int speed = map(rawSpeed, 0, 1023, 0, 240);        // Максимальна швидкість 240 км/год
  int rpm = map(rawRPM, 0, 1023, 0, 8000);           // Максимальні оберти 8000 RPM
  int fuel = map(rawFuel, 0, 1023, 0, 100);         // Максимальний рівень палива 100%
  int temperature = map(rawTemperature, 0, 1023, 0, 120); // Максимальна температура 120°C
}

void loop() {
  arduino();//з датчиків
  //can();//з can шини
  // Передача даних через Serial в форматі CSV
  Serial1.print(speed);
  Serial1.print(",");
  Serial1.print(rpm);
  Serial1.print(",");
  Serial1.print(fuel);
  Serial1.print(",");
  Serial1.print(oilStatus);
  Serial1.print(",");
  Serial1.print(temperature);
  Serial1.print(",");
  Serial1.println(errors);

  delay(1000); // Затримка перед наступним зчитуванням даних
}
```

Додаток Б

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <WebSocketsServer.h>

const char* ssid = "Car";
const char* password = "12345678";

ESP8266WebServer server(80);
WebSocketsServer webSocket = WebSocketsServer(81);

int speed = 0;
int rpm = 0;
int fuel = 100;
int oilStatus = 1;
int temperature = 90;
int errors = 0;

void handleRoot() {
  String html = "<!DOCTYPE html><html lang='uk'><head><meta
charset='UTF-8'><meta name='viewport' content='width=device-width,
initial-scale=1.0'><title>Бортова панель
автомобіля</title><style>";
  html += "body {font-family: Arial, sans-serif;background-color:
#333;color: #fff;display: flex;flex-direction: column;align-items:
center;justify-content: center;height: 100vh;margin: 0;}";
  html += ".dashboard {text-align: center;background-color:
#555;padding: 20px;border-radius: 10px;border: 2px solid #000;}";
  html += ".large-number {font-size: 5em;}";
  html += ".small-text {font-size: 2em;color: #000;}";
  html += ".row {display: flex;justify-content: center;align-items:
baseline;gap: 40px;}";
  html += ".column {display: flex;flex-direction: column;align-
items: center;}";
  html += ".status {padding: 10px;margin: 10px;border-radius:
5px;cursor: pointer;}";
  html += ".status.black {color: white;}";
  html += ".status.red {color: red;}";
  html += ".status.yellow {color: yellow;}";
  html += ".circle {border: 2px solid #000;border-radius:
50%;padding: 20px;}";
  html += ".square {border: 2px solid #000;padding: 10px;}";
  html += ".medium {font-size: 2em;color: #000;}";
  html += "</style></head><body>";
  html += "<div class='dashboard'>";
  html += "<div class='row'>";
  html += "<div class='column circle'><div class='large-number'
id='speed'>" + String(speed) + "</div><div class='small-
text'>КМ/ГОД</div></div>";
  html += "<div class='column circle'><div class='large-number'
id='rpm'>" + String(rpm) + "</div><div class='small-
text'>RPM</div></div>";
```

```
  html += "</div>";
  html += "<div class='row medium square'>";
  html += "<div class='column'>Паливо: <span id='fuel'>" +
String(fuel) + "</span>%</div>";
  html += "<div class='column'>OIL: <span id='oilStatus'
class='status " + String(oilStatus ? "black" : "red") +
"'>OIL</span></div>";
  html += "<div class='column'>CHECK: <span id='checkStatus'
class='status " + String(errors ? "yellow" : "black") + "'
onclick='checkErrors()'>CHECK</span></div>";
  html += "<div class='column'>Температура: <span id='temperature'>"
+ String(temperature) + "</span>°C</div>";
  html += "</div>";
  html += "</div>";
  html += "<script>var connection = new WebSocket('ws://' +
window.location.hostname + ':81/');";
  html += "connection.onmessage = function(event) {var data =
JSON.parse(event.data);";
  html += "document.getElementById('speed').innerText =
data.speed;";
  html += "document.getElementById('rpm').innerText = data.rpm;";
  html += "document.getElementById('fuel').innerText = data.fuel;";
  html += "document.getElementById('oilStatus').className = 'status
' + (data.oilStatus ? 'black' : 'red');";
  html += "document.getElementById('temperature').innerText =
data.temperature;";
  html += "document.getElementById('checkStatus').className =
'status ' + (data.errors ? 'yellow' : 'black');";
  html += "};";
  html += "function checkErrors() {
alert(document.getElementById('checkStatus').className.includes('y
ellow') ? 'Проблема з двигуном!' : 'Помилок немає'); }</script>";
  html += "</body></html>";

  server.send(200, "text/html", html);
}

void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");

  server.on("/", handleRoot);
  server.begin();
  Serial.println("HTTP server started");

  webSocket.begin();
```

```
  webSocket.onEvent(webSocketEvent);
  Serial.println("WebSocket server started");
}

void loop() {
  server.handleClient();
  webSocket.loop();

  if (Serial.available()) {
    String data = Serial.readStringUntil('\n');
    data.trim();
    Serial.println("Received data: " + data); // Додаємо виведення
отриманих даних для налагодження

    int values[6];
    int index = 0;

    char* token = strtok((char*)data.c_str(), ",");
    while (token != NULL && index < 6) {
      values[index++] = atoi(token);
      token = strtok(NULL, ",");
    }

    if (index == 6) {
      speed = values[0];
      rpm = values[1];
      fuel = values[2];
      oilStatus = values[3];
      temperature = values[4];
      errors = values[5];

      String json = "{\"speed\":" + String(speed) + ",\"rpm\":" +
String(rpm) + ",\"fuel\":" + String(fuel) + ",\"oilStatus\":" +
String(oilStatus) + ",\"temperature\":" + String(temperature) +
",\"errors\":" + String(errors) + "}";
      webSocket.broadcastTXT(json);

      Serial.println("Updated values:");
      Serial.println("Speed: " + String(speed));
      Serial.println("RPM: " + String(rpm));
      Serial.println("Fuel: " + String(fuel));
      Serial.println("Oil Status: " + String(oilStatus));
      Serial.println("Temperature: " + String(temperature));
      Serial.println("Errors: " + String(errors));
    }
  }
}

void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload,
size_t length) {
  if (type == WStype_DISCONNECTED) {
    Serial.printf("[%u] Disconnected!\n", num);
  } else if (type == WStype_CONNECTED) {
```

```
      IPAddress ip = webSocket.remoteIP(num);
      Serial.printf("[%u] Connection from ", num);
      Serial.println(ip);
    }
  }
```

Додаток В

Ідентифікатор тест-кейсу: TC001

Короткий опис тест-кейсу:

Тестування з'єднання мікроконтролера з точкою доступу Wi-Fi для забезпечення передачі даних на віддалений сервер.

Вхідні значення:

- SSID (назва точки доступу): "TestNetwork".
- Пароль: "TestPassword".
- Мікроконтролер: ESP8266.

Очікувані вихідні значення:

- Успішне підключення до мережі Wi-Fi.
- Отримання IP-адреси від точки доступу.
- Стан з'єднання: "CONNECTED"
- Сигнал наявності підключення в консолі або інтерфейсі користувача: "Successfully connected to TestNetwork"

Реальні вихідні значення:

- Підключення до мережі Wi-Fi: успішне.
- Отримана IP-адреса: 192.168.1.100.
- Стан з'єднання: "CONNECTED".
- Відображення повідомлення в консолі: "Successfully connected to TestNetwork".

Інформація про збіг або розбіжності між очікуваними та реальними вихідними значеннями:

Очікувані та реальні вихідні значення збігаються. З'єднання до точки доступу було успішно виконано, IP-адреса отримана, та стан з'єднання підтверджений.

Висновок про успішність проходження тест-кейсу:

Тест-кейс успішно пройдено. Мікроконтролер ESP8266 підключився до точки доступу "TestNetwork" з використанням вказаного пароля, отримав IP-адресу та підтвердив стан з'єднання, що відповідає очікуваним результатам.

Ідентифікатор тест-кейсу: TC002

Короткий опис тест-кейсу:

Тестування підключення до веб-сторінки для передачі та відображення даних в реальному часі.

Вхідні значення:

IP веб-сторінки: 192.168.1.100.

Мікроконтролер: ESP8266

Очікувані вихідні значення:

- Успішне підключення до веб-сторінки.
- Відправка даних на веб-сторінку.
- Отримання відповіді від веб-сторінки.
- Відображення даних на веб-сторінці.

Реальні вихідні значення:

- Підключення до веб-сторінки успішне.
- Дані передані.
- Отримана відповідь від сервера.
- Дані відображені на веб-сторінці в реальному часі.

Інформація про збіг або розбіжності між очікуваними та реальними вихідними значеннями:

Очікувані та реальні вихідні значення збігаються. Підключення до веб-сторінки було успішним, дані передані та отримані сервером, і відображені на веб-сторінці.

Висновок про успішність проходження тест-кейсу:

Тест-кейс успішно пройдено. Відбулося підключення до веб-сторінки, передані дані, отримали позитивну відповідь від сервера та забезпечили відображення даних на веб-сторінці, що відповідає очікуваним результатам.

Ідентифікатор тест-кейсу: ТС003

Короткий опис тест-кейсу:

Тестування виводу даних на панель індикації для відображення основних показників стану автомобіля.

Вхідні значення:

– Показники для відображення: швидкість, оберти двигуна, рівень палива, температура двигуна.

– Мікроконтролер: Arduino Mega

– Панель індикації

Очікувані вихідні значення:

– Швидкість: 60 км/год

– Оберти двигуна: 3000 об/хв

– Рівень палива: 80%

– Температура двигуна: 90°C

– Відображення відповідних значень на панелі індикації

Реальні вихідні значення:

– Швидкість: 60 км/год

– Оберти двигуна: 3000 об/хв

– Рівень палива: 80%

– Температура двигуна: 90°C

– Відображення значень на панелі індикації

Інформація про збіг або розбіжності між очікуваними та реальними вихідними значеннями:

Очікувані та реальні вихідні значення збігаються. Дані виведені на панель індикації, і показники відповідають очікуваним значенням.

Висновок про успішність проходження тест-кейсу:

Тест-кейс успішно пройдено. Мікроконтролер Arduino Mega коректно відобразив дані на панелі індикації. Відображені значення швидкості, обертів двигуна, рівня палива та температури двигуна відповідають очікуваним

результатам.

Ідентифікатор тест-кейсу: ТС004

Короткий опис тест-кейсу:

Тестування відображення помилок системи на веб-сторінці для забезпечення моніторингу та діагностики стану системи в реальному часі.

Вхідні значення:

- Помилки системи: перевищення допустимого температурного режиму, низький рівень палива і помилки двигуна.
- Мікроконтролер ESP8266.
- URL веб-сторінки для відображення помилок.

Очікувані вихідні значення:

- Пробеми з дигуном.
- Перевищення допустимого температурного режиму.
- Низький рівень палива.
- Відображення помилок на веб-сторінці в режимі реального часу.

Реальні вихідні значення:

- Пробеми з дигуном.
- Перевищення допустимого температурного режиму.
- Низький рівень палива.
- Відображення помилок на веб-сторінці в режимі реального часу.

Інформація про збіг або розбіжності між очікуваними та реальними вихідними значеннями:

Очікувані та реальні вихідні значення збігаються. Всі помилки системи були коректно передані та відображені на веб-сторінці.

Висновок про успішність проходження тест-кейсу:

Тест-кейс успішно пройдено. Мікроконтролер ESP8266 коректно передав інформацію про помилки системи на веб-сторінку. Помилки відображені в режимі реального часу, що відповідає очікуваним результатам.