ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Розробка розумної розетки з дистанційним управлінням»
(тема кваліфікаційної роботи українською мовою)

«Development of a smart socket with remote control»
(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 123 Комп'ютерна інженерія
(код, назва спеціальності)
Освітня програма Комп'ютерна інженерія
(назва)
Кабаков Данило Олександрович
(прізвище, ім'я, по-батькові здобувача)

Керівник д.т.н., проф. Гунченко Ю.О. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)
Рецензент ст. викл. Шаріпова І.В.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
комп'ютерних систем та технологій
№ ___ від ____.____.20___ р.

Завідувач кафедри
_____ Юрій ГУНЧЕНКО
(підпис) (прізвище, ім'я)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.

Оцінка_____/___/_____
(за національною шкалою / шкалою ECTS / бали)

Голова ЕК
_____ Світлана АНТОЩУК
(підпис) (прізвище, ім'я)

Одеса 2025

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці розумної розетки з дистанційним управлінням, яка поєднує в собі функції моніторингу енергоспоживання та захисту електроприладів. Актуальність теми обумовлена зростаючою потребою в енергоефективних, безпечних і зручних рішеннях для побуту в умовах розвитку технологій «розумного дому» та Інтернету речей.

Метою роботи є розробка розумної розетки з дистанційним керуванням, яка забезпечує базові функції моніторингу, захисту та керування живленням електроприладів. У межах дослідження проведено аналіз існуючих технічних рішень, обґрунтовано вибір елементної бази, розроблено принципову схему, реалізовано апаратну частину системи, а також створено програмне забезпечення.

Отриманий результат забезпечує дистанційне керування електроприладами, захист від перевантажень і можливість моніторингу параметрів живлення. Розроблена система може використовуватись у приватних оселях, офісах, майстернях та інших середовищах, де необхідне інтелектуальне керування електроживленням.

ABSTRACT

The bachelor's thesis is devoted to the development of a smart socket with remote control, which combines the functions of monitoring energy consumption and protecting electrical appliances. The relevance of the topic is due to the growing need for energy-efficient, safe and convenient solutions for everyday life in the context of the development of "smart home" technologies and the Internet of Things.

The purpose of the work is to develop a smart socket with remote control, which provides basic functions of monitoring, protecting and controlling the power supply of electrical appliances. Within the framework of the study, an analysis of existing technical solutions was conducted, the choice of the element base was justified, a schematic diagram was developed, the hardware part of the system was implemented, and software was created.

The result obtained provides remote control of electrical appliances, protection against overloads and the ability to monitor power parameters. The developed system can be used in private homes, offices, workshops and other environments where intelligent power supply control is required.

ЗМІСТ

ВСТУП

У сучасному світі цифрових технологій стрімко зростає потреба в автоматизації побутових процесів, підвищенні енергоефективності та зручності взаємодії людини з електронними пристроями. Одним із рішень у цьому напрямі є використання розумних розеток – пристроїв, які дозволяють керувати електроживленням підключених приладів дистанційно, автоматизовано або за заданими сценаріями. Ці системи знаходять широке застосування у сфері «розумного дому» та Інтернету речей (IoT), що формують новий рівень комфорту та енергозбереження.

Актуальність теми зумовлена декількома чинниками. По-перше, збільшення кількості електроприладів у побуті потребує більш ефективного керування енергоспоживанням. По-друге, стрімкий розвиток технологій бездротового зв’язку, таких як Wi-Fi і Bluetooth, спрощує інтеграцію побутових пристроїв у єдину мережу. По-третє, попит на пристрої з можливістю віддаленого керування постійно зростає, що відкриває нові перспективи для їх удосконалення. Моніторинг споживання електроенергії дозволяє не тільки контролювати витрати, але й запобігати аварійним ситуаціям. Розробка доступної та функціональної системи моніторингу струму є актуальним завданням для промислових та побутових застосувань. Світові тенденції демонструють зацікавленість провідних компаній у створенні багатофункціональних, компактних та недорогих рішень з високим ступенем надійності.

У той же час аналіз існуючих розробок показує, що багато пристроїв або є надмірно складними у реалізації для кінцевого споживача, або мають обмежений функціонал. Це створює передумови для створення нової моделі розумної розетки з акцентом на баланс між функціональністю, зручністю використання та низькою вартістю реалізації. Вибір теми обумовлений прагненням розробити універсальний пристрій, який міг би як забезпечувати

керування живленням приладів, так і надавати базові функції захисту та моніторингу енергоспоживання.

Мета роботи – розробити розумну розетку з дистанційним керуванням, яка забезпечує базові функції моніторингу, захисту та керування живленням електроприладів.

Завдання для досягнення мети:

1) провести аналіз існуючих технічних рішень у сфері розумних розеток;

2) обґрунтувати вибір мікроконтролера та інших компонентів;

3) розробити принципову електричну схему пристрою;

4) створити програмне забезпечення для пристрою;

5) протестувати працездатність системи та оцінити її ефективність у типових сценаріях використання.

Таким чином, дана кваліфікаційна робота спрямована на вирішення актуальної прикладної задачі у сфері комп’ютерної інженерії, що має практичну доцільність для застосування в умовах сучасного побуту. Розроблена система розумної розетки дозволяє забезпечити не лише комфортне дистанційне керування електроприладами, але й сприяє підвищенню енергоефективності, безпеки користувача та раціональному використанню електроенергії. Отримані результати можуть бути використані як у приватних житлових приміщеннях, так і в офісах, майстернях або інших середовищах, де потрібне інтелектуальне керування електроживленням.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд та аналіз існуючих рішень

У межах дослідження проаналізовано наявні актуальні наукові публікації, присвячені розробці та вдосконаленню пристроїв дистанційного керування електроживленням – розумних розеток. Вони мають схоже функціональне призначення, проте відрізняються як апаратною реалізацією, так і логікою керування.

У роботі авторів Zhang M. та Du J. [1] представлено проєкт розумної розетки, реалізованої на базі мікроконтролера STM32. Автори зосередили увагу на модульній архітектурі системи: до складу пристрою входять модулі для вимірювання струму та напруги, модуль захисту від перевантажень, а також бездротовий передавач (Wi-Fi-модуль ESP8266).

Переваги даного проекту:

- високоточне вимірювання електричних параметрів завдяки використанню STM32;
- реалізований алгоритм автоматичного відключення при виявленні перевантаження;
- можливість роботи як у локальній, так і віддаленій мережі.

Недоліки:

- висока складність схеми та необхідність ретельного налаштування STM32;
- відсутність інтеграції з IoT-платформами або мобільними додатками.

У публікації автора Tsai K.-L. [2] описується система управління енергоспоживанням на базі бездротової розетки, яка є складовою більш широкої IoT-інфраструктури. Основна ідея – передача даних на централізований сервер, де відбувається аналіз споживання електроенергії та формування керуючих сигналів.

Переваги:

- повна інтеграція з хмарними сервісами;
- можливість моніторингу та керування через мобільний додаток;
- ефективний алгоритм оптимізації енергоспоживання.

Недоліки:

- залежність від постійного інтернет-з'єднання;
- відсутність автономного режиму роботи;
- складність реалізації з боку кінцевого користувача.

У роботі [3] під керівництвом Al-Hassan E. запропоновано вдосконалену модель розумної розетки, головною особливістю якої є поєднання функцій моніторингу та захисту побутових приладів. Автори акцентують увагу на захисті від короткого замикання, перенапруги та перегріву. Крім того, реалізована підтримка Bluetooth для локального керування.

Переваги:

- надійний захист електроприладів;
- підтримка двох режимів підключення: Bluetooth і Wi-Fi;
- простота налаштування для кінцевого користувача.

Недоліки:

- обмежена дальність зв’язку при використанні Bluetooth;
- відсутність аналітичних функцій або оптимізації енергоспоживання.

Для зручності аналізу створено порівняльну таблицю (табл. 1.1) характеристик розглянутих вище робіт.

Таблиця 1.1 – Порівняльна таблиця характеристик

Характеристика	Розумна розетка на STM32	Система управління смарт-розеткою як IoT	Смарт-розетка контролю домашніх пристроїв
Мікроконтролер	STM32	Не вказано (власна розробка)	ARM Cortex-M4

Продовження таблиці 1.1

Спосіб підключення	Wi-Fi (ESP8266)	Wi-Fi (з IoT-платформою)	Wi-Fi + Bluetooth
Вимірювання параметрів електромережі	Так	Так	Так
Захист від перевантажень / короткого замикання	Так	Ні	Так
Керування через мобільний додаток	Ні	Так	Так
Підтримка автономної роботи	Частково	Ні	Так
Складність впровадження	Висока	Висока	Середня
Інтеграція з IoT	Низька	Висока	Середня

Проведений аналіз засвідчує, що всі три рішення мають певні переваги, однак жодне з них не поєднує одночасно простоту впровадження, багатофункціональність та гнучкість керування. Наприклад, рішення на STM32 є технічно досконалим, але складним для масового застосування. Робота з системою управління на основі Інтернету речей орієнтована на масштабну IoT-інфраструктуру, але не враховує локальні обмеження. Рішення смарт-розетки для контролю побутових електроприладів є більш збалансованим, але з дещо обмеженим функціоналом.

Таким чином, у розробці власної системи доцільно врахувати такі аспекти:

– застосування простого мікроконтролера з достатніми обчислювальними можливостями;

– реалізація базових функцій захисту та енергомоніторингу;

– простота підключення і налаштування для користувача.

1.2 Постановка завдання

На основі проведеного аналізу існуючих технічних рішень виявлено, що сучасні моделі розумних розеток або є надто складними для реалізації пересічним користувачем, або мають обмежений функціонал. Це створює підґрунтя для розробки нової моделі, яка поєднує доступність, гнучкість та необхідний набір функцій для безпечного та ефективного керування електроживленням побутових приладів.

Мета роботи – розробити розумну розетку з дистанційним керуванням, яка забезпечує базові функції моніторингу, захисту та керування живленням електроприладів.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1) провести аналіз існуючих технічних рішень у сфері розумних розеток;
2) обґрунтувати вибір мікроконтролера та інших компонентів;
3) розробити принципову електричну схему пристрою;
4) створити програмне забезпечення для пристрою;
5) протестувати працездатність системи та оцінити її ефективність у типових сценаріях використання.

Реалізація зазначених завдань дозволить створити компактний, зручний у використанні та функціонально збалансований пристрій, здатний задовольнити потреби користувача у безпечному та гнучкому керуванні електроживленням.

Запропоноване рішення має бути простим у налаштуванні та придатним для використання в умовах типового житла чи офісу.

1.3 Обґрунтування проектного рішення

Для реалізації пристрою розумної розетки з дистанційним керуванням прийнято рішення використати сучасну апаратну платформу, яка забезпечує стабільну роботу, високу інтеграцію та можливість розширення функціоналу без значних змін в архітектурі системи. Всі компоненти підібрані з урахуванням критеріїв енергоефективності, доступності, компактності, безпеки та зручності програмної інтеграції.

Центральною частиною пристрою є ESP32-WROOM-32 (рис 1.1) – високопродуктивний 32-бітний мікроконтролер [4] із підтримкою Wi-Fi та Bluetooth. Він оснащений двоядерним процесором, має велику кількість цифрових та аналогових входів/виходів, підтримує PWM, SPI, I²C, UART, а також вбудовані таймери та АЦП. Завдяки вбудованому модулю бездротового зв’язку, ESP32 дозволяє реалізувати повноцінне дистанційне керування розеткою через мобільний застосунок або веб-інтерфейс без потреби в додаткових комунікаційних модулях. Крім того, мікроконтролер має низьке енергоспоживання у режимі очікування, що важливо для постійно увімкнених пристроїв.

Рисунок 1.1 – ESP32-WROOM-32

Загальна схема пристрою та усі зв’язки наведено на рисунку 1.2.

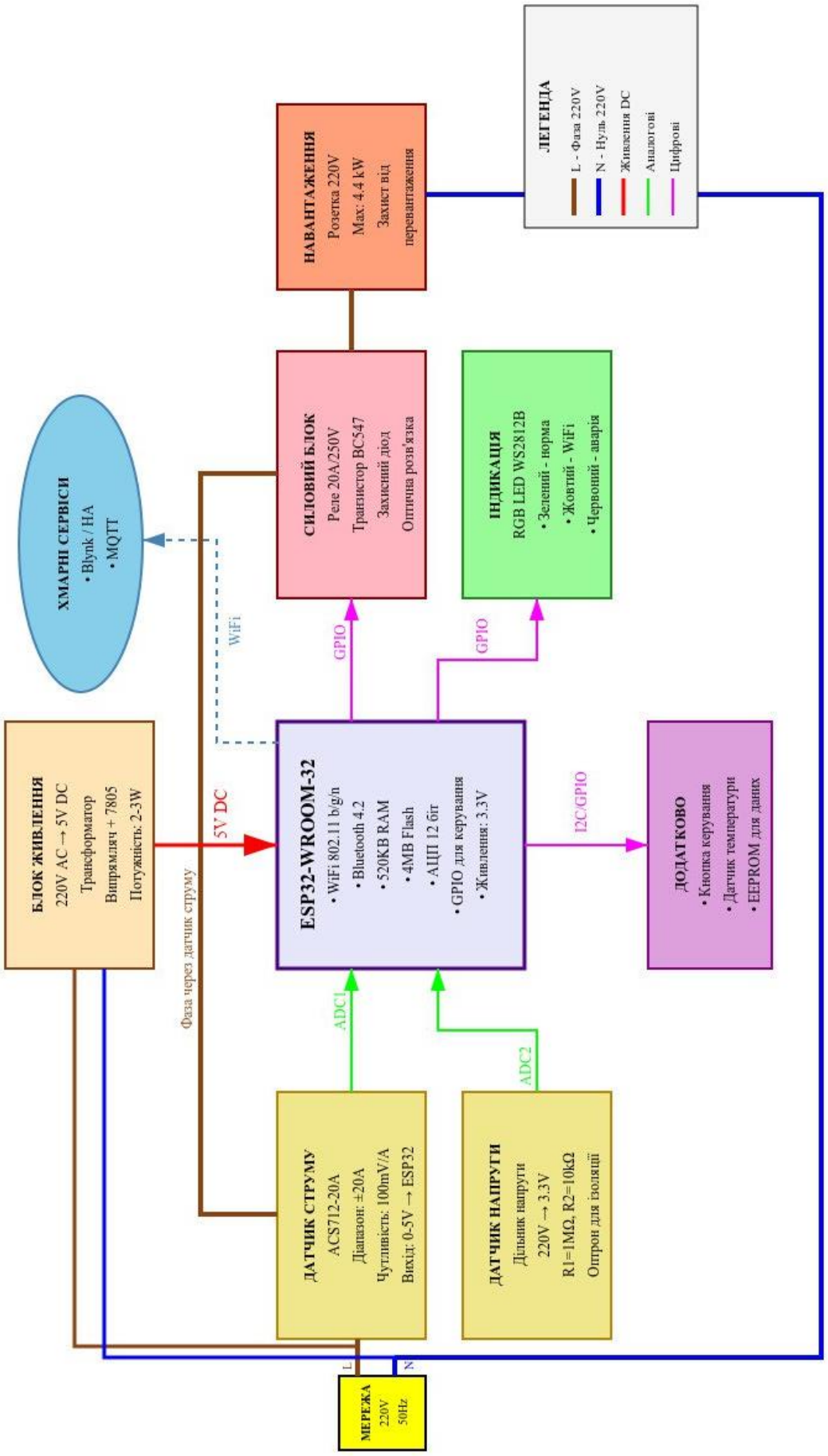

Рисунок 1.2 – Загальна схема пристрою

ESP32 обрано як основу системи через наступні переваги: вбудований Wi-Fi модуль (802.11 b/g/n), двоядерний процесор 240 МГц, 520 КБ оперативної пам’яті, 34 GPIO піни, 12-бітний АЦП, низьке енергоспоживання, доступна ціна.

У таблиці 1.2 наведено порівняння обраного мікроконтролера з доступними аналогами:

Таблиця 1.2 – Порівняння з аналогами

Характеристика	ESP32	Arduino Uno + WiFi Shield	Raspberry Pi
Ціна	$5-10	$25-40	$35-60
Споживання	80-240 мА	300-500 мА	700-1000 мА
GPIO	34	14	40
АЦП	18 каналів	6 каналів	Відсутній
Розміри	Компактний	Великий	Великий

Для моніторингу споживаного струму використано датчик ACS712ELCTR-20A-T (рис. 1.3), який забезпечує точне вимірювання змінного струму до 20 А. Сенсор [5] має гальванічну розв’язку, що гарантує безпеку електроніки, та аналоговий вихід, який передає сигнал на вхід АЦП ESP32. Датчик забезпечує лінійний вихідний сигнал зі зміщенням відносно половини напруги живлення (2.5 В), що дозволяє точно розраховувати значення струму в обох напрямках.

Живлення від мережі 220 В реалізовано через модуль HLK-10M05, який виконує функцію імпульсного перетворювача змінної напруги в стабілізовану постійну 5 В. Цей модуль є промисловим рішенням з високим ККД, гальванічною розв’язкою та вбудованими захистами. Для додаткової стабілізації та подачі живлення на мікроконтролер використано лінійний стабілізатор AMS1117, який знижує напругу з 5 В до 3,3 В, необхідних для живлення ESP32.

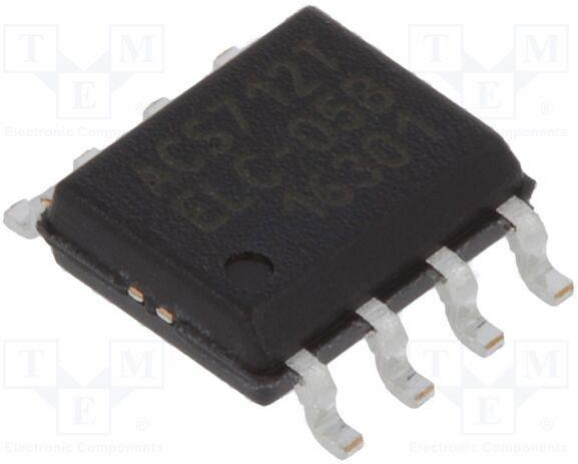

Рисунок 1.3 – ACS712ELCTR-05B-T

Для реалізації функції керування навантаженням застосовано електромагнітне реле типу JW2SN-DC5V, яке керується мікроконтролером через транзисторний ключ на основі BC547B. Транзистор увімкнено у ключовому режимі, а на його базу подається керуючий сигнал через резистор з одного з цифрових виходів ESP32. Така схема дозволяє вмикати або вимикати підключене навантаження (побутовий прилад), не перевантажуючи мікроконтролер. Для захисту транзистора від імпульсів зворотної електрорушійної сили, які виникають при вимкненні реле, встановлено захисний діод 1N4007 паралельно обмотці реле.

Для реалізації візуального зворотного зв’язку або розширення інтерфейсів взаємодії із користувачем схема передбачає підключення до ESP32 індикаторів, сенсорів або цифрових елементів через стандартні GPIO-піни. У разі необхідності можуть бути підключені датчики температури, вологоміри, сенсори руху або сенсорні кнопки.

Підібрана конфігурація компонентів дозволяє реалізувати пристрій, який:

- забезпечує стабільне живлення та захист електроніки;
- здійснює точний контроль сили струму в навантаженні;
- має дистанційне керування за допомогою Wi-Fi;

- підтримує розширення функціоналу та модернізацію;
- відповідає вимогам безпеки та надійності при роботі з мережею 220В;
- має компактні габарити завдяки інтегрованим рішенням.

Загальна схема пристрою орієнтована на застосування в побутових умовах, а також в умовах офісу або майстерні, де потрібне гнучке та безпечне керування електроживленням. Структура рішення дозволяє легко адаптувати систему до потреб користувача – зокрема, підключити хмарну аналітику, реалізувати розклад увімкнень або автоматичну реакцію на зміну навантаження.

Для забезпечення зворотного зв’язку з користувачем у пристрої реалізовано індикацію стану за допомогою RGB-світлодіода WS2812B. Він керується через цифровий вихід ESP32 та дозволяє виводити кольорові сигнали про поточний стан системи: зелений – нормальна робота, жовтий – підключення до Wi-Fi, червоний – аварійна ситуація або перевантаження. Така індикація підвищує зручність у використанні та дозволяє візуально контролювати статус пристрою без підключення до застосунку.

Також архітектура системи передбачає можливість підключення до хмарних сервісів через вбудований Wi-Fi-модуль мікроконтролера ESP32. Варіантами є такі платформи, як Blynk, Home Assistant (HA), MQTT-брокери, що дозволяє здійснювати повноцінний моніторинг і керування пристроєм із мобільного телефону або через інтернет. Це розширює сферу застосування розетки та робить її сумісною з системами «розумного дому» різного рівня складності.

Таким чином, загальна структура пристрою забезпечує не лише базові функції контролю струму та керування навантаженням, але й підтримку сучасних мережевих технологій, що дозволяє інтегрувати систему в автоматизовані середовища, підвищуючи її цінність і гнучкість у реальному застосуванні.

2 ПРАКТИЧНА ЧАСТИНА

2.1 Розробка електричної принципової схеми

У даному підрозділі представлено електричну принципову схему, яка є основою функціонування розумної розетки з дистанційним керуванням. Система побудована на базі сучасного мікроконтролера ESP32-WROOM-32, сенсора струму ACS712 та елементів силової електроніки для комутації навантаження. У таблиці 2.1 наведено перелік основних компонентів схеми та їх кількість.

Таблиця 2.1 – Перелік компонентів

№	Назва	Кількість
1	Модуль живлення HLK-10M05 (220В → 5В)	1 шт
2	Лінійний стабілізатор AMS1117-3.3	1 шт
3	Мікроконтролер ESP32-WROOM-32	1 шт
4	Датчик струму ACS712ELCTR-20A-T	1 шт
5	Реле JW2SN-DC5V	1 шт
6	Транзистор BC547B	1 шт
7	Діод 1N4007	1 шт
8	Резистор базовий (1 кОм)	1 шт
9	Джерело змінної напруги 220 В	1 шт
10	Навантаження (лампа або інший пристрій)	1 шт

Підключення пристрою (рис. 2.1) здійснюється безпосередньо до мережі змінного струму 220 В. Для зниження напруги та забезпечення живлення електронної частини використовується імпульсний модуль HLK-10M05, який забезпечує стабільні 5 В постійного струму. Завдяки своїй компактності та ізоляції цей модуль є безпечним та ефективним рішенням для вбудованих систем.

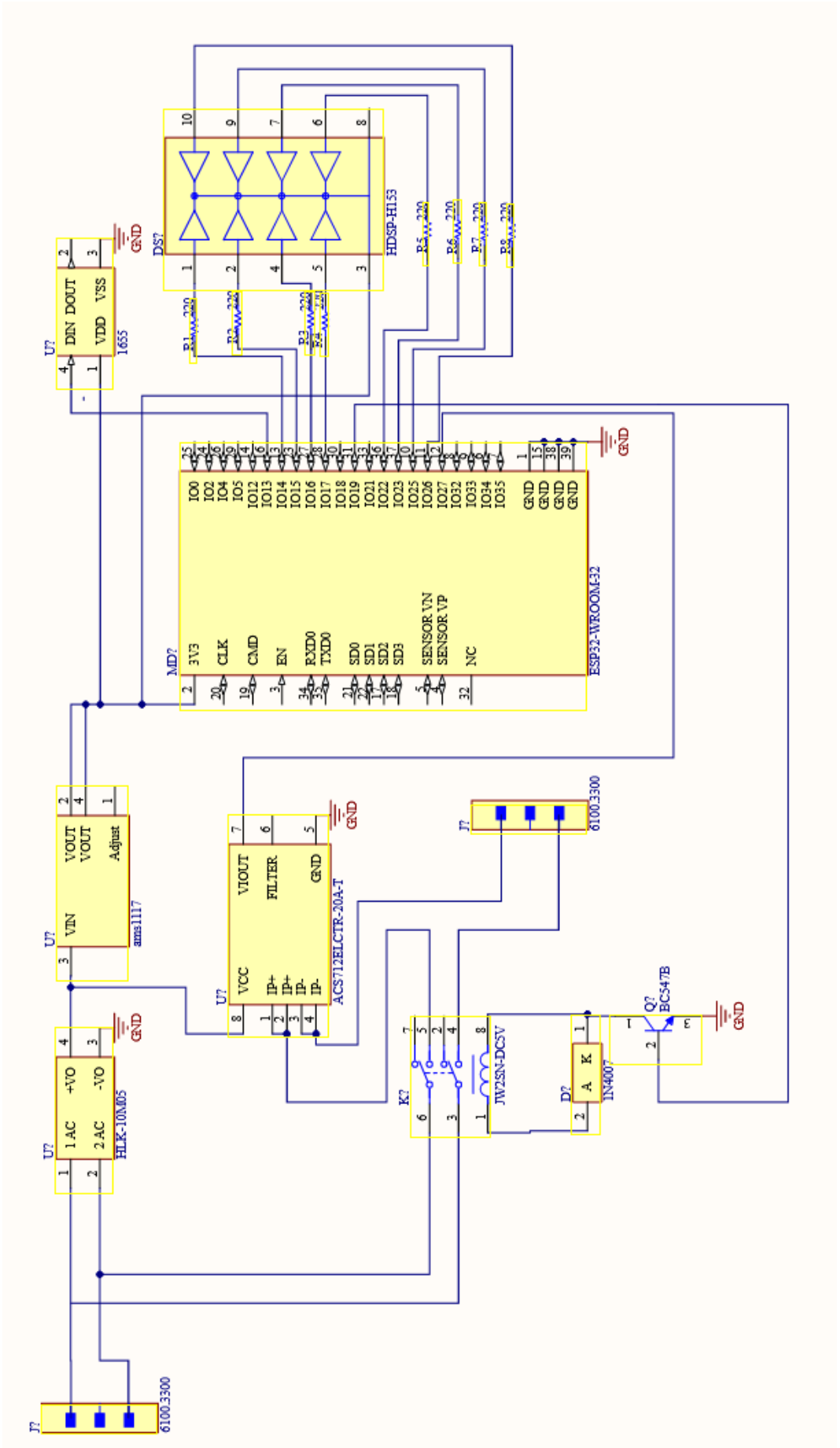

Рисунок 2.1 – Принципова схема підключення

Для живлення мікроконтролера ESP32, який працює при 3,3 В, у схемі додатково використано лінійний стабілізатор AMS1117, що знижує напругу з 5 В до 3,3 В. Цей стабілізатор має вбудований тепловий захист і дозволяє забезпечити надійну подачу живлення на цифрову логіку пристрою.

ESP32-WROOM-32 виконує роль головного обчислювального елемента. Він забезпечує обробку сигналу з сенсора струму, керування реле та обмін даними з користувачем через інтерфейс Wi-Fi. До аналогового входу ESP32 підключено вихід сенсора струму для зчитування поточного значення струму в навантаженні.

ACS712ELCTR-20A-T використовується для вимірювання сили змінного струму в колі навантаження. Сенсор має гальванічну розв’язку, що гарантує безпеку для мікроконтролера та інших компонентів. Аналоговий сигнал, пропорційний струму, подається на АЦП ESP32. Таким чином, система може контролювати рівень струму в реальному часі, а також реалізувати функції попередження чи захисту від перевантаження.

Для замикання чи розмикання електричного кола використовується електромагнітне реле JW2SN-DC5V. Його керування реалізується за допомогою NPN-транзистора BC547B, який працює в ключовому режимі. База транзистора підключена до цифрового виходу ESP32 через резистор номіналом 1 кОм, що обмежує струм бази до безпечного рівня. При подачі логічної «1» на базу транзистора він відкривається, і через коло реле протікає струм, що активує комутацію.

Для захисту транзистора від зворотної ЕРС, яка виникає при розмиканні обмотки реле, використовується захисний діод 1N4007, включений паралельно до обмотки реле у зворотному напрямку.

У якості навантаження до реле може підключатися будь-який побутовий пристрій, наприклад, лампа, обігрівач або блок живлення. Комутація здійснюється через нормально розімкнуті контакти реле. Увімкнення або вимкнення навантаження відбувається за командою від ESP32 відповідно до отриманих даних від сенсора струму або від користувача.

2.2 Програмна реалізація

У цьому підрозділі розглянуто програмну частину реалізації пристрою, зокрема структуру коду, основні функціональні блоки та алгоритми взаємодії мікроконтролера з підключеними компонентами. Повний код наведено у додатку А.

Програма побудована за модульним принципом (рис. 2.2):

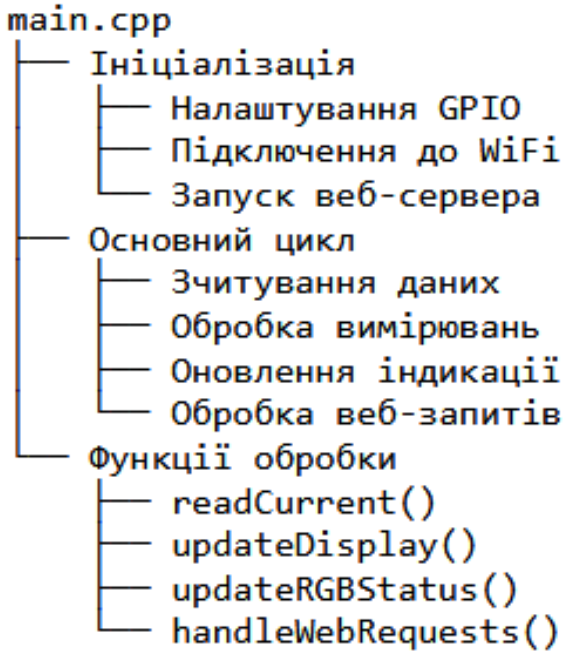

Рисунок 2.2 – Структура програми

Алгоритм вимірювання струму наведено в лістингу 2.1. Для точного вимірювання змінного струму використовується метод RMS (середньоквадратичне значення):

```
float readCurrent() {
     const int numSamples = 100;
     float sum = 0;
for (int i = 0; i < numSamples; i++) {
```

Лістинг 2.1 – Алгоритм вимірювання струму, аркуш 1

```
    int sensorValue = analogRead(CURRENT_PIN);
    float voltage = (sensorValue / 4095.0) * 3.3;
    float current = (voltage - 2.5) / 0.066;
    sum += current * current;
    delayMicroseconds(200);
}
    float rms = sqrt(sum / numSamples);
    return rms;
}
```

Лістинг 2.1 – Алгоритм вимірювання струму, аркуш 2

Алгоритм виконує 100 вимірювань протягом 20 мс (один період 50 Гц), обчислює квадрат кожного значення, усереднює та витягує квадратний корінь.

Розрахунок потужності та енергії відбувається наступним чином. Потужність розраховується за формулою:

$$P = I \times U \tag{2.1}$$

Енергія накопичується інтегруванням потужності за часом:

$$E = \int P dt \approx \Sigma(P \times \Delta t) \tag{2.2}$$

Система має два режими роботи, а саме:

1) автоматичний режим:
   – вмикання реле при I > 1.0 А,
   – вимикання реле при I < 0.5 А,
   – аварійне вимикання при I > порогу;
2) ручний режим:
   – керування реле через веб-інтерфейс,
   – збереження стану при перезавантаженні.

Індикацію стану забезпечує семисегментний індикатор та RGB-світлодіод. Перший почергово кожні 3 секунди відображає такі

характеристики як струм (0-9,9 А) та потужність (0-990 Вт), які відповідно знаходяться зліва та справа.

RGB-світлодіод у свою чергу має наступні стани:

– зелений – нормальна робота;
– жовтий – помилка WiFi з’єднання;
– червоний – аварія (перевищення струму).

Для забезпечення віддаленого керування пристроєм реалізовано REST API [6], який надає зручні HTTP-ендпоїнти для взаємодії з системою. Це дозволяє стороннім додаткам або користувачам отримувати стан системи, керувати реле, змінювати режим роботи, а також налаштовувати пороги та параметри живлення.

У таблиці 2.2 подано перелік доступних ендпоїнтів:

Таблиця 2.2 – HTTP ендпоінти

Метод	URL	Опис
GET	/	Головна сторінка
GET	/api/status	JSON статус системи
GET	/relay/on	Увімкнути реле
GET	/relay/off	Вимкнути реле
GET	/mode/toggle	Перемкнути режим
POST	/threshold/set	Встановити поріг
GET	/energy/reset	Скинути лічильник
POST	/voltage/set	Встановити напругу

Реалізація такого API забезпечує гнучкість у використанні пристрою та можливість його інтеграції з іншими системами, зокрема у середовищах розумного дому або промислових автоматизованих рішеннях.

У системі реалізовано базову обробку типових помилкових ситуацій для забезпечення стабільної та безпечної роботи пристрою. Зокрема, передбачено такі механізми:

– втрати Wi-Fi з’єднання: у разі розриву з’єднання модуль автоматично ініціює спроби повторного підключення до мережі;

– перевищення допустимого струму: у випадку виявлення надструму система негайно вимикає реле для запобігання пошкодженням обладнання;

– некоректні або зашумлені дані з АЦП: перед подальшою обробкою всі показники проходять фільтрацію та усереднення, що знижує вплив випадкових сплесків і забезпечує достовірність вимірювань.

2.3 Веб-інтерфейс

Веб-інтерфейс розроблено для моніторингу та керування пристроєм на базі ESP32 у реальному часі. Він побудований у вигляді Single Page Application (SPA), що дозволяє динамічно оновлювати дані без необхідності перезавантаження всієї сторінки. Кожні 5 секунд здійснюється автоматичне оновлення інформації про стан системи, що забезпечує актуальність відображених параметрів.

Такий підхід суттєво підвищує зручність користування, зменшує навантаження на мережу та покращує взаємодію користувача з системою в умовах обмеженого або нестабільного інтернет-з’єднання.

Інтерфейс має просту та зрозумілу структуру, яка складається з таких основних блоків (лістинг 2.2):

```
<!DOCTYPE html><html>
<head>
     <title>ESP32 Control</title>
     <meta charset='utf-8'>
     <meta name='viewport' content='width=device-width, initial-
     scale=1'>
</head>
<body>
     <div class='container'>
```

Лістинг 2.2 – Структура веб-інтерфейсу, аркуш 1

```
    <h1>ESP32 Current Monitor</h1>
    <div class='status'>SYSTEM STATUS</div>
    <div class='data'>Current: X.XX A</div>
    <div class='data'>Power: XXX W</div>
    <div class='data'>Energy: X.XX Wh</div>
    <div class='controls'>
    <button>Relay ON</button>
    <button>Relay OFF</button>
    <button>Toggle Mode</button>
    </div>
    </div>
</body>
</html>
```

Лістинг 2.2 – Структура веб-інтерфейсу, аркуш 2

Для оформлення інтерфейсу використано адаптивний дизайн, що дозволяє коректно відображати сторінку на пристроях з різною роздільною здатністю. Основні особливості стилізації:

– максимальна ширина контейнера – 600px, що дозволяє зручно переглядати інтерфейс на мобільних пристроях;

– кольорова індикація статусу – зміна кольору елементів статусу залежно від стану системи (наприклад, зелений – активний, червоний – помилка);

– великі кнопки керування, що полегшують натискання на сенсорних екранах;

– автоматичне масштабування – забезпечує збереження пропорційного вигляду незалежно від розміру екрана.

Також реалізована функціональність веб-інтерфейсу на мові JavaScript. Основні задачі скриптів:

1) автоматичне оновлення сторінки кожні 5 секунд (лістинг 2.3):

```
setTimeout(() => location.reload(), 5000);
```

Лістинг 2.3 – Оновлення сторінки

Це дозволяє динамічно отримувати оновлені дані без втручання користувача.

2) AJAX-запити для керування (лістинг 2.4):

```
fetch('/relay/on').then(() => location.reload());
```

Лістинг 2.4 – AJAX-запити

Кнопки керування надсилають запити до відповідних REST-ендпоїнтів. Після виконання дії відбувається повторне завантаження сторінки для оновлення стану.

Інтерфейс оптимізовано для всіх основних типів пристроїв: смартфони (320px – 768px), планшети (768px – 1024px) та десктопи (>1024px). Реалізована адаптивна верстка дає змогу уникнути горизонтального прокручування та забезпечує зручне керування незалежно від типу пристрою. Кнопки масштабуються автоматично, а дані залишаються читабельними на будь-якому екрані.

Реалізація SPA веб-інтерфейсу на основі HTML, CSS та JavaScript забезпечує ефективне керування пристроєм та зручний моніторинг стану в реальному часі. У веб-інтерфейсі (рис.2.3) передбачено встановлення ліміту струму в межах 0,5–20 А.

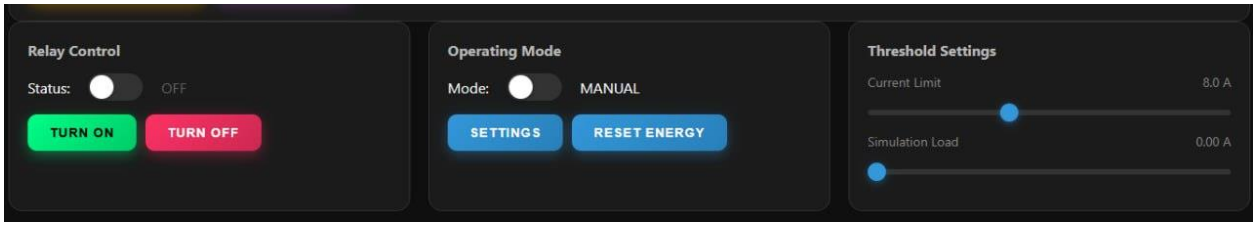

Рисунок 2.3 – Налаштування

У режимі автоматичного керування система контролює струм: якщо значення перевищує встановлений ліміт – розетка вимикається. У ручному

режимі контроль не здійснюється. Також інтерфейс містить кнопку керування живленням, яка дозволяє вручну вмикати або вимикати розетку за потреби.

У центральній частині інтерфейсу відображаються основні параметри: зліва – струм, справа – напруга, далі – потужність, а ще правіше – споживання електроенергії. У нижній частині сторінки розміщено графік для візуального моніторингу динаміки роботи пристрою (рис. 2.4). Нижче на рис. 2.5 наведено приклад різкого зниження потужності, що наочно відображено на графіку.

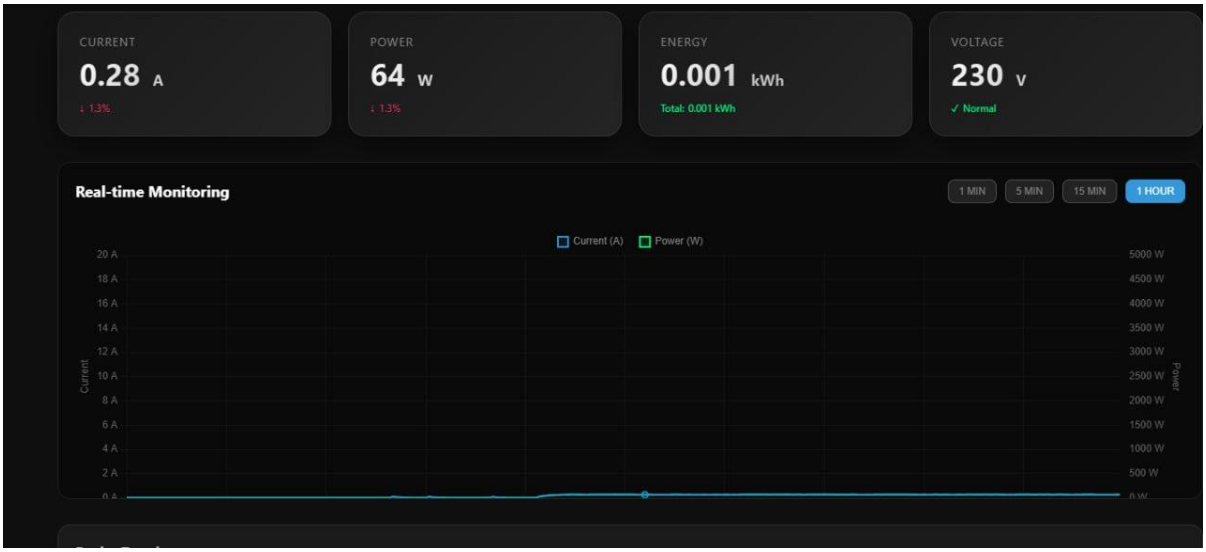

Рисунок 2.4 – Основні параметри

Рисунок 2.5 – Приклад різкого зниження потужності

Така візуалізація дозволяє швидко виявляти аномальні стани в роботі пристрою та реагувати на них у режимі реального часу.

Інтерфейс дозволяє задавати параметри навантаження безпосередньо на екрані. Наприклад, при ввімкненні пристрою потужністю 700 Вт (230 В та 3 А) система миттєво починає обчислення енергоспоживання, що відображається у відповідному полі (рис. 2.6).

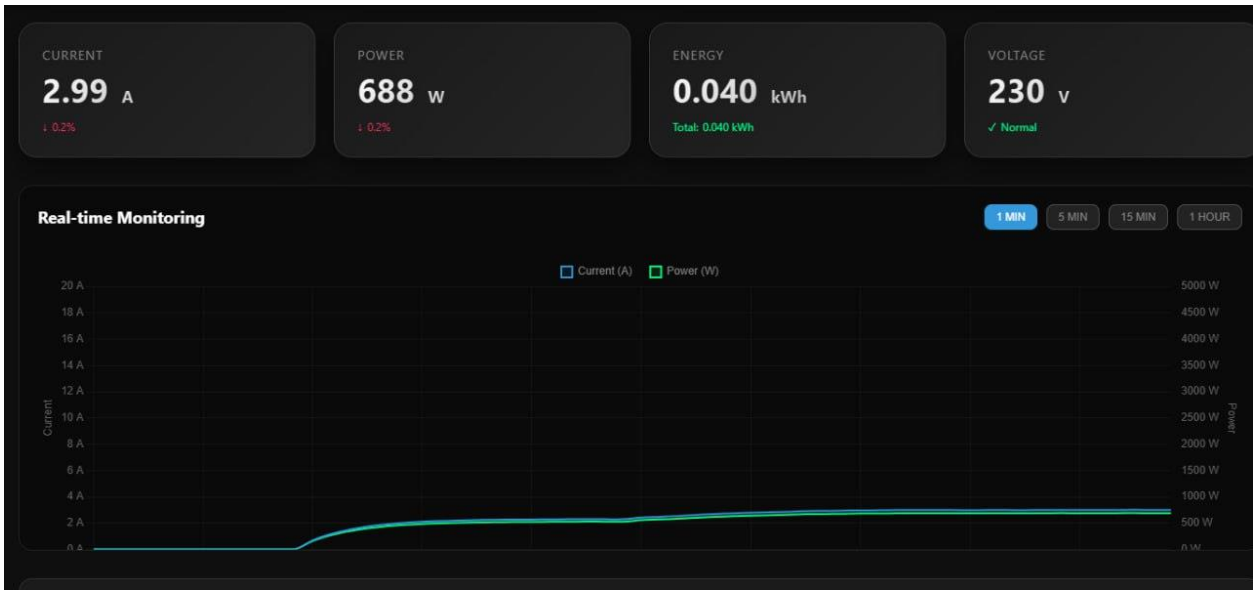

Рисунок 2.6 – Розрахунок енергоспоживання

У нижній частині інтерфейсу розміщена секція шаблонів пристроїв (темплейтів), що значно спрощує керування розеткою. Користувач може вибрати типове навантаження, наприклад: лампа, ноутбук, комп’ютер, зарядний пристрій тощо. Для кожного з варіантів автоматично підставляються типові параметри потужності, струму та напруги, що зручно при частому використанні одних і тих самих пристроїв.

Поруч із кожним шаблоном розміщені кнопки вмикання та вимикання розетки, що дозволяє швидко активувати чи знеструмити вибране навантаження. Такий підхід забезпечує інтуїтивне керування системою навіть без глибоких технічних знань. Перелік базових варіантів зображено на рисунку 2.7.

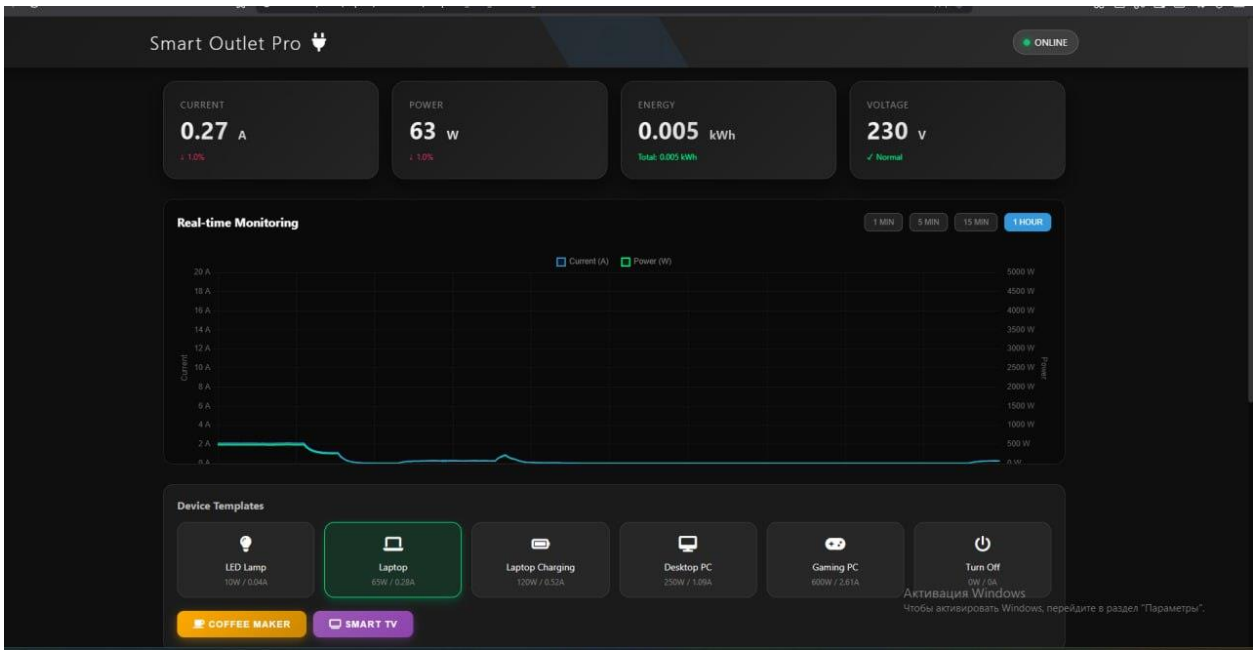

Рисунок 2.7 – Секція шаблонів пристроїв

Розроблений веб-інтерфейс став ключовим компонентом системи керування пристроєм на базі ESP32, забезпечивши зручний і ефективний засіб взаємодії користувача з апаратною частиною. Завдяки використанню концепції SPA, досягнуто високої швидкодії, мінімізації затримок при оновленні даних та загального покращення користувацького досвіду.

Інтерфейс дозволяє здійснювати постійний моніторинг основних електричних параметрів у режимі реального часу, таких як струм, напруга, потужність і загальне споживання електроенергії. Автоматичне оновлення даних кожні 5 секунд реалізовано за допомогою JavaScript-скриптів, що забезпечує безперервну актуалізацію відображуваної інформації без необхідності ручного оновлення сторінки.

Особливу увагу приділено адаптивності дизайну, що дозволяє використовувати веб-інтерфейс на різних пристроях – від смартфонів до стаціонарних комп’ютерів. Завдяки чітко структурованій верстці, оптимізованій ширині контейнерів, масштабованим кнопкам керування та адаптивному розміщенню інформаційних блоків, система забезпечує зручність у використанні незалежно від розміру екрана користувача.

Кольорові індикатори стану системи дають змогу миттєво визначити її робочий стан, що особливо важливо у критичних або аварійних ситуаціях.

Інтерфейс передбачає два режими роботи – автоматичний та ручний. У автоматичному режимі реалізовано контроль за перевищенням струму: при виявленні значень, що виходять за межі встановленого ліміту (0,5–20 А), система автоматично знеструмлює розетку. Це значно підвищує безпеку експлуатації пристрою та захищає підключене обладнання від можливих пошкоджень. У ручному режимі керування віддається повністю користувачу – система не втручається в електричний ланцюг до моменту прямої команди.

Центральна частина інтерфейсу містить блоки для виведення поточних вимірювань, що дозволяє оперативно відслідковувати зміну електричних параметрів. Вбудований графік зміни потужності забезпечує візуальне уявлення про динаміку навантаження – наприклад, миттєве падіння споживання можна легко ідентифікувати, як показано на прикладі в інтерфейсі. Це суттєво полегшує аналіз роботи пристрою та діагностику можливих відхилень або збоїв.

Крім моніторингу, веб-інтерфейс підтримує можливість задання параметрів навантаження вручну. Важливою особливістю інтерфейсу стала секція шаблонів пристроїв – темплейтів, яка дозволяє користувачу обрати типове електронавантаження із попередньо визначеними параметрами. Це значно економить час і спрощує керування, особливо при повторному використанні одних і тих самих пристроїв.

Таким чином, розроблений веб-інтерфейс успішно поєднує функціональність, зручність, адаптивність і безпеку. Його використання суттєво розширює можливості апаратної частини проєкту, дозволяє здійснювати гнучке та оперативне керування системою, а також забезпечує якісний зворотний зв’язок у вигляді візуалізації та повідомлень. Такий підхід сприяє загальному підвищенню ефективності керування електроживленням у побутових та промислових умовах.

3 ВИПРОБУВАННЯ ПРИСТРОЮ

Метою тестування розробленої системи є перевірка її функціональності, точності, надійності та відповідності поставленим вимогам. Система повинна не лише точно виконувати вимірювання електричного струму, а й забезпечувати зручний та стабільний доступ до результатів вимірювань через веб-інтерфейс, працювати надійно в реальних умовах експлуатації та своєчасно реагувати на аварійні ситуації.

Для досягнення повноти перевірки проведено три типи тестувань:

1) тестування точності вимірювань;
2) тестування веб-інтерфейсу;
3) тестування надійності системи.

Кожен з підтипів тестування передбачав серію окремих експериментів із фіксацією результатів та аналізом похибок.

Точність вимірювання сили струму перевірена шляхом порівняння показів системи з еталонним лабораторним амперметром. Випробування проводилися при різних навантаженнях, що охоплюють типовий діапазон експлуатації – від 0,5 до 15,0 ампер.

Окремо перевірялася лінійність – тобто, наскільки похибка залишається сталою або пропорційною у всьому діапазоні вимірювання. Всі результати вимірювань внесені у таблицю 3.1 для подальшого аналізу.

Таблиця 3.1 – Точність вимірювань

Еталонне значення (А)	Виміряно (А)	Похибка (%)
0.5	0.48	4.0
1.0	0.97	3.0
5.0	4.92	1.6
10.0	9.85	1.5
15.0	14.78	1.5

Середня похибка: 2.3%.

Тестування веб-інтерфейсу дало можливість оцінити стабільність, функціональність та швидкодію веб-інтерфейсу користувача, а також його сумісність з різними платформами.

Тестування проводилося на популярних веб-браузерах: Google Chrome, Mozilla Firefox, Microsoft Edge. Перевірялися наступні аспекти:

– коректне відображення інтерфейсу;

– можливість виконання запитів на читання даних;

– відповідь системи на стрес-навантаження (100+ запитів за секунду);

– сумісність з мобільними пристроями.

Інтерфейс продемонстрував стабільну роботу в усіх зазначених браузерах. При стрес-тестуванні система витримувала понад 100 запитів на секунду без зависань чи втрати з'єднання. Час відгуку на веб-запит становив 15–25 мс, що є надзвичайно хорошим показником для вбудованих IoT-пристроїв.

Тестування надійності дало можливість оцінити стабільність роботи пристрою в довготривалому режимі, при зміні мережевих умов та перевантаженнях по струму. Проведено три типи перевірок.

1) Тривале тестування – пристрій працював безперервно протягом 72 годин.

2) Зміна мережевого середовища – здійснювалося періодичне вимкнення та вмикання Wi-Fi-з'єднання, перевірялася реакція системи на втрату з'єднання.

3) Імітація перевантаження – навантаження збільшувалося вище номінального значення, відстежувалася робота аварійного захисту.

Протягом 72 годин система працювала стабільно, без зависань чи потреби у перезапуску. При втраті Wi-Fi-з'єднання модуль ESP32 автоматично перепідключався після відновлення доступу до мережі. Аварійний захист спрацьовував за менш ніж 100 мс, що дозволяє ефективно захистити підключені пристрої від перевантаження.

У процесі тестування системи виявлено певні технічні недоліки, які могли вплинути на функціонування пристрою. Усі проблеми проаналізовані, після чого впроваджені відповідні технічні рішення для їх усунення. Основні з них наведено далі.

1) Шуми аналогово-цифрового перетворювача (АЦП).

Під час вимірювання малих значень струму спостерігалися коливання результатів, викликані шумами на вході АЦП. Це призводило до нестабільних та неточних показів.

Рішення: реалізовано програмну фільтрацію результатів вимірювання шляхом усереднення кількох послідовних вимірів. Застосовано ковзне середнє з вікном у 10 вимірів, що дозволило зменшити вплив випадкових шумів без суттєвої втрати швидкодії. Завдяки цьому точність вимірювань при низьких навантаженнях помітно покращилася.

2) Нестабільність Wi-Fi-з'єднання.

У деяких випадках модуль ESP32 втрачав підключення до Wi-Fi, особливо при слабкому сигналі або зміні точки доступу. Без додаткових механізмів це призводило до «зависання» веб-інтерфейсу.

Рішення: впроваджено автоматичне перепідключення до Wi-Fi при втраті з’єднання. Крім того, в системі передбачено періодичну перевірку стану підключення. Якщо підключення виявляється неактивним — ініціюється повторне з'єднання без потреби в перезапуску пристрою. Це значно підвищило стійкість до мережевих збоїв.

3) Дрейф нуля.

АЦП та сенсор струму схильні до дрейфу нульового рівня, зумовленого температурними або живильними відхиленнями. Унаслідок цього навіть за відсутності навантаження система могла показувати хибні міліамперні значення.

Рішення: додано процедуру калібрування при запуску. У момент ініціалізації (коли навантаження відсутнє) система зчитує «нульове» значення з АЦП і зберігає його як базовий рівень. Подальші вимірювання коригуються

з урахуванням цього зсуву. Це дозволило усунути помилки в початкових вимірах і забезпечити точність навіть при дуже малих навантаженнях.

Для реалізації пристрою використані доступні, масово поширені електронні компоненти, сумарна вартість яких наведена у таблиці 3.2.

Таблиця 3.2 – Вартість компонентів системи

Компонент	Кількість	Ціна (грн)	Сума (грн)
ESP32-WROOM-32	1	150	150
ACS712ELCTR-20A	1	120	120
HDSP-H153 (індикатор)	1	80	80
HLK-10M05 (живлення)	1	100	100
Реле JW2SN-DC5V	1	30	30
Інші компоненти (резистори, корпус, друкована плата тощо)	—	—	70
Разом			550 грн

Ця вартість включає всі необхідні елементи для збирання повнофункціонального пристрою. На ринку існує низка комерційних рішень для моніторингу енергоспоживання, зокрема:

– промислові моніторингові системи (наприклад, Schneider Electric, Siemens, Omron) – їхня ціна починається від 5000 грн і може сягати десятків тисяч гривень у залежності від комплектації, наявності дисплея, з’єднання через Ethernet/RS485 та інших функцій;

– розумні розетки з моніторингом споживання (TP-Link, Xiaomi, Sonoff) – вартість коливається від 700 до 1500 грн, але більшість із них не дозволяє переглядати струм у реальному часі з точністю, яку забезпечує розроблена система, або мають обмежену інтеграцію з локальними серверами.

Таким чином, запропонована система обходиться щонайменше у 10 разів дешевше за базові промислові рішення при збереженні ключової функціональності (табл. 3.3).

Таблиця 3.3 – Функціональні критерії

Критерій	Комерційна система	Розроблена система
Орієнтовна вартість	5000+ грн	550 грн
Вимірювання струму	Так	Так
Веб-інтерфейс	Не завжди	Так
Гнучкість налаштування	Обмежена	Повна
Відкритий код / модифікація	Ні	Так

Це робить пристрій привабливим для малих підприємств, домашніх лабораторій, побутових користувачів та освітніх проєктів. Залежно від обсягів енергоспоживання, середня окупність системи становить 3–6 місяців. Наприклад:

– у разі щомісячного рахунку за електроенергію в 600 грн – 15% економії становлять 90 грн/міс., а окупність – близько 6 місяців;

– у разі промислового використання (наприклад, підприємство з рахунком 2000 грн/міс.) – економія 300 грн/міс., а окупність – менш ніж за 2 місяці.

Отже, система пройшла комплексне тестування, яке охоплює ключові аспекти: точність, зручність та стабільність. Всі перевірки підтвердили високу якість реалізації апаратної та програмної частини. Подальше вдосконалення можливе в напрямку зменшення похибки при низьких навантаженнях, а також розширення функціоналу інтерфейсу користувача. Крім технічних переваг, система також є економічно доцільною: її собівартість у десятки разів нижча за промислові аналоги, а можливість зниження споживання електроенергії забезпечує швидку окупність. Таким чином, рішення є не лише ефективним, а й фінансово вигідним для широкого кола користувачів – від побутових споживачів до малих підприємств.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досягнуто поставлену мету – розроблено розумну розетку з дистанційним керуванням, яка забезпечує базові функції моніторингу, захисту та керування живленням електроприладів. Створена система поєднує в собі технічну ефективність, надійність, енергоефективність і доступність, що робить її практично придатною для використання у повсякденному побуті.

У процесі виконання практики вирішено всі заплановані завдання:

1) проведено огляд сучасних технічних рішень у сфері розумних розеток;

2) обґрунтовано вибір апаратної складової, включно з мікроконтролером, сенсором струму та комутаційними елементами;

3) розроблено електричну принципову схему пристрою;

4) створено базове програмне забезпечення для керування пристроєм та реалізації захисних функцій;

5) протестовано працездатність системи та оцінено її ефективність.

Досягнуті результати:

– точність вимірювання струму: ±2.3%;

– діапазон вимірювання: 0-20 А;

– дистанційне керування через WiFi;

– автоматичний захист від перевантаження;

– облік спожитої енергії.

Переваги розробленої системи:

– низька вартість;

– простота встановлення;

– відкритий вихідний код;

– можливість розширення функціоналу.

Результати роботи демонструють, що розроблена розумна розетка може ефективно застосовуватись у таких сферах, як побутові житлові приміщення,

офіси, майстерні, лабораторії, серверні кімнати тощо – у всіх ситуаціях, де потрібне гнучке та безпечне керування електроживленням.

Практичне застосування роботи можливе як у вигляді окремого пристрою для моніторингу споживання в квартирах, захисту обладнання від перевантаження, обліку енергії для окремих споживачів, так і в складі більших систем «розумного дому». Завдяки відкритій архітектурі та простоті програмного забезпечення, розробку можна адаптувати для підключення до бездротових модулів, хмарних сервісів або мобільних додатків, що розширює її функціональні можливості.

Основними напрямками подальшого розвитку системи є:

– додавання підтримки трифазних мереж;

– інтеграція з системами «розумного дому»;

– впровадження алгоритмів енергозбереження та історії споживання.

Отримані результати мають потенціал для подальшої наукової та прикладної реалізації, зокрема у співпраці з підприємствами, що спеціалізуються на електроніці або системах автоматизації побуту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zhang M, Du J. Design and development of smart socket based on STM32. Journal of Computational Methods in Sciences and Engineering. 2020;20(1):227-243. doi:10.3233/JCM-193761.

2. K. -L. Tsai, F. -Y. Leu and I. You, "Residence Energy Control System Based on Wireless Smart Socket and IoT," in IEEE Access, vol. 4, pp. 2885-2894, 2016, doi: 10.1109/ACCESS.2016.2574199.

3. E. Al-Hassan, H. Shareef, M. M. Islam, A. Wahyudie and A. A. Abdrabou, "Improved Smart Power Socket for Monitoring and Controlling Electrical Home Appliances," in IEEE Access, vol. 6, pp. 49292-49305, 2018, doi: 10.1109/ACCESS.2018.2868788.

4. Espressif Systems. ESP32-WROOM-32 Datasheet [Електронний ресурс] – Режим доступу: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (дата звернення: 10.05.2025).

5. Allegro MicroSystems. ACS712 Current Sensor Datasheet [Електронний ресурс]. – 2020. – 15 с. – Режим доступу: https://www.allegromicro.com/-/media/files/datasheets/acs712-datasheet.pdf (дата звернення: 10.05.2025).

6. REST API Design Best Practices [Електронний ресурс]. – Режим доступу: https://restfulapi.net/ (дата звернення: 10.05.2025).

ДОДАТОК А

Програмний код

```
#include <WiFi.h>
#include <WebServer.h>
#include <ArduinoJson.h>
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
const int CURRENT_PIN = 27;
const int RELAY_PIN = 19;
const int RGB_RED = 13;
const int RGB_GREEN = 13;
const int RGB_BLUE = 13;
const int SEG_A = 14;
const int SEG_B = 15;
const int SEG_C = 16;
const int SEG_D = 17;
const int SEG_E = 22;
const int SEG_F = 23;
const int SEG_G = 25;
const int SEG_DP = 26;

const byte segmentPins[] = {SEG_A, SEG_B, SEG_C, SEG_D, SEG_E,
SEG_F, SEG_G};
const byte digitPatterns[] = {
  0b0111111,
  0b0000110,
  0b1011011,
  0b1001111,
  0b1100110,
  0b1101101,
  0b1111101,
  0b0000111,
  0b1111111,
  0b1101111,
  0b1110111,
  0b1111100,
  0b0111001,
  0b1011110,
  0b1111001,
  0b1110001
};
```

Лістинг А.1 – Програмний код, аркуш 1

```
WebServer server(80);

float currentAmps = 0;
float voltageRMS = 230.0;
float powerWatts = 0;
float energyWh = 0;
float maxCurrent = 10.0;
float alarmThreshold = 8.0;
bool relayState = false;
bool autoMode = true;
bool displayMode = false;
unsigned long lastUpdate = 0;
unsigned long lastEnergyUpdate = 0;
unsigned long wifiCheckInterval = 5000;
unsigned long lastWifiCheck = 0;
unsigned long displaySwitchTime = 0;

enum SystemStatus {
  STATUS_OK,
  STATUS_WIFI_ERROR,
  STATUS_ALARM
};

SystemStatus currentStatus = STATUS_OK;

void setup() {
  Serial.begin(115200);

  pinMode(CURRENT_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(RGB_RED, OUTPUT);
  pinMode(RGB_GREEN, OUTPUT);
  pinMode(RGB_BLUE, OUTPUT);

  for (int i = 0; i < 7; i++) {
    pinMode(segmentPins[i], OUTPUT);
  }
  pinMode(SEG_DP, OUTPUT);

  digitalWrite(RELAY_PIN, LOW);

  WiFi.begin(ssid, password);
  Serial.println("Connecting to WiFi...");
```

Лістинг А.1 – Програмний код, аркуш 2

```
  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    delay(500);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
    currentStatus = STATUS_OK;
  } else {
    Serial.println("\nFailed to connect to WiFi");
    currentStatus = STATUS_WIFI_ERROR;
  }
  lastEnergyUpdate = millis();

  setupWebServer();
  server.begin();
}
void setupWebServer() {
  server.on("/", HTTP_GET, []() {
    String html = "<!DOCTYPE html><html><head><title>ESP32
Control</title>";
    html += "<meta charset='utf-8'><meta name='viewport'
content='width=device-width, initial-scale=1'>";
    html += "<style>body{font-
family:Arial;margin:20px;background:#f0f0f0;}";
    html += ".container{max-width:600px;margin:0
auto;background:white;padding:20px;border-radius:10px;box-
shadow:0 2px 10px rgba(0,0,0,0.1);}";
    html += "h1{color:#333;text-align:center;}";
    html += ".status{padding:15px;margin:10px 0;border-
radius:5px;color:white;text-align:center;}";
    html +=
".ok{background:#27ae60;}.warning{background:#f39c12;}.alarm{bac
kground:#e74c3c;}";
    html += ".data{display:flex;justify-content:space-
between;margin:10px 0;padding:10px;background:#ecf0f1;border-
radius:5px;}";
    html += ".btn{padding:10px
20px;margin:5px;border:none;border-
radius:5px;cursor:pointer;font-size:16px;}";
```

Лістинг А.1 – Програмний код, аркуш 3

```
html += ".btn-on{background:#27ae60;color:white;}.btn-
off{background:#e74c3c;color:white;}";
    html += ".btn-auto{background:#3498db;color:white;}";
    html += "</style></head><body>";
    html += "<div class='container'><h1>ESP32 Current
Monitor</h1>";
    String statusClass = currentStatus == STATUS_OK ? "ok" :
(currentStatus == STATUS_WIFI_ERROR ? "warning" : "alarm");
    String statusText = currentStatus == STATUS_OK ? "SYSTEM OK"
: (currentStatus == STATUS_WIFI_ERROR ? "WIFI ERROR" : "CURRENT
ALARM");
    html += "<div class='status " + statusClass + "'>" +
statusText + "</div>";
    html += "<div class='data'><span>Current:</span><span>" +
String(currentAmps, 2) + " A</span></div>";
    html += "<div class='data'><span>Power:</span><span>" +
String(powerWatts, 1) + " W</span></div>";
    html += "<div class='data'><span>Energy:</span><span>" +
String(energyWh, 2) + " Wh</span></div>";
    html += "<div class='data'><span>Relay:</span><span>" +
String(relayState ? "ON" : "OFF") + "</span></div>";
    html += "<div class='data'><span>Mode:</span><span>" +
String(autoMode ? "AUTO" : "MANUAL") + "</span></div>";
    html += "<div class='data'><span>Threshold:</span><span>" +
String(alarmThreshold, 1) + " A</span></div>";
    html += "<div style='text-align:center;margin-top:20px;'>";
    html += "<button class='btn btn-on'
onclick='fetch(\"/relay/on\").then(()=>location.reload())'>Relay
ON</button>";
    html += "<button class='btn btn-off'
onclick='fetch(\"/relay/off\").then(()=>location.reload())'>Rela
y OFF</button>";
    html += "<button class='btn btn-auto'
onclick='fetch(\"/mode/toggle\").then(()=>location.reload())'>To
ggle Mode</button>";
    html += "</div>";
    html +=
"<script>setTimeout(()=>location.reload(),5000);</script>";
    html += "</div></body></html>";
    server.send(200, "text/html", html);
  server.on("/energy/reset", HTTP_GET, [] () {
    energyWh = 0;
    lastEnergyUpdate = millis();
    server.send(200, "text/plain", "OK");});
```

Лістинг А.1 – Програмний код, аркуш 4

```
server.on("/voltage/set", HTTP_POST, []() {
    if (server.hasArg("value")) {
      float newVoltage = server.arg("value").toFloat();
      if (newVoltage >= 100 && newVoltage <= 300) {
        voltageRMS = newVoltage;
        server.send(200, "text/plain", "OK");
      } else {
        server.send(400, "text/plain", "Invalid voltage");
      }
    } else {
      server.send(400, "text/plain", "Missing value");
    }
  });

  server.on("/api/status", HTTP_GET, []() {
    StaticJsonDocument<200> doc;
    doc["current"] = currentAmps;
    doc["power"] = powerWatts;
    doc["energy"] = energyWh;
    doc["voltage"] = voltageRMS;
    doc["relay"] = relayState;
    doc["mode"] = autoMode ? "auto" : "manual";
    doc["status"] = currentStatus == STATUS_OK ? "ok" :
(currentStatus == STATUS_WIFI_ERROR ? "wifi_error" : "alarm");
    doc["threshold"] = alarmThreshold;
    String json;
    serializeJson(doc, json);
    server.send(200, "application/json", json);
  });
  server.on("/relay/on", HTTP_GET, []() {
    if (!autoMode) {
      relayState = true;
      digitalWrite(RELAY_PIN, HIGH);
    }
    server.send(200, "text/plain", "OK");
  });

  server.on("/relay/off", HTTP_GET, []() {
    if (!autoMode) {
      relayState = false;
      digitalWrite(RELAY_PIN, LOW);
    }
    server.send(200, "text/plain", "OK");
  });
```

Лістинг А.1 – Програмний код, аркуш 5

```
server.on("/mode/toggle", HTTP_GET, []() {
    autoMode = !autoMode;
    server.send(200, "text/plain", autoMode ? "AUTO" :
"MANUAL");
  });

  server.on("/threshold/set", HTTP_POST, []() {
    if (server.hasArg("value")) {
      float newThreshold = server.arg("value").toFloat();
      if (newThreshold > 0 && newThreshold <= maxCurrent) {
        alarmThreshold = newThreshold;
        server.send(200, "text/plain", "OK");
      } else {
        server.send(400, "text/plain", "Invalid value");
      }
    } else {
      server.send(400, "text/plain", "Missing value");
    }
  });
}

float readCurrent() {
  const int numSamples = 100;
  float sum = 0;

  for (int i = 0; i < numSamples; i++) {
    int sensorValue = analogRead(CURRENT_PIN);
    float voltage = (sensorValue / 4095.0) * 3.3;
    float current = (voltage - 2.5) / 0.066;
    sum += current * current;
    delayMicroseconds(200);
  }
  float rms = sqrt(sum / numSamples);
  return rms;
}

void displayDigit(int digit) {
  if (digit < 0 || digit > 15) return;

  byte pattern = digitPatterns[digit];
  for (int i = 0; i < 7; i++) {
    digitalWrite(segmentPins[i], (pattern >> i) & 1);
  }
}
```

Лістинг А.1 – Програмний код, аркуш 6

```
void displayCurrent(float current) {
  int displayValue = (int)(current * 10);
  displayValue = constrain(displayValue, 0, 99);

  if (millis() % 1000 < 500) {
    displayDigit(displayValue / 10);
    digitalWrite(SEG_DP, HIGH);
  } else {
    displayDigit(displayValue % 10);
    digitalWrite(SEG_DP, LOW);
  }
}

void displayPower(float power) {
  int displayValue = (int)(power / 10);
  displayValue = constrain(displayValue, 0, 99);

  if (millis() % 1000 < 500) {
    displayDigit(displayValue / 10);
    digitalWrite(SEG_DP, LOW);
  } else {
    displayDigit(displayValue % 10);
    digitalWrite(SEG_DP, HIGH);
  }
}
void updateDisplay() {
  if (millis() - displaySwitchTime > 3000) {
    displaySwitchTime = millis();
    displayMode = !displayMode;
  }

  if (displayMode) {
    displayPower(powerWatts);
  } else {
    displayCurrent(currentAmps);
  }
}
void updateRGBStatus() {
  switch (currentStatus) {
    case STATUS_OK:
      analogWrite(RGB_RED, 0);
      analogWrite(RGB_GREEN, 255);
      analogWrite(RGB_BLUE, 0);
      break;
```

Лістинг А.1 – Програмний код, аркуш 7

```
    case STATUS_WIFI_ERROR:
      analogWrite(RGB_RED, 255);
      analogWrite(RGB_GREEN, 255);
      analogWrite(RGB_BLUE, 0);
      break;
    case STATUS_ALARM:
      analogWrite(RGB_RED, 255);
      analogWrite(RGB_GREEN, 0);
      analogWrite(RGB_BLUE, 0);
      break;
  }
}

void checkWiFiStatus() {
  if (millis() - lastWifiCheck > wifiCheckInterval) {
    lastWifiCheck = millis();

    if (WiFi.status() != WL_CONNECTED) {
      if (currentStatus != STATUS_ALARM) {
        currentStatus = STATUS_WIFI_ERROR;
      }
      WiFi.reconnect();
    } else if (currentStatus == STATUS_WIFI_ERROR) {
      currentStatus = STATUS_OK;
    }
  }
}
void loop() {
  server.handleClient();
  if (millis() - lastUpdate > 100) {
    lastUpdate = millis();

    currentAmps = readCurrent();
    powerWatts = currentAmps * voltageRMS;

    if (millis() - lastEnergyUpdate > 1000) {
      float hoursPassed = (millis() - lastEnergyUpdate) /
3600000.0;
      energyWh += powerWatts * hoursPassed;
      lastEnergyUpdate = millis();
    }
    if (currentAmps > alarmThreshold) {
                    currentStatus = STATUS_ALARM;
```

Лістинг А.1 – Програмний код, аркуш 8

```
      if (autoMode) {
        relayState = false;
        digitalWrite(RELAY_PIN, LOW);
      }
    } else if (currentStatus == STATUS_ALARM) {
      currentStatus = (WiFi.status() == WL_CONNECTED) ?
STATUS_OK : STATUS_WIFI_ERROR;
    }

    if (autoMode && currentStatus != STATUS_ALARM) {
      if (currentAmps > 1.0 && !relayState) {
        relayState = true;
        digitalWrite(RELAY_PIN, HIGH);
      } else if (currentAmps < 0.5 && relayState) {
        relayState = false;
        digitalWrite(RELAY_PIN, LOW);
      }
    }

    updateDisplay();
    updateRGBStatus();
  }

  checkWiFiStatus();
}
```

Лістинг А.1 – Програмний код, аркуш 9