НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих

комп'ютерних систем

До захисту допущено:

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування і спеціалізовані комп'ютерні систем»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Комп'ютерні засоби на базі мікроконтролера STM32 для моніторингу та управління енергоспоживанням побутових пристроїв»

Виконав:
студент IV курсу, групи КВ-13
Ольховський Максим Олександрович _____

Керівник:
доц.каф. СПіСКС, к.т.н.,с.н.с
Клятченко Ярослав Михайлович _____

Консультант з нормконтролю:
доц.каф. СПіСКС, к.т.н.,с.н.с
Клятченко Ярослав Михайлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)
Освітньо-професійна програма
«Системне програмування та спеціалізовані
комп'ютерні системи»
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
«___» _____2025 р.

ЗАВДАННЯ
на дипломний проєкт студента
Ольховського Максима Олександровича

1. Тема проєкту «Комп'ютерні засоби на базі мікроконтролера STM32 для моніторингу та управління енергоспоживанням побутових пристроїв», керівник проєкту доц.каф. СПіСКС, к.т.н., с.н.с. Клятченко Ярослав Михайлович, затверджені наказом по університету від «29» травня 2025 р. №2205-с
2. Термін подання студентом проєкту «5» червня 2025 р.
3. Вихідні дані до проєкту див. Технічне завдання
4. Зміст пояснювальної записки:
   - Аналіз існуючих рішень і теоретичних основ моніторингу та управління енергоспоживанням;
   - Апаратна платформа комп'ютерного засобу на базі STM32;
   - Розробка програмного забезпечення;
   - Тестування та опис роботи системи.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):
   - Структурна схема взаємодії сервер – емулятор – веб-клієнт;
   - Структурна схема основних модулів програмного забезпечення;
   - Блок-схема алгоритму обробки запиту на сервері;
   - Блок-схема алгоритму роботи емулятора STM32

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Доцент кафедри СПіСКС, к.т.н., Клятченко Ярослав Михайлович		

7. Дата видачі завдання 22 грудня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.04.2025	
2.	Розроблення та узгодження технічного завдання	18.04.2025	
3.	Аналіз існуючих рішень	30.04.2025	
4.	Підготовка матеріалів першого розділу проєкту	07.05.2025	
5.	Підготовка матеріалів другого розділу проєкту	14.05.2025	
6.	Розробка програмного забезпечення проєкту	26.05.2025	
7.	Підготовка матеріалів графічної частини проєкту	28.05.2025	
8.	Оформлення документації дипломного проєкту	02.06.2025	
9.	Попередній огляд матеріалів на кафедрі	05.06.2025	

Студент _____ Максим ОЛЬХОВСЬКИЙ
(підпис) (ініціали, прізвище)

Керівник проєкту _____ Ярослав Клятченко
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (67 с., 20 рис. 2 табл., 4 додатки).

Об’єктом розробки – створення комп’ютерної системи моніторингу та управління енергоспоживанням побутових пристроїв на базі мікроконтролера STM32, яка дозволяє здійснювати збір даних про енергоспоживання та надавати користувачеві інструменти для його оптимізації.

Комп’ютерна система передбачає: збір даних про енергоспоживання побутових пристроїв; передачу даних до центрального вузла на базі мікроконтролера STM32; візуалізацію даних моніторингу через веб-інтерфейс. В процесі розробки передбачається використання мікроконтролера STM32 для збору та обробки даних, а також технологій веб-розробки для створення користувацького інтерфейсу.

В ході розробки:

- проведено аналіз вже існуючих систем моніторингу та управління енергоспоживанням;
- сформульовані вимоги до комп’ютерної системи моніторингу та управління енергоспоживанням побутових пристроїв на базі мікроконтролера STM32;
- розроблено емулятор мікроконтролера STM32;
- розроблено веб-інтерфейс для моніторингу енергоспоживання;
- проведено тестування розробленої системи на основі імітованих даних про енергоспоживання;

Упровадження цієї системи в побутових умовах дозволить підвищити енергоефективність використання електроенергії, зменшити витрати на комунальні послуги та сприяти більш свідомому споживанню ресурсів.

Ключові слова: КОМП’ЮТЕРНА СИСТЕМА МОНІТОРИНГУ ТА УПРАВЛІННЯ ЕНЕРГОСПОЖИВАННЯМ, STM32, МІКРОКОНТРОЛЕР, ВЕБ-ІНТЕРФЕЙС, МОНІТОРИНГ ЕНЕРГІЇ, УПРАВЛІННЯ ЕНЕРГІЄЮ.

ABSTRACT

The qualification work includes an explanatory note (67 p., 20 figures, 2 tables, 4 appendices).

The object of development – the creation of a computer system for monitoring and managing the energy consumption of household devices based on the STM32 microcontroller, which enables the collection of energy consumption data and provides the user with tools for its optimization.

The computer system provides: the collection of energy consumption data from household devices; data transmission to a central unit based on the STM32 microcontroller; and visualization of monitoring data through a web interface. The development process involves the use of the STM32 microcontroller for data acquisition and processing, as well as web development technologies to create a user interface.

During the development process:

- an analysis of existing energy consumption monitoring and management systems was carried out;
- requirements for the computer system for monitoring and managing energy consumption of household devices based on STM32 were formulated;
- emulator of the STM32 microcontroller was developed;
- a web interface for energy consumption monitoring was created;
- the developed system was tested using simulated energy consumption data.

The implementation of this system in household environments will improve energy efficiency, reduce utility costs, and promote more conscious resource consumption.

Keywords: COMPUTER SYSTEM FOR MONITORING AND ENERGY MANAGEMENT, STM32, MICROCONTROLLER, WEB INTERFACE, ENERGY MONITORING, ENERGY MANAGEMENT.

ЗМІСТ

					ІАЛЦ.467200.002 ТЗ		
Змін	Арк.	№ докум.	Підпис	Дата			
Розробив		Ольховський М.О.			«Комп'ютерні засоби на базі мікроконтролера STM32 для моніторингу та управління енергоспоживанням побутових пристроїв »	Літ.	Аркуш 1 / Аркушів 4
Перевірив		Клятченко Я.М.					
Н. контроль		Клятченко Я.М.			***Технічне завдання***	КПІ ім. Ігоря Сікорського, ФПМ КВ-13	
Затвердив		Романкевич В.О.					

1.НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Комп’ютерні засоби на базі мікроконтролера STM32 для моніторингу та управління енергоспоживанням побутових пристроїв».

Галузь застосування: контроль енергоспоживання побутових пристроїв у домашніх умовах з метою оптимізації енергоефективності.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення засобів на базі STM32 для моніторингу та управління енергоспоживанням побутових пристроїв, що дозволить користувачам контролювати та оптимізувати використання електроенергії в домашніх умовах.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, довідники, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- зручний для використання веб-інтерфейс для моніторингу та управління;
- сумісність з сучасними веб-браузерами на різних операційних системах;
- базова обробка отриманих даних;

- зберігання даних про енергоспоживання для подальшого відображення у вигляді історії;
- наявність зручної системи сповіщень ініціатора та виконавця;
- відображення стану підключених побутових пристроїв (увімкнено/вимкнено);

5.2 Вимоги до апаратного забезпечення

- комп'ютер для розробки та запуску;
- наявність доступу до мережі Internet (EDGE, 3G, 4G);

5.3 Вимоги до програмного та апаратного забезпечення користувача

- Сучасний веб-браузер (Google Chrome, Mozilla Firefox, Safari, Edge тощо) для доступу до веб-інтерфейсу на будь-якій операційній системі, що підтримується браузером;
- Мати встановлений Node.js та Python

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту
1.	Вивчення літератури за тематикою проєкту	15.04.2024
2.	Розроблення та узгодження технічного завдання	18.04.2024
3.	Аналіз існуючих рішень	30.04.2024
4.	Підготовка матеріалів першого розділу дипломного проєкту	07.05.2024
5.	Підготовка матеріалів другого розділу дипломного проєкту	14.05.2024
6.	Підготовка матеріалів третього розділу дипломного проєкту	26.05.2024
7.	Підготовка матеріалів четвертого розділу дипломного проєкту	28.05.2024
8.	Підготовка графічної частини дипломного проєкту	31.05.2024
9.	Оформлення документації дипломного проєкту	02.06.2024
10.	Попередній огляд матеріалів диплому на кафедрі	03.06.2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

MCU — Microcontroller Unit (мікроконтролер)

STM32 — серія 32-бітних мікроконтролерів на базі ARM Cortex від STMicroelectronics

ESP8266 / ESP32 — мікроконтролери з підтримкою Wi-Fi / Bluetooth, часто використовуються для IoT

UART — Universal Asynchronous Receiver-Transmitter (універсальний асинхронний приймач-передавач)

I2C — Inter-Integrated Circuit (послідовна дволінійна шина для з'єднання мікросхем)

SPI — Serial Peripheral Interface (послідовний периферійний інтерфейс)

ADC — Analog-to-Digital Converter (аналогово-цифровий перетворювач)

API — Application Programming Interface (програмний інтерфейс прикладного рівня)

REST API — архітектурний стиль обміну даними між клієнтом і сервером через HTTP

HTML — HyperText Markup Language (мова розмітки гіпертексту)

CSS — Cascading Style Sheets (каскадні таблиці стилів)

JS / JavaScript — скриптова мова для створення інтерактивних веб-сторінок

Node.js — середовище виконання JavaScript на стороні сервера

Express.js — фреймворк для побудови серверних додатків на Node.js

Chart.js — бібліотека для побудови графіків на JavaScript

JSON — JavaScript Object Notation (формат обміну даними)

PWM — Pulse Width Modulation (широтно-імпульсна модуляція)

RAM — Random Access Memory (оперативна пам’ять)

ВСТУП

У сучасному світі все більшої актуальності набуває питання раціонального використання енергетичних ресурсів. Водночас зростання цін на енергоносії та усвідомлення необхідності зменшення негативного впливу на навколишнє середовище спонукають до пошуку та впровадження інноваційних способів моніторингу та управління споживанням енергії, зокрема в житловому секторі. Значна частина енергії в оселях витрачається неефективно через відсутність належного контролю та своєчасного втручання в роботу приладів.

Досягнення мікроелектроніки та вбудованих систем відкрили широкі можливості для створення інтелектуальних систем, які можуть точно контролювати споживання енергії та надавати користувачам інструменти оптимізації. Серед різноманітних доступних платформ мікроконтролерів серія STM32 від STMicroelectronics займає особливе місце. Завдяки високій продуктивності, низькому енергоспоживанню, широкому спектру периферійних пристроїв і розвиненій екосистемі мікроконтролери STM32 є перспективною основою для розробки систем моніторингу та управління енергією. Важливість розробки комп’ютерних інструментів для контролю споживання енергії в домашніх умовах полягає у зростаючій потребі в простих, економічних та ефективних рішеннях. Впровадження таких систем дозволить користувачам отримувати детальну інформацію про енергоспоживання їхніх пристроїв, виявляти неефективні моделі роботи та дистанційно керувати ними, щоб зменшити витрати та сприяти більш свідомому використанню енергії.

Дана робота присвячена дослідженню можливості створення такої системи на базі мікроконтролерів STM32 з використанням веб-інтерфейсу взаємодії з користувачем.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ І ТЕОРЕТИЧНИХ ОСНОВ МОНІТОРИНГУ ТА УПРАВЛІННЯ ЕНЕРГОСПОЖИВАННЯМ

1.1 Класифікація за електроспоживанням побутових електропристроїв

Наші оселі сьогодні важко уявити без електроприладів. Вони роблять наше життя більш ефективним, безпечним та зручним. Саме побутова техніка споживає більшу частину електроенергії в житлових будинках. Щоб користуватися енергією з розумом, корисно розібратися, які прилади для чого призначені, наскільки часто їх використовують і скільки електрики вони споживають.

1.1.1 Основні групи побутових електропристроїв

Побутові пристрої можна поділити на кілька груп за функціональністю: пристрої приготування їжі, системи кліматичного контролю, аудіо- та відеообладнання, інформаційно-комунікаційне обладнання, пристрої для догляду за речами, освітлювальні прилади, сантехнічні системи з електроприводом та інші малопотужні пристрої.

Нижче наведена таблиця 1.1 з характеристиками та прикладами пристроїв:

Функціональна група	Приклади пристроїв	Орієнтовна потужність (Вт)	Характер використання
Пристрої приготування їжі	Електроплити, духовки, мікрохвильові печі.	650 - 3100	Короткочасне, інтенсивне
Системи кліматичного контролю	Кондиціонери, обігрівачі, тепло-вентилятори.	1100 - 5200	Сезонне, тривале
Пристрої для догляду за речами	Пральні машини, праски, пилососи.	600 - 3000	Періодичне, середньої тривалості
Аудіо- та відео-обладнання	Телевізори, домашні кінотеатри, медіаплеєри.	100 - 500	Регулярне, режим очікування
Освітлювальні прилади	Лампи розжарювання, настільні лампи.	10 - 100+	Тривале, залежить від часу доби
Інформаційно-комунікаційне обладнання	Персональні комп’ютери, ноутбуки	30 - 300	Щоденне, тривале
Сантехнічні системи з електроприводом	Електричні бойлери, системи «розумного дому».	1500 - 3500+	Тривале, залежить від потреби
Малопотужні пристрої	Фени, електричні кухонні ваги, годинники.	10 - 200	Короткочасне, постійне

Таблиця 1.1– Приклади різних пристроїв та їх характеристики

1.1.2 Рівні споживання електроенергії: високе, середнє, низьке

Для ефективного управління енергією важливо розуміти не лише призначення приладів, а й те, скільки вони споживають. Це допомагає оптимально розподіляти навантаження на електромережу, економити та визначати, за чим варто пильніше стежити. Залежно від потужності та часу роботи, прилади поділяють на три групи:

1. Низьке споживання. Це пристрої, що беруть до 120-350 Вт або працюють дуже мало часу. Сюди відносяться світлодіодні лампи, зарядні пристрої для смартфонів, настільні лампи, роутери, кухонні ваги, електричні зубні щітки, годинники. Хоча вони малопотужні, створюється фонове навантаження завдяки їхній постійній роботі. Загалом їхній внесок невеликий, але, якщо таких приладів багато або вони використовуються неефективно, їхня роль може зрости.
2. Середнє споживання. Сюди відносяться пристрої потужністю 320-1600 Вт, які використовуються періодично, але регулярно. Тут можна згадати пральні машини, електрочайники, холодильники (середнє споживання за годину), персональні комп'ютери, телевізори, мікрохвильові печі. Згадані прилади є робочими одиницями вдома і становлять більшу частину денного енергоспоживання. Їхня оптимізація – важливий та ефективний крок до економії.
3. До класу високого споживання електроенергії входять пристрої з потужністю понад 1450 Вт, які при роботі спричиняють пікові навантаження. Головними представниками такої категорії є кондиціонери, електричні бойлери, сушарки для білизни, обігрівачі та пилососи високої потужності. Використання такі прилади, слід стежити за часом їхньої роботи, а ще краще – порібно автоматизувати керування ними. Вони часто працюють зранку або під вечір, що створює нерівномірне навантаження на мережу.

1.1.3 Часові профілі використання електропристроїв

Для того, щоб зрозуміти, скільки енергії споживається, важливо враховувати не тільки потужність приладу, а й те, коли і як довго ним користуються. Кожен пристрій має свій унікальний графік роботи – це комбінація часу доби, тривалості роботи, частоти включень та періодичності. Такі графіки часто показують типове навантаження на домашню електромережу і є важливими для створення систем моніторингу та управління енергією.

Деякі прилади, такі як холодильники, працюють постійно. Можна згадати Wi-Fi роутери, системи відеоспостереження чи сигналізації, які функціонують безперервно. Вони й створюють базове і постійне навантаження.

Інші пристрої мають чітко визначений розклад – вони активні вдень або ввечері. Одразу спадають на думку мікрохвильові печі, чайники та кавоварні машини, які найчастіше використовуються вранці (на сніданок) або ввечері (на вечерю). Освітлення вмикається в сутінках або вночі, особливо взимку і восени. Телевізори, ноутбуки та радіо переважно працюють ввечері, коли люди повертаються додому та відпочивають.

Особливої уваги заслуговують прилади, що працюють сезонно або епізодично. Обігрівачі активно використовуються в холодну пору року, а кондиціонери, навпаки – влітку. Бойлери можуть працювати постійно або інтенсивніше ввечері, коли з’являється більша потреба у гарячій воді. Деякі прилади, такі як пральні машини або посудомийки, вмикаються час від часу, що ускладнює можливість прогнозувати їхній вплив на енергоспоживання та систему вцілому, але дає можливість ефективно розподілити навантаження за допомогою автоматизованих систем.

Знання таких графіків є важливим для створення "розумного" будинку. Воно дозволяє з розумом керувати приладами: запускати енергоємні пристрої пізніше, рівномірно розподіляти навантаження, уникати пікових стрибків і підвищувати енергоефективність системи.

1.1.4 Вплив енергоспоживання на загальне навантаження в мережі

Споживання електрики в будинках безпосередньо впливає на те, наскільки стабільно та ефективно працює вся енергомережа. Кожен електроприлад створює певне навантаження, і всі ці навантаження разом формують загальний профіль споживання у конкретному будинку, районі або навіть в масштабах всієї країни.

Найбільш проблемними є пікові періоди, коли багато людей одночасно вмикають потужні прилади – зазвичай, це відбувається вранці або ввечері. Така активність призводить до різких стрибків навантаження, які вимагають від енергосистеми додаткових потужностей, щоб підтримувати нормальну напругу. У невеликих містах або селищах, де один трансформатор обслуговує багато будинків, нерівномірне навантаження може призвести до перевантаження ліній електропередач, втрат енергії та навіть аварій. Для прикладу, масове використання електрообігрівачів узимку може сильно перевищити допустимі навантаження. Ситуація ускладнюється тим, що більшість наших приладів не мають вбудованих систем, які б реагували на перевантаження мережі або зміни тарифів. Це означає, що керувати споживанням потрібно власноруч або за допомогою зовнішніх розумних систем.

Інша вагома проблема – це якість електрики. Коли працює багато потужних приладів, особливо тих, що вмикаються і вимикаються імпульсно, у мережі можуть виникати коливання напруги, спотворення сигналу та зниження коефіцієнта потужності. Це може негативно впливати на чутливу електроніку, збільшувати втрати енергії та напряму впливати на термін експлуатації побутової техніки. Особливо це стосується старих будинків, де проводка застаріла та не розрахована на одночасну роботу кількох потужних пристроїв.

З іншого боку, якщо впровадити інтелектуальний моніторинг та управління енергоспоживанням, це дозволить не лише зменшити споживання в окремому будинку, а й згладити пікові навантаження в масштабах мікрорайону

або міста. Мікроконтролери, такі як STM32, дають можливість створювати розумні системи, які автоматично регулюють графіки роботи приладів, оптимізують їх включення відповідно до тарифів або рівня навантаження, а також інформують користувача про можливі небезпеки.

Розумне управління споживанням електроенергії побутовими приладами – це правильний шлях до стабільності всієї електричної інфраструктури. Воно зменшує ризики перевантаження, рахунки за електрику стають дешевшими і прокладається шлях до енергоефективного майбутнього.

1.2 Проблеми енергоспоживання та методи його оптимізації

Коли люди використовують забагато електроенергії в оселях, це створює проблеми не тільки для гаманця, а й для природи. Щоб зрозуміти, чому це так, слід розглянути наслідки та способи оптимізації.

1.2.1 Наслідки надмірного енергоспоживання (економічні, екологічні)

Коли використовують забагато електрики, це має кілька неприємних наслідків:

1. В першу чергу для природи. Більшість електрики в Україні досі виробляється на теплових електростанціях, які спалюють вугілля, газ або нафту. При цьому в повітря викидається багато шкідливих речовин, зокрема вуглекислий газ (CO_2), що спричиняє глобальне потепління та зміни клімату. Крім того, виробництво енергії забруднює повітря, воду та ґрунт, що шкодить нашому здоров'ю та виснажує природні ресурси.
2. Стає досить відчутно та боляче для кишені. Чим більше енергії споживаємо, тим вищі рахунки за світло і, виходячи із цього, більше грошей слід платити щомісяця. Також це особливо помітно, коли діють різні тарифи, в залежності від часу доби або є ліміти на споживання.

3. Для всієї країни. Коли користувачі споживають забагато, енергетична система починає задихатися і можуть відбуватися блекаути, які були доволі популярним явищем ще рік тому. Як наслідок, треба більше коштів вкладати у будівництво нових електростанцій, ремонт старих мереж. Держава, у свою чергу, вимушена підвищувати тарифи для всіх або витрачати бюджетні кошти на підтримку енергетики.

Окрім цього, високі витрати на електрику б'ють по бюджету малозабезпечених сімей, а пікові навантаження збільшують ризик аварій на електромережах, що призводить до відключень світла. Тому, надмірне споживання енергії – це глобальна проблема, яка вимагає професійного вирішення.

1.2.2 Принципи енергоефективного використання побутової техніки

Енергоефективність у побуті – це мистецтво та здатність отримувати максимум користі від приладів, витративши при цьому мінімум електрики. Це не тільки зберігає кошти платників, а й зменшує навантаження на мережу та має позитивний вплив на довкілля.

Першим принципом є вибір розумної побутової техніки. Коли обираєш нову посудомийку чи морозильну камеру, слід звертати увагу на клас енергоспоживання (вони існують від A до G). Прилади з маркуванням A , особливо A+, A++, A+++, споживають помітно менше електроенергії, виконуючи по суті ті ж самі функції.

Другим не менш значущим аспектом є раціональне використання техніки. Пральну машину чи посудомийку треба включати лише при повному її завантаженні. У чайнику рекомендовано кип'яти води рівно стільки, скільки потрібно для вживання. Ще однією порадою є необхідність відключати прилади з розетки, коли ними не користуєшся. Багато з них продовжують їсти електрику

навіть у вимкненому стані (це називається "приховане" або "фонове" споживання).

Третій принцип — використання приладів у відповідний час доби, тобто, коли світло коштує дешевше. У таких системах вартість кіловат-години вночі значно нижча, ніж удень або ввечері, тому доцільно переносити роботу енергоємного обладнання (бойлера, посудомийки) на нічний час. Це не тільки зекономить кошти, а й допоможе розвантажити енергосистему в години пік.
Четвертий принцип — треба обслуговувати та дбати про техніку. Брудні фільтри, пошкоджені гумки (ущільнювачі) або старі деталі можуть змушувати прилад працювати менш ефективно і споживати більше енергії. Регулярне обслуговування допоможе техніці працювати гідно.

Окремо варто зазначити розумний дім. Інтеграція приладів у систему розумного дому дозволяє автоматизувати їхню роботу, бачити, скільки енергії вони споживають у реальному часі, та оптимізувати загальне навантаження..

1.2.3 Поняття "розумного дому" у контексті енергозбереження

Розумний дім (smart home) – це не тільки зручно, а й дуже важливо для економії енергії. Тут представлена ціла система, яка за допомогою спеціальних пристроїв (мікроконтролерів, датчиків, програм) може автоматично або дистанційно керувати всіма побутовими процесами, в тому числі й пристроями. Головна мета – зробити споживання енергії максимально ефективним.

Основою розумного дому є сенсорна мережа, яка забезпечує моніторинг стану приміщень та пристроїв у режимі реального часу. Датчики температури, руху, вологості чи світла постійно стежать за станом приміщень. Для прикладу, світло автоматично вимкнеться, коли людина вийде з кімнати, або опалення налаштується залежно від погоди та присутності людей у квартирі чи будинку. Це дозволяє економити енергію, не втрачаючи комфорту, що є важливим.

Іншою важливою складовою є інтелектуальні контролери, які здатні визначати, коли навантаження на мережу найбільше, і відкладати запуск енергоємних приладів (зарядка електромобіля, бойлер), на той час, коли тарифи нижчі або мережа менш завантажена. Це не тільки економить витрати користувачів, а й допомагає збалансувати загальне навантаження.

Крім керування, розумний дім забезпечує детальний облік та аналіз енергоспоживання. Розумний дім дозволяє в будь-який момент побачити, скільки електрики споживає кожен прилад, оцінити його ефективність і навіть знайти прихованих споживачів, які непомітно використовують енергію. Це створює передумови для більш свідомого користування електроприладами.

Сучасні системи можуть інтегруватися з сонячними панелями або іншими відновлюваними джерелами енергії, автоматично підлаштовуючи споживання під власне виробництво. Таким чином, розумний дім може не тільки зменшувати споживання, а й робити будинок частково або повністю енергонезалежним.

Створювати такі системи стає все простіше завдяки мікроконтролерам, таким як STM32. Вони є гнучкою основою для індивідуальних систем автоматизації, які можна налаштувати, підключити до інтернету та ними можна керувати через мобільні додатки.

Приклад розумного будинку зображено на рисунку 1.1.

Рисунок 1.1 – Система розумного будинку

1.2.4 Алгоритмічні методи оптимізації (графіки вмикання, пріоритети навантаження)

Алгоритмічні методи оптимізації споживання енергії є ключовим інструментом у сучасних автоматизованих системах керування побутовою електротехнікою. Їх застосування дозволяє знизити загальне споживання електроенергії та розподілити навантаження у часі, запобігаючи піковим перевантаженням і економлячи кошти завдяки використанню дешевших тарифів у позапікові години. До основних таких методів належать побудова графіків вмикання пристроїв, визначення пріоритетів навантаження та застосування адаптивних алгоритмів зворотного зв'язку.

Графіки вмикання — це алгоритми, що регламентують час початку та тривалість роботи побутових пристроїв. Вони диктують, коли пристроям починати працювати і як довго. Якщо є нічний тариф, алгоритм може автоматично вмикати бойлер чи пральну машину вночі, коли електрика дешевша. Графіки можуть бути статичними, тобто, користувач сам їх налаштовує, і динамічними, значить, система сама їх створює, враховуючи навантаження, тарифи або навіть звички людини. Наприклад, вона може навчитися вмикати посудомийну машину о 4:00 ночі, бо тоді найменше навантаження.

Іншим важливим підходом є визначення пріоритетів навантаження. Прилади діляться на критичні, тобто, які мають працювати завжди, наприклад, холодильник або освітлення, та некритичні, роботу яких можна відкласти, наприклад, електросушка або зарядний пристрій. Алгоритм стежить за навантаженням: якщо воно стає занадто високим, він вимикає некритичні прилади, залишаючи працювати тільки найважливіші. Це допомагає уникнути аварій і продовжити термін служби техніки.

Сучасні системи також використовують адаптивні алгоритми керування з елементами машинного навчання. Ці системи можуть навчатися на основі

минулого споживання, враховувати зовнішні фактори (погоду, зміни тарифів) і навіть прогнозувати майбутнє навантаження. Для прикладу, влітку система може передбачити, що кондиціонери скоро підвищать навантаження, і заздалегідь оптимізувати графіки інших приладів.

Крім того, такі алгоритми можуть спілкуватися з розумною енергомережею (smart grid), яка передає дані про загальне навантаження на електростанції. Це дозволяє розумному дому синхронізуватися з потребами всієї енергосистеми, роблячи споживання не тільки ефективним, а й гнучким у масштабах місцевості.

1.3 Існуючі системи моніторингу енергоспоживання

На сучасному ринку представлено багато готових рішень для відстеження споживання електроенергії вдома. Ці пристрої зручні у використанні, мають широкий функціонал і легко інтегруються в системи розумного дому. Серед найпопулярніших – TP-Link HS110 та Shelly EM, які допомагають детально контролювати витрати електрики та керувати приладами віддалено.

1.3.1 Короткий огляд комерційних пристроїв

TP-Link HS110 (рисунок 1.2) – це розумна Wi-Fi розетка. Вона дозволяє вмикати та вимикати підключені прилади дистанційно через мобільний додаток Kasa Smart. Головна перевага HS110 – вбудований модуль для вимірювання енергоспоживання. Користувачі можуть у реальному часі бачити, скільки електрики витрачає конкретний пристрій, переглядати статистику за день, тиждень або місяць, а також прогнозувати витрати. Розетка підтримує таймери та розклади, що дає змогу налаштовувати автоматичне керування – наприклад, вимикати обігрівач вночі або вмикати світло у певний час. TP-Link HS110 також

працює з голосовими помічниками Amazon Alexa та Google Assistant, що спрощує її інтеграцію в екосистему розумного дому.

Рисунок 1.2 – Wi-Fi розетка TP-Link HS110

Shelly EM (рисунок 1.3) – це більш просунуте рішення, призначене для моніторингу споживання електроенергії всього будинку або окремих груп приладів. Пристрій може вимірювати струм і напругу на двох незалежних каналах, дозволяючи оцінювати як активне, так і реактивне навантаження. Завдяки вбудованому Wi-Fi-модулю Shelly EM передає дані в хмарне сховище або на локальний сервер за допомогою протоколів MQTT, REST API або HTTP. Це робить його придатним для складних автоматизованих систем з індивідуальними алгоритмами керування. У поєднанні зі спеціальним програмним забезпеченням або інтеграцією в платформи, як-от Home Assistant, OpenHAB чи Domoticz, Shelly EM пропонує гнучкий інтерфейс для контролю та аналізу енергоспоживання.

Рисунок 1.3 – Пристрій Shelly EM

Варто зазначити, що ці пристрої значно відрізняються за призначенням. TP-Link HS110 найкраще підходить для точкового контролю окремих приладів (наприклад, телевізора або чайника), тоді як Shelly EM здатний відстежувати енергоспоживання більш потужних систем, таких як електричні панелі, кондиціонери або водонагрівачі.

1.3.2 Огляд відкритих рішень та DIY-систем

Крім комерційних продуктів, широке поширення отримали відкриті (open-source) та DIY-рішення (Do It Yourself). Вони дозволяють дослідникам та розробникам створювати гнучкі, масштабовані та налаштовані під індивідуальні потреби системи моніторингу енергоспоживання. Такі проєкти часто базуються на доступних апаратних платформах (наприклад, Arduino, ESP32, STM32) і дозволяють глибоко налаштовувати програмне забезпечення. Найвідоміші приклади – OpenEnergyMonitor та ESPHome, які забезпечують повну прозорість роботи, можливість розширення та мають активні спільноти підтримки.

OpenEnergyMonitor – це відкритий проєкт, створений для розробки повноцінної системи моніторингу енергії в реальному часі. Основою платформи є апаратний модуль EmonTx (працює на контролері ATmega328), який дозволяє підключати трансформатори струму (CT) та сенсори напруги для вимірювання енергоспоживання. Дані передаються на базову станцію EmonBase, яку можна реалізувати на базі Raspberry Pi із встановленим сервером EmonCMS. Цей сервер є потужною веб-платформою для візуалізації, аналізу та збереження історії енергоспоживання. Перевагою OpenEnergyMonitor є його модульність: можна підключати додаткові датчики температури, вологості, освітлення, що перетворює систему на багатофункціональну екосистему для енергоефективного дому.

ESPHome – це прошивка з відкритим кодом для мікроконтролерів ESP8266/ESP32, яка дозволяє програмувати логіку пристроїв через просту конфігурацію у форматі YAML. У контексті моніторингу енергоспоживання, ESPHome часто використовується разом з датчиками струму (наприклад, SCT-013) та модулями типу HLW8012, PZEM-004T або ADE7953, що забезпечують точне вимірювання струму, напруги, потужності та енергії. Однією з найважливіших переваг ESPHome є тісна інтеграція з Home Assistant – однією з найпопулярніших платформ для побудови систем розумного дому. Завдяки цьому всі вимірювані параметри легко відображаються у зручному інтерфейсі, підтримується автоматизація, сповіщення та звіти.

DIY-рішення мають кілька значних переваг: по-перше, вони значно дешевші за комерційні аналоги, зберігаючи при цьому функціональність; по-друге, користувач отримує повний контроль над системою – від програмної логіки до фізичного підключення. Водночас, вони вимагають базових знань з електроніки, мікроконтролерів та програмування.

1.3.3 Функціональні можливості: віддалений доступ, звітність, автоматизація

Сучасні системи моніторингу енергоспоживання, як комерційні, так і відкриті, пропонують користувачам широкий спектр функціональних можливостей. Ці можливості значно підвищують ефективність контролю за споживанням електроенергії в побуті. До ключових серед них належать: віддалений доступ, автоматичне формування звітності та можливість інтегрованої автоматизації.

Віддалений доступ є однією з найважливіших функцій. Вона дозволяє користувачеві в будь-який момент переглядати поточне споживання електроенергії або керувати окремими пристроями через мобільний додаток чи веб-інтерфейс. Це особливо корисно для тих, хто перебуває поза оселею, але бажає стежити за станом техніки або оперативно реагувати на нештатні ситуації (наприклад, забутий увімкнений обігрівач). Більшість пристроїв (TP-Link HS110, Shelly EM, ESPHome) підтримують хмарні сервіси або локальний доступ через Wi-Fi і можуть взаємодіяти з іншими компонентами розумного дому за допомогою MQTT, HTTP чи спеціалізованих API.

Звітність у таких системах реалізується через автоматичне збереження та обробку історичних даних споживання електроенергії. Це дає змогу створювати графіки, таблиці та аналітичні звіти, які відображають щоденні, щотижневі або місячні тенденції використання електроенергії. Деякі системи, як-от OpenEnergyMonitor або Home Assistant, дозволяють також експортувати ці дані у формати, наприклад, CSV, або інтегрувати їх у сторонні аналітичні сервіси. Звітність є важливим інструментом для виявлення енергоємних приладів, порівняння ефективності використання енергії в різні періоди та планування подальшої оптимізації.

Автоматизація — це третій і найбільш потужний рівень функціональності. Системи моніторингу енергоспоживання можуть не тільки збирати дані, але й реагувати на них у реальному часі. Для прикладу можна

навести ситуацію, коли, при перевищенні встановленого порогу споживання система може автоматично вимикати некритичні пристрої, надсилати повідомлення користувачеві або змінювати сценарії роботи. Автоматизація також дає змогу реалізувати сценарії на основі часу доби, рівня освітленості, температури або присутності людей у приміщенні. У системах на базі Home Assistant або Node-RED користувачі можуть створювати складні логічні правила без програмування — лише за допомогою графічного інтерфейсу.

1.3.4 Порівняльний аналіз рішень

Вибір системи моніторингу енергоспоживання залежить від цілей, технічних можливостей приміщення, бюджету, а також, вимог до точності вимірювань та функціональності. Доцільно порівняти найбільш популярні комерційні та відкриті рішення – такі як TP-Link HS110, Shelly EM, OpenEnergyMonitor та ESPHome – за трьома ключовими критеріями: вартість, точність вимірювань та зручність інтеграції в побутові або автоматизовані системи.

1. Вартість є одним із найважливіших факторів для кінцевого споживача. Комерційні пристрої, як-от TP-Link HS110 або Shelly EM, зазвичай мають фіксовану вартість, яка включає як апаратну частину, так і готове програмне забезпечення з підтримкою. Наприклад, TP-Link HS110 коштує приблизно 700–1000 грн, тоді як Shelly EM – від 1500 до 2500 грн залежно від комплектації (наявність трансформаторів струму). Водночас DIY-рішення на основі ESPHome або OpenEnergyMonitor можуть бути дешевшими, особливо при замовленні компонентів окремо. Наприклад, модуль ESP32 з PZEM-004T та датчиком струму може коштувати в межах 300–500 грн, але вимагатиме часу на збірку, налаштування та прошивання.
2. Щодо точності вимірювань, Shelly EM та OpenEnergyMonitor демонструють високий рівень достовірності завдяки використанню якісних датчиків

та каліброваних вимірювальних схем. TP-Link HS110 забезпечує базовий рівень точності (похибка може становити 5–10%), що достатньо для побутового контролю, але не завжди підходить для технічного аналізу. Точність DIY-рішень на ESPHome значно залежить від обраного датчика – наприклад, HLW8012 має середню точність, тоді як ADE7953 – високу, але складнішу реалізацію.

3. Зручність інтеграції також суттєво відрізняється. TP-Link HS110 підключається за кілька хвилин через мобільний додаток, але його можливості обмежені екосистемою Kasa. Shelly EM підтримує відкриті протоколи (MQTT, HTTP API), що дозволяє легко інтегрувати його у Home Assistant, Node-RED або інші системи автоматизації. DIY-рішення (ESPHome, OpenEnergyMonitor) мають найвищу гнучкість, однак вимагають більше технічних знань, особливо на етапі початкового налаштування. Проте, після інтеграції вони можуть забезпечити потужний функціонал, який часто перевершує комерційні пристрої.

Результати аналізу наведені в таблиці 1.2:

Пристрій / Система	Орієнтовна вартість	Точність вимірювання	Інтеграція
TP-Link HS110	700–1000 грн	Низька / Середня	Висока (обмежено Kasa)
Shelly EM	1500–2500 грн	Висока	Висока (MQTT, API)
OpenEnergyMonitor	1200–3000 грн	Висока	Висока (EmonCMS, локальний сервер)
ESPHome (DIY)	300–700 грн	Залежить від сенсора	Висока (гнучка, Home Assistant)

Таблиця 1.2 – Порівняльний аналіз систем моніторингу енергоспоживання

1.4 Роль мікроконтролерів у системах керування енергоспоживанням

Мікроконтролери відіграють надважливу роль у створенні сучасних систем для відстеження та управління енергоспоживанням. Вони забезпечують потрібну потужність для обчислень, гнучкість у налаштуванні та можливість підключення до різних зовнішніх пристроїв. Завдяки своїм невеликим розмірам, економному споживанню енергії та великій кількості вбудованих функцій, мікроконтролери стали основою як для побутових "розумних" пристроїв, так і для промислових систем обліку енергії.

1.4.1 Загальна характеристика мікроконтролерів для моніторингу

Основне завдання мікроконтролера в системі моніторингу – це зчитування даних із датчиків струму та напруги, обробка цих сигналів, а потім передача результатів (наприклад, активної потужності, спожитої енергії, коефіцієнта потужності) до інших пристроїв або хмарних сервісів. Для цього мікроконтролер повинен мати спеціальні складові: аналогово-цифрові перетворювачі (ADC), модулі послідовного зв'язку (UART, I2C, SPI), таймери та інтерфейси бездротового зв'язку (Wi-Fi, Bluetooth, Zigbee).

Серед найпопулярніших мікроконтролерів, які використовуються для енергомоніторингу, можна виділити ESP8266 та ESP32 від компанії Espressif, STM32 від STMicroelectronics, а також Atmega328P, що використовується в платах Arduino. Серія ESP має вбудований Wi-Fi, що дозволяє передавати дані без дротів без додаткових модулів. STM32, своєю чергою, пропонує більшу обчислювальну потужність, точніші ADC та гнучкіший набір периферії. Це робить їх ідеальним вибором для складніших систем, які потребують високої точності або обробки сигналів у реальному часі.

У системах енергоспоживання мікроконтролери часто працюють разом з вимірювальними модулями, такими як PZEM-004T, ADE7753 або HLW8012, які

виконують попереднє вимірювання параметрів мережі. Мікроконтролер отримує ці дані і, відповідно до запрограмованої логіки, приймає рішення: відобразити інформацію на екрані, передати її по мережі, записати в пам'ять або запустити алгоритми автоматизації. Програмне забезпечення для мікроконтролера зазвичай пишеться мовами C/C++, MicroPython або за допомогою Arduino IDE.

Універсальність мікроконтролерів також дозволяє додавати функції користувацького інтерфейсу (кнопки, екрани), системи сповіщення (світлові або звукові сигнали), а також реалізовувати складні алгоритми прийняття рішень, що базуються на часових графіках, порогових значеннях або сценаріях поведінки. Крім того, за допомогою мікроконтролерів можна зберігати історичні дані в локальній пам'яті або на SD-картці, що робить систему незалежною від зовнішніх серверів.

1.4.2 Можливості STM32 у задачах енергоменеджменту

Мікроконтролери сімейства STM32, розроблені компанією STMicroelectronics, посідають провідне місце у сфері вбудованих систем, зокрема й у завданнях управління енергією. Завдяки високій продуктивності, енергоефективності, широкому спектру периферійних модулів та підтримці сучасних інтерфейсів, ці мікроконтролери дозволяють створювати як прості пристрої для моніторингу споживання електрики, так і складні системи автоматичного керування навантаженням.

Однією з ключових переваг STM32 є наявність високоточних АЦП (аналогово-цифрових перетворювачів) з роздільною здатністю до 16 біт. Це дозволяє точно зчитувати сигнали з датчиків струму та напруги, що є важливим для систем, де потрібно відстежувати навіть незначні зміни у споживанні енергії або аналізувати характер навантаження. Багатоканальні АЦП

дозволяють одночасно зчитувати декілька параметрів у реальному часі, що забезпечує точний енергетичний облік.

Мікроконтролери STM32 також мають апаратну підтримку математичних операцій, зокрема завдяки наявності FPU (Floating Point Unit) у моделях STM32F4, STM32F7, STM32H7. Це забезпечує швидку обробку сигналів, обчислення потужності, енергії, коефіцієнтів спотворення тощо. Це дає змогу реалізовувати локальну аналітику без потреби передавати великі обсяги необроблених даних на зовнішні сервери.

Завдяки наявності різних інтерфейсів зв’язку — UART, SPI, I2C, USB, Ethernet, CAN, а також підтримці LoRa, Wi-Fi або Bluetooth через додаткові модулі — STM32 легко інтегрується в IoT-екосистеми, розумні будинки або промислові SCADA-системи. Поєднання STM32 з Wi-Fi-модулем ESP8266 дозволяє організувати бездротову передачу даних до сервера моніторингу або мобільного застосунку, а використання Ethernet дає стабільне з'єднання в промислових умовах.

Ще однією сильною стороною STM32 є гнучкість у керуванні енергоспоживанням самого мікроконтролера. Багато моделей підтримують енергозберігаючі режими (Sleep, Stop, Standby), що дозволяє значно зменшити споживання електроенергії системою в цілому. Це особливо актуально для автономних пристроїв, що працюють на батареї або з обмеженим живленням.
Розвинене середовище розробки — STM32CubeIDE, STM32CubeMX, а також широка екосистема бібліотек (HAL, LL, CMSIS) полегшують створення програмного забезпечення навіть для складних задач. Додатково, підтримка вільних ОС реального часу (FreeRTOS) дозволяє будувати багатозадачні системи з точним розподілом пріоритетів, наприклад: окремі задачі для обробки даних, передачі по мережі та управління навантаженнями.

1.4.3 Підключення датчиків струму та напруги до мікроконтролера

Для створення системи моніторингу енергоспоживання необхідно вимірювати два основні електричні параметри — струм і напругу. До мікроконтролера підключаються датчики, які перетворюють ці параметри в сигнали, придатні для обробки (зазвичай — аналогові напруги або цифрові імпульси). Підключення таких сенсорів до мікроконтролера, зокрема STM32, залежить від типу сенсора, рівня сигналу, а також вимог до точності, безпеки та ізоляції.

Датчики струму бувають декількох типів: шунтові резистори, трансформатори струму (наприклад, SCT-013), датчики на ефекті Холла (ACS712, ACS758) тощо. Шунтові резистори створюють падіння напруги пропорційно струму, яке можна виміряти за допомогою АЦП STM32. Такий підхід є дешевим і точним, однак вимагає належної гальванічної розв’язки для безпечної роботи з мережею 220 В. Трансформатори струму, як SCT-013, забезпечують безпечну ізоляцію і дають змінну напругу, яку також можна вимірювати, попередньо підключивши її через дільник напруги або випрямляч, якщо потрібно обробляти постійний рівень. Датчики на ефекті Холла генерують аналоговий сигнал, який прямо зчитується АЦП мікроконтролера.

Датчики напруги зазвичай реалізуються як дільники напруги, що знижують рівень мережевої напруги (наприклад, 230 В) до безпечного значення для входу АЦП (наприклад, до 3.3 В для STM32). Дільник складається з двох резисторів великого номіналу (для мінімізації струму), і зазвичай доповнюється захисними діодами або оптоізоляторами. Крім того, для точних вимірювань застосовують спеціалізовані модулі, такі як ZMPT101B — датчик змінної напруги з ізоляцією, що видає аналоговий сигнал, безпечний для АЦП.

Підключення до STM32 здійснюється через канали АЦП (наприклад, ADC1_IN0, ADC1_IN1 тощо), які мають високу роздільну здатність і швидкодію. Для забезпечення точних вимірювань слід також враховувати

використання фільтрів нижніх частот (RC-фільтри) для усунення шуму та паразитних коливань, а також правильне заземлення і розведення плати. У більш просунутих системах можна використовувати диференціальні входи АЦП або внутрішнє посилення, якщо це дозволяє конкретна модель STM32.

1.4.4 Комунікаційні інтерфейси (Wi-Fi, Bluetooth, RS-485) для передачі даних

У системах моніторингу та керування енергоспоживанням побутових пристроїв важливо не лише отримати дані з сенсорів, але й ефективно передати ці дані для подальшої обробки, зберігання або візуалізації. Залежно від архітектури системи, умов її експлуатації та вимог до віддаленого доступу, застосовуються різні комунікаційні інтерфейси. Вони забезпечують зв’язок між мікроконтролером (наприклад, STM32) та зовнішніми пристроями чи серверами. Найпоширенішими серед них є Wi-Fi, Bluetooth, RS-485, Ethernet, а також сучасні бездротові протоколи IoT-класу.

Wi-Fi (IEEE 802.11) є популярним інтерфейсом для підключення побутових пристроїв до локальної мережі та Інтернету. Його головна перевага — висока швидкість передачі даних (до сотень Мбіт/с) і широке розповсюдження інфраструктури (домашні роутери). STM32 не має вбудованого Wi-Fi, однак легко інтегрується з Wi-Fi-модулями на базі ESP8266 або ESP32 через UART або SPI. Це дає змогу організувати передачу даних на сервери збору телеметрії (наприклад, MQTT-брокери, платформи Blynk, ThingsBoard) або забезпечити віддалене керування через мобільний застосунок.

Bluetooth, зокрема його енергоефективна версія BLE (Bluetooth Low Energy), підходить для короткодистанційного бездротового зв’язку — наприклад, між мікроконтролером і смартфоном. Для STM32 існує безліч готових модулів (HC-05, HM-10, MDBT42Q), які можна підключити через UART. Хоча швидкість Bluetooth нижча, ніж у Wi-Fi, його перевага — мала затримка, низьке енергоспоживання і відсутність потреби у зовнішній мережі.

RS-485 — це надійний інтерфейс для дротової передачі даних на великі відстані (до 1200 м), який часто застосовується у промисловості та системах "розумного дому" з високим рівнем перешкод. У зв’язці зі стандартом Modbus RTU RS-485 дозволяє побудувати просту та ефективну шину обміну даними між кількома пристроями. У STM32 передача здійснюється через UART з використанням зовнішнього трансивера (наприклад, MAX485), що перетворює логічні рівні UART у диференціальні сигнали RS-485.

Ethernet забезпечує швидке дротове з’єднання з локальною мережею і є популярним у стаціонарних рішеннях. Деякі моделі STM32 (наприклад, STM32F107, STM32H7) мають вбудований MAC-контролер Ethernet, який за допомогою зовнішнього трансівера PHY (наприклад, LAN8720) дозволяє реалізувати повноцінне TCP/IP-з’єднання. Це дає змогу обмінюватись даними з сервером, хмарною платформою або іншими пристроями локальної мережі.

Також варто згадати LoRa та Zigbee — інтерфейси, орієнтовані на IoT (інтернет речей) і розподілені системи з низькою швидкістю, але великою дальністю зв’язку або енергоефективністю. LoRa дозволяє передавати дані на десятки кілометрів, використовуючи вузькосмуговий протокол, і активно використовується у смарт-містах та облікових пристроях. Zigbee, натомість, створює mesh-мережу з великою кількістю вузлів і добре підходить для розумних будинків.

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

Питання ефективного моніторингу та управління енергоспоживанням побутових пристроїв набуває дедалі більшої актуальності у зв’язку зі зростанням навантаження на електромережі та потребою в ощадливому використанні ресурсів. Побутова техніка може суттєво відрізнятися за рівнем споживання електроенергії, тому її класифікація за потужністю — високе, середнє та низьке споживання — дозволяє більш раціонально планувати розподіл навантаження. Значну роль відіграє також час добового використання, адже навіть пристрої з невеликою потужністю, які працюють тривалий час, здатні спричиняти суттєве навантаження на мережу. Важливим фактором є усвідомлення негативних наслідків надмірного енергоспоживання — від зростання вартості комунальних послуг до забруднення довкілля через підвищені викиди парникових газів. Саме тому енергоефективність та раціональне використання електроприладів стають основою сталого підходу до енергокористування. Концепція «розумного дому» демонструє потенціал автоматизованих систем у зниженні витрат енергії завдяки сценаріям управління, віддаленому контролю та адаптації до реального споживання. Алгоритмічні методи, зокрема графіки вмикання пристроїв і призначення пріоритетів, забезпечують додаткову оптимізацію.

Аналіз ринку продемонстрував наявність великої кількості як готових комерційних рішень, так і відкритих проєктів з можливістю модифікації. Комерційні пристрої, як-от TP-Link HS110 або Shelly EM, пропонують просте підключення та зручні сервіси, тоді як DIY-рішення на базі OpenEnergyMonitor або ESPHome дають ширший контроль над системою. Функціональність таких систем охоплює збір даних, генерацію звітів, мобільне керування та інтеграцію з іншими системами розумного дому. Порівняння показує, що ключовими критеріями вибору є точність вимірювання, доступність, простота використання та підтримка розширень. Технічна реалізація подібних систем нерозривно пов’язана з мікроконтролерами, серед яких STM32 виділяються своєю

гнучкістю, наявністю потужних АЦП, підтримкою широкого спектра інтерфейсів і низьким енергоспоживанням. Для зчитування електричних параметрів широко використовуються сенсори струму та напруги, що підключаються до аналогових входів мікроконтролера. Передача зібраної інформації до центрального пристрою або хмарного сервісу реалізується за допомогою таких інтерфейсів, як UART, Bluetooth, Wi-Fi або RS-485.

Сумарно, ці компоненти та підходи формують цілісну концепцію побудови сучасного інтелектуального засобу керування енергоспоживанням на базі STM32, що поєднує апаратну надійність, точність вимірювань і можливості автоматизованого керування.

2. АПАРАТНА ПЛАТФОРМА КОМП'ЮТЕРНОГО ЗАСОБУ НА БАЗІ STM32

2.1 Обгрунтування вибору мікроконтролера STM32 та його архітектура

При виборі мікроконтролера для розробки енергоефективного пристрою моніторингу та управління енергоспоживанням побутових приладів, важливо знайти баланс між обчислювальною потужністю, енергоспоживанням, наявністю периферійних інтерфейсів та гнучкістю програмної підтримки. Серед найпопулярніших рішень на сучасному ринку для таких проєктів варто розглянути STM32, ESP32 та ATmega328. Кожен із них має свої сильні та слабкі сторони.

2.1.1 Порівняння STM32 з іншими популярними мікроконтролерами ESP32 та ATmega328

Мікроконтролери серії STM32, зокрема STM32F103C8T6 (рисунок 2.1), вирізняються завдяки своїй архітектурі ARM Cortex-M3, високій продуктивності на тактових частотах до 72 МГц, розширеному набору периферії (включно з ADC, DMA, SPI, I2C, UART, таймерами, PWM тощо) та підтримці енергозберігаючих режимів. Завдяки цьому STM32 може ефективно працювати з різними типами сенсорів струму та напруги, здійснюючи точні вимірювання у реальному часі та забезпечуючи низьке енергоспоживання при збереженні високої точності.

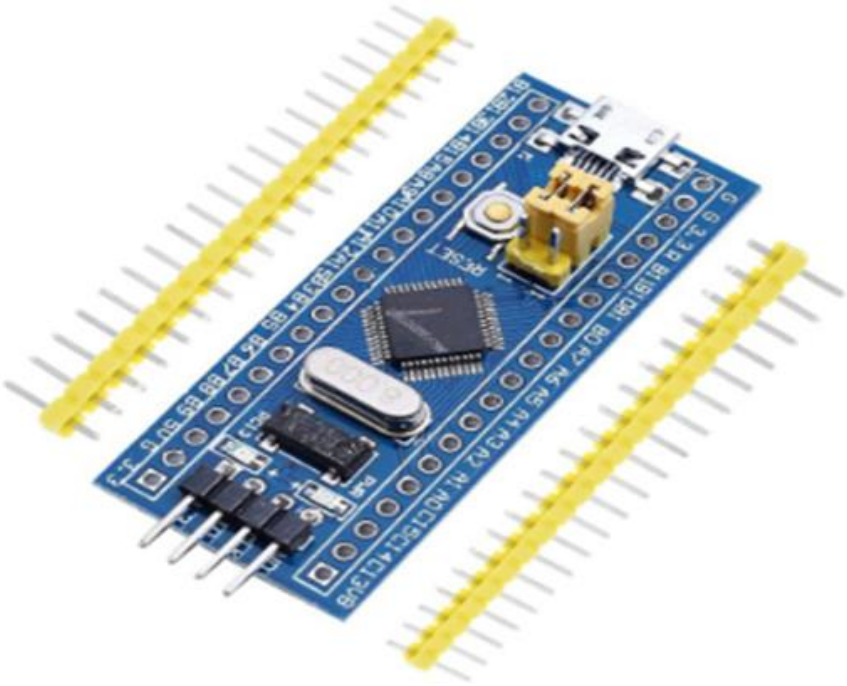

Рисунок 2.1 – Мікроконтроллер STM32

У порівнянні з STM32, інший мікроконтролер, ATmega328 (рисунок 2.2), який широко використовується в платах Arduino Uno, побудований на 8-бітній архітектурі AVR. Він має обмежену тактову частоту до 16 МГц і меншу кількість апаратних ресурсів. Хоча цей мікроконтролер досить зручний у використанні завдяки популярності платформи Arduino та численним бібліотекам, він не забезпечує достатньої обчислювальної потужності для складних розрахунків, пов’язаних із обробкою даних з декількох аналогових каналів, особливо коли потрібно виконувати фільтрацію сигналів, розрахунок миттєвої потужності або передачу даних через сучасні інтерфейси зв’язку. Крім того, ATmega328 має обмежену кількість пам’яті (2 КБ SRAM та 32 КБ Flash), що значно обмежує масштабованість програмного забезпечення у складніших проєктах.

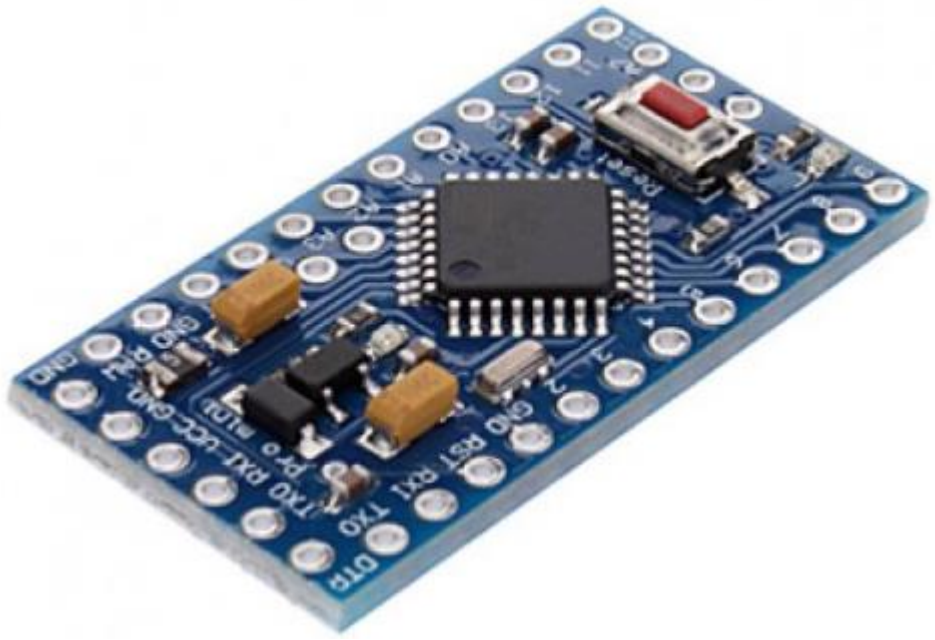

Рисунок 2.2 – Зовнішній вигляд мікроконтроллера ATmega328

Зі свого боку, ESP32 є потужнішим 32-бітним мікроконтролером з інтегрованими модулями Wi-Fi та Bluetooth, двоядерним процесором та широким спектром периферійних інтерфейсів. Цей мікроконтролер чудово підходить для реалізації IoT-проєктів, де важливо забезпечити бездротову передачу даних та взаємодію з хмарними сервісами. Проте ESP32 має й свої особливості: він споживає більше енергії у порівнянні з STM32, що критично у випадках автономного живлення від батареї, а також може бути менш стабільним у реальному часі внаслідок складнішої багатоядерної архітектури та пріоритетів задач. У деяких випадках розробка під ESP32 вимагає глибшого розуміння системного рівня, а бібліотеки можуть бути менш зрілими у порівнянні з тими, що доступні для STM32 у середовищі STM32CubeIDE.

2.1.2 Архітектура STM32F103C8T6

STM32F103C8T6 — це представник сімейства STM32F1 компанії STMicroelectronics, що базується на 32-бітному ядрі ARM Cortex-M3. Цей мікроконтролер розроблено спеціально для задач реального часу з високими

вимогами до обчислювальної ефективності, багатофункціональності та енергоефективності. Ядро Cortex-M3, що є центральним елементом STM32F103C8T6, працює на частоті до 72 МГц та підтримує набір команд Thumb-2, який забезпечує більш щільне кодування інструкцій, зменшуючи розмір прошивки і покращуючи продуктивність. Крім того, архітектура ядра включає систему переривань Nested Vectored Interrupt Controller (NVIC), яка дозволяє обробляти події з мінімальною затримкою — це особливо важливо в системах моніторингу, де своєчасна реакція на зміну параметрів енергоспоживання має вирішальне значення.

STM32F103C8T6 має 64 КБ флеш-пам’яті для збереження програми та 20 КБ оперативної пам’яті (SRAM), що є достатнім обсягом для реалізації алгоритмів збору, обробки та передачі даних. Важливу роль у багатофункціональності мікроконтролера відіграє розвинена система периферійних пристроїв. До складу STM32F103C8T6 входять 10- і 12-бітні аналогово-цифрові перетворювачі (ADC), які дозволяють точно зчитувати аналогові сигнали з сенсорів напруги й струму, зокрема ACS712, HLW8012, INA219 тощо. Завдяки підтримці до 16 аналогових входів і можливості запуску ADC за допомогою таймерів, можна реалізувати синхронізовані цикли вимірювання, що зменшує похибки при обчисленні миттєвої потужності.

Крім аналогової периферії, STM32F103C8T6 підтримує цифрові інтерфейси: UART, SPI, I2C, USB та інші. Вони дозволяють підключати зовнішні пристрої (модулі бездротового зв’язку, дисплеї, карти пам’яті) і організовувати обмін даними з іншими мікроконтролерами або комп’ютерними системами. UART часто використовується для обміну з Bluetooth-модулями (HC-05/HC-06), SPI — для швидкого з’єднання з сенсорами або картами пам’яті, а I2C — для підключення пристроїв, таких як дисплеї. Також підтримуються інтерфейси програмного налагодження (SWD, JTAG), що спрощують відлагодження на етапі розробки.

У питанні енергоспоживання STM32F103C8T6 демонструє високі показники енергоефективності завдяки підтримці декількох режимів енергозбереження: Sleep, Stop та Standby. У режимі Sleep тактовий генератор ядра зупиняється, зберігаючи при цьому роботу периферії, що дозволяє суттєво знизити споживання без втрати продуктивності у фонових задачах. У режимі Stop споживання знижується ще більше, тоді як у Standby — до мінімального значення в одиниці мікроампер. Такі режими дозволяють проєктувати пристрої, які можуть працювати від автономних джерел живлення (акумуляторів, батарей), що є критично важливим для систем моніторингу енергоспоживання в умовах, де недоцільно або неможливо постійне мережеве живлення.

Архітектура STM32F103C8T6 також підтримує використання DMA (Direct Memory Access) — контролера прямого доступу до пам’яті, який дозволяє виконувати обмін даними між периферією та пам’яттю без участі ядра. Це звільняє ресурси процесора та покращує енергозбереження. Ця особливість особливо корисна при реалізації безперервного збору даних з сенсорів у реальному часі або для створення енергоефективного зберігання даних на карту пам’яті чи передачі по інтерфейсах. Такий підхід дозволяє зменшити навантаження на процесор та оптимізувати загальну продуктивність пристрою.

2.1.3 Можливості інтеграції з системами моніторингу та керування

STM32F103C8T6 має широкі можливості інтеграції в системи моніторингу та керування енергоспоживанням завдяки своїй гнучкій апаратній архітектурі, численним периферійним інтерфейсам і підтримці сучасних програмних інструментів. Цей мікроконтролер здатен об’єднати в одному рішенні збір даних з аналогових сенсорів струму та напруги, їхню цифрову обробку та передачу на зовнішні системи.

Підключення до таких сенсорів, як ACS712, HLW8012 або INA219, реалізується через аналогові входи або цифрові інтерфейси. При цьому STM32

дозволяє використовувати як одиничні зразки даних, так і потоки для побудови графіків енергоспоживання в реальному часі. Мікроконтролер підтримує передачу зібраної інформації через UART, I2C або SPI інтерфейси до зовнішніх пристроїв, зокрема модулів Wi-Fi (ESP8266), Bluetooth (HC-05), або дисплеїв, що дозволяє гнучко реалізувати локальний і віддалений моніторинг.

Завдяки USB Full-Speed контролеру, STM32 також можна підключати безпосередньо до ПК для передачі даних або налаштувань. Підтримка зовнішньої пам’яті (наприклад, EEPROM або SD-карти) дозволяє зберігати історію споживання, а можливість реалізації програмного протоколу Modbus по RS-485 — включити пристрій у промислові SCADA-системи. Для керування навантаженням можуть використовуватися цифрові виходи, реле або напівпровідникові ключі, які активуються залежно від заданих алгоритмів або часу доби. Усе це можливо реалізувати завдяки гнучкості мікроконтролера в роботі з таймерами, інтервалами, перериваннями та логікою пріоритетів, а наявність підтримки бібліотек HAL/LL та інтеграції з середовищами STM32CubeIDE, PlatformIO або Arduino значно полегшує розробку прошивок. Важливо зазначити, що STM32 має достатню продуктивність для реалізації базової обробки сигналів, фільтрації та навіть обчислення миттєвої потужності чи коефіцієнта потужності (PF), що дозволяє не лише збирати дані, а й приймати рішення на основі аналітики.

2.2 Огляд датчиків струму, напруги та енергоспоживання

2.2.1 Принцип дії сенсорів ACS712, HLW8012, ZMPT101B

Сенсори ACS712, HLW8012 і ZMPT101B широко використовуються для вимірювання електричних параметрів у побутових та промислових системах, зокрема, при побудові засобів моніторингу енергоспоживання. Кожен із цих

сенсорів має власний принцип дії, який визначає його можливості, точність і тип сигналу, що подається на мікроконтролер.

Сенсор ACS712 — це датчик струму на основі ефекту Холла. Він забезпечує гальванічну розв'язку між вимірювальним колом і мікроконтролером, що підвищує безпеку. Струм, що протікає через провідник у сенсорі, створює магнітне поле, яке вловлюється елементом Холла та перетворюється на аналоговий сигнал, пропорційний струму. Напруга на виході змінюється в межах 0–5 В залежно від напрямку й величини струму. Цей сенсор є особливо зручним для вимірювання змінного й постійного струму з амплітудою до ±5 А, ±20 А або ±30 А, залежно від моделі.

HLW8012 — це інтегральна мікросхема, яка вимірює миттєві значення напруги, струму та активної потужності. Її принцип дії базується на внутрішніх резистивних дільниках напруги та шунтах для струму, після чого дані обробляються за допомогою вбудованих АЦП та цифрового ядра. Вона передає результати у вигляді імпульсів (через виводи CF, CF1), частота яких пропорційна енергетичним параметрам, що значно спрощує цифрове зчитування мікроконтролером. Завдяки високій точності та цифровому виходу HLW8012 ідеально підходить для моніторингу побутових пристроїв, особливо у проєктах, що потребують компактності й надійності.

Сенсор ZMPT101B використовується для вимірювання змінної напруги. Він містить прецизійний трансформатор напруги, який знижує напругу з 220 В до безпечного рівня. Далі сигнал фільтрується та посилюється, і в результаті формується аналоговий сигнал, що подається на АЦП мікроконтролера.

2.2.2 Точність вимірювання та межі струму/напруги

Точність вимірювання електричних параметрів і діапазони, в яких працюють сенсори, є важливими аспектами для забезпечення надійності систем моніторингу енергоспоживання. Розглядаючи сенсори, варто оцінювати їхні метрологічні характеристики, а також фактори, які впливають на точність.

Сенсор ACS712 має три варіанти: ±5 А, ±20 А і ±30 А, що дозволяє підбирати модель залежно від передбачуваного струмового навантаження. Вихідний аналоговий сигнал лінійно залежить від струму, але точність суттєво залежить від шумів, температурної нестабільності та похибки внутрішнього елемента Холла. Типова похибка становить ±1.5% при номінальних умовах, однак для побутових застосувань така точність є прийнятною. Для покращення результатів рекомендується усереднення результатів із кількох вимірів на АЦП, а також калібрування в конкретному застосуванні.

У свою чергу, HLW8012 є цифровим сенсором, і завдяки внутрішнім АЦП забезпечує високу точність: до ±1% для напруги та струму, та близько ±2% для активної потужності, згідно з документацією. Його діапазон струму обмежений лише типом зовнішнього шунта, а напруга зазвичай вимірюється у межах 100–250 В змінного струму. Точність залежить від якості зовнішніх елементів — шунта та резисторного дільника, що вимагає уважного підбору під час проєктування.

Для ZMPT101B діапазон вимірюваної напруги залежить від налаштування підсилення на платі, однак найчастіше використовується для 220 В змінного струму. Попри простоту, цей сенсор має обмежену точність — близько ±5% — через вплив шумів, температурних змін і нелінійності трансформатора. Коректна фільтрація, калібрування та використання зовнішніх опорних напруг покращують результати, однак для задач високої точності краще використовувати альтернативи.

2.2.3 Особливості підключення до STM32 та обробка сигналів

Підключення сенсорів струму та напруги до мікроконтролера STM32 потребує врахування як електричних, так і логічних характеристик як самого мікроконтролера, так і датчиків.

Для сенсора ACS712, який генерує аналоговий вихід, потрібен один канал АЦП (ADC) STM32. Напруга на виході змінюється в межах 0–5 В, але більшість мікроконтролерів STM32 (зокрема STM32F103C8T6) працюють з напругою 3.3 В. Тому для безпечного зчитування сигналу через АЦП необхідно використовувати дільник напруги або забезпечити живлення самого датчика від 3.3 В, якщо це допускається його характеристиками. Також важливо враховувати шумність сигналу — типове рішення передбачає використання фільтра нижніх частот на вході АЦП.

Сенсор ZMPT101B, який також має аналоговий вихід, підключається за схожою схемою, однак варто звертати увагу на правильне налаштування посилення вбудованого операційного підсилювача на модулі. Високий рівень шуму цього сенсора потребує ретельного фільтрування та калібрування. Дані з аналогових сенсорів надходять у вигляді синусоїдального сигналу, тому для обробки в мікроконтролері необхідне дискретне вибіркове вимірювання, усереднення або застосування методів цифрової обробки сигналів, таких як вирахування середньоквадратичного значення (RMS).

Інакше працює HLW8012, який надає цифровий вихід у вигляді імпульсів з частотою, пропорційною потужності, напрузі або струму. Цей сенсор підключається до цифрових входів мікроконтролера і вимагає вимірювання частоти імпульсів, що зручно реалізується через таймери STM32 або лічильники переривань. Для аналізу, наприклад, на виводі CF формується сигнал частотою, пропорційною активній потужності, і STM32 може використовувати вбудований таймер для підрахунку кількості імпульсів за певний час або вимірювання періоду імпульсів. Важливо, що HLW8012 використовує один вивід SEL для перемикання між режимами струму і напруги, що потребує додаткового управління з боку мікроконтролера.

2.2.4 Альтернативні варіанти сенсорів

Окрім поширених датчиків ACS712, HLW8012 і ZMPT101B, у системах моніторингу енергоспоживання активно застосовуються альтернативні сенсори, які пропонують вищу точність, розширені можливості та простішу цифрову інтеграцію.

Одним із таких є INA219 — цифровий сенсор від Texas Instruments, призначений для вимірювання струму, напруги і потужності. Цей датчик має вбудований 12-розрядний АЦП і працює через інтерфейс I2C, що значно спрощує підключення до мікроконтролера STM32. INA219 може вимірювати струм через шунтовий резистор з точністю до ±1%, а також розраховує напругу на шині живлення та споживану потужність. Його діапазон струмів — до ±3.2 А (залежно від шунта), напруга — до 26 В, що робить його придатним для низьковольтних постійних систем, зокрема у живленні через USB, акумулятори чи сонячні контролери. Значною перевагою є висока точність і можливість інтегрувати декілька сенсорів у одну I2C-шину.

Інший варіант — PZEM-004T, модуль промислового рівня, що вимірює напругу, струм, потужність і енергію у змінному струмі 220 В. Цей сенсор зазвичай постачається з трансформатором струму, підключення здійснюється через послідовний UART-інтерфейс. Пристрій підтримує обмін даними у вигляді запит–відповідь і має достатньо високу точність (±1% для напруги та струму, ±2% для потужності). Його зручно використовувати у побутових проєктах, оскільки не потребує складного налаштування і забезпечує повний набір параметрів для аналізу. Особливо цінним є вбудований лічильник спожитої енергії, що дозволяє вести облік у кВт·год. При інтеграції з STM32 слід враховувати рівні логіки UART (3.3 В для STM32), а також використовувати ізоляцію за потреби, оскільки PZEM працює з небезпечною напругою.

Залежно від задачі, INA219 краще підходить для постійного струму й мобільних застосувань, а PZEM-004T ідеально підходить для постійного моніторингу побутових пристроїв, які живляться від мережі.

2.3 Вибір інтерфейсів передачі даних

2.3.1 UART, I2C, SPI — призначення, переваги, особливості використання

У сучасних мікроконтролерних системах ефективна передача даних між компонентами є важливою для стабільної роботи пристрою. Серед основних інтерфейсів, які підтримуються STM32, слід виділити UART, I2C та SPI — кожен із них має свої переваги, обмеження та сценарії застосування.

Інтерфейс UART (Universal Asynchronous Receiver-Transmitter) є одним із найпоширеніших способів обміну даними. Він забезпечує двонаправлену передачу інформації з використанням лише двох ліній — TX (передача) і RX (прийом). UART не потребує спільного тактового сигналу між пристроями, що спрощує підключення. Протокол широко використовується для зв’язку з комп'ютером, Bluetooth-модулями (HC-05, HC-06), GSM-модулями та багатьма іншими периферійними пристроями.

I2C (Inter-Integrated Circuit) — це синхронний інтерфейс, який використовує лише дві лінії: SDA (дані) та SCL (тактова лінія). Він дозволяє підключати до одного контролера кілька пристроїв (до 127 теоретично), завдяки адресації. I2C ідеально підходить для підключення сенсорів, EEPROM, дисплеїв (наприклад, OLED або LCD з адаптером), годинників реального часу тощо. Основною перевагою є простота реалізації та можливість підключення багатьох пристроїв через спільну шину.

SPI (Serial Peripheral Interface) — ще один синхронний інтерфейс, який забезпечує значно вищу швидкість обміну даними порівняно з I2C. SPI використовує чотири лінії: MOSI (Master Out Slave In), MISO (Master In Slave

Out), SCLK (тактова лінія), та SS/CS (вибір підпорядкованого пристрою). Завдяки своїй високій швидкості та простій реалізації, SPI широко використовується для зв’язку з швидкодіючими пристроями, такими як АЦП, флеш-пам'ять, дисплеї та модулі передачі даних.

STM32 підтримує апаратну реалізацію всіх трьох протоколів, що дає розробнику гнучкість у виборі інтерфейсу залежно від конкретних задач. У системах моніторингу енергоспоживання I2C та SPI зазвичай застосовуються для підключення сенсорів, тоді як UART часто використовується для зв’язку з зовнішніми пристроями або модулями бездротової передачі даних.

2.3.2 Вибір оптимального інтерфейсу для віддаленого доступу

Забезпечення віддаленого доступу до даних моніторингу енергоспоживання є ключовим завданням для сучасних систем автоматизації побуту. Вибір оптимального інтерфейсу для такої взаємодії залежить від низки факторів: від вимог до швидкості передавання даних і енергоспоживання — до складності реалізації, наявної інфраструктури, способу взаємодії з користувачем і можливості масштабування.

Для локального доступу на невеликій відстані ідеальним варіантом виступає Bluetooth, зокрема модулі HC-05 або HC-06. Цей інтерфейс дозволяє безпосередньо з’єднувати мобільний пристрій користувача із системою без потреби в маршрутизаторі або доступі до Інтернету. У поєднанні з відповідним застосунком (наприклад, на Android), можна легко реалізувати інтерфейс перегляду показників, журналу споживання або навіть керування пристроями.
У випадках, коли потрібно мати доступ з будь-якої точки світу або інтегрувати пристрій у хмарну екосистему, Wi-Fi є набагато кращим вибором. Модулі ESP8266 або ESP01 дозволяють легко інтегрувати мікроконтролер у локальну мережу та передавати дані до серверів або платформ типу Blynk, ThingSpeak, Home Assistant, MQTT-брокерів тощо. Це відкриває широкі можливості:

надсилання сповіщень, побудова графіків, розширена аналітика, зберігання історії, а також створення тригерів для автоматизації. Wi-Fi також дозволяє реалізувати OTA-оновлення прошивки (Over-The-Air), що особливо важливо для підтримки пристрою після встановлення.

У деяких випадках для стабільного та надійного з'єднання, особливо в умовах промислового використання або у великих будинках, доцільно використовувати провідні інтерфейси, такі як RS-485, що забезпечує зв'язок на великі відстані і стійкість до завад. Також популярним є Ethernet, проте він рідше використовується в побутових умовах через складність підключення і більші габарити пристроїв.

Кінцевий вибір інтерфейсу має базуватись на таких критеріях:

1. Технологічне середовище (наявність Wi-Fi мережі або бажання автономної роботи);
2. Енергоспоживання (Wi-Fi споживає значно більше енергії, ніж Bluetooth);
3. Дальність зв'язку (Wi-Fi — до десятків метрів; Bluetooth — 5–10 м);
4. Потреба у хмарній аналітиці чи звичайному локальному перегляді даних.

У багатьох випадках розумним рішенням є комбіноване використання кількох інтерфейсів, де Bluetooth використовується для початкового налаштування та резервного доступу, а Wi-Fi — для повноцінного моніторингу та взаємодії з мережею Інтернет.

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

У цьому розділі детально розібрано апаратну платформу комп’ютерного засобу на базі мікроконтролера STM32, призначеного для моніторингу та управління енергоспоживанням побутових пристроїв.

Здійснено порівняльний аналіз мікроконтролера STM32F103C8T6 з популярними альтернативами, такими як ESP32 та ATmega328. На основі порівняння обґрунтовано вибір STM32 завдяки його оптимальному балансу між обчислювальною потужністю, енергоефективністю, широким набором периферійних інтерфейсів та розвиненою програмною підтримкою. Детально розглянуто архітектуру STM32F103C8T6, включаючи ядро Cortex-M3, обсяги пам'яті, можливості периферії та режими енергозбереження, що підкреслює його придатність для задач реального часу та автономної роботи. Також розглянуто можливості інтеграції STM32 з різноманітними системами моніторингу та керування завдяки його гнучким інтерфейсам та підтримці різних протоколів.

Розглянуто датчики струму, напруги та енергоспоживання, які можуть бути використані в розробці. Детально проаналізовано принцип дії, точність вимірювання та особливості підключення таких сенсорів, як ACS712, HLW8012 та ZMPT101B. Окрему увагу приділено альтернативним варіантам сенсорів, таким як INA219 та PZEM-004T, які пропонують покращені характеристики та спрощену інтеграцію.

Розглянуто ключові інтерфейси передачі даних, доступні на мікроконтролері STM32: UART, I2C та SPI. Описано їхнє призначення, переваги та особливості використання та наведено приклади їхнього застосування в контексті систем моніторингу енергоспоживання. Особливу увагу приділено вибору оптимального інтерфейсу для забезпечення віддаленого доступу до даних, порівняно можливості Bluetooth та Wi-Fi, а також розглянуто провідні альтернативи.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Огляд варіантів інтерфейсу користувача

При розробці комп’ютерних засобів моніторингу та управління енергоспоживанням побутових пристроїв значну увагу було приділено створенню інформативного, зручного та функціонального інтерфейсу користувача. Оскільки інтерфейс виступає головним засобом взаємодії користувача з системою, його структура, зручність, гнучкість і візуальне подання даних мають прямий вплив на ефективність використання всієї системи. У цій частині пояснювальної записки буде розглянуто основні типи інтерфейсів, які були потенційними варіантами під час проєктування, а також обґрунтовано вибір реалізованого веб-інтерфейсу.

Одним з поширених варіантів є створення GUI (графічного інтерфейсу користувача**)** у вигляді десктопного застосунку, реалізованого у середовищі Qt, JavaFX або Windows Forms. Такий підхід дозволяє створити локальний застосунок для операційних систем Linux, Windows чи macOS, який забезпечує пряме з’єднання із пристроєм через USB, COM-порт або IP-протокол. Перевагами такого варіанту є:

- Висока швидкодія;
- Прямий доступ до обладнання;
- Можливість зберігати великі обсяги даних локально;
- Інтеграція з локальними службами (принтер, файлова система).

До недоліків створення графічного інтерфейсу користувача можна віднести необхідність встановлення програмного забезпечення, менша гнучкість при віддаленому доступі та доволі складна розробка й оновлення.

Ще одним варіантом є створення мобільного додатку. Такі застосунки дозволяють забезпечити постійний моніторинг і керування енергоспоживанням через смартфон, що є доволі зручним в повсякденному використанні для користувачів.

Переваги даного варіанту інтерфейсу:

- Змога контролю для користувача через власний смартфон;
- Можливість надсилання сповіщень;
- Інтеграція з голосовими помічниками, геолокацією.

Недоліки стоврення мобільного застосунку:

- Потреба робити регулярні оновлення, від яких залежить функціонал;
- Технологічно складна розробка;
- Залежність від хмарної інфраструктури.

Основним обраним рішенням для реалізації інтерфейсу стала розробка веб-інтерфейсу, що дозволяє отримувати доступ до системи з будь-якого пристрою, підключеного до локальної мережі або Інтернету. Цей варіант рішення є зручним, універсальним і масштабованим, особливо для користувача. До переваг можна віднести:

- Гнучке налаштування інтерфейсу;
- Відсутність потреби в інсталяції;
- Доступність через будь-який сучасний браузер;
- Можливість легкої адаптації під мобільні пристрої;
- Простота у тестуванні та демонстрації роботи.

3.2 Вибір мови програмування та інструментів розробки

Під час розробки комп'ютерного засобу для моніторингу та управління енергоспоживанням побутових пристроїв, було прийнято рішення створити симульовану програмну систему з можливістю запуску на персональному комп’ютері, без використання фізичного обладнання на етапі демонстрації. Такий підхід дозволив зосередитися на логіці взаємодії користувача з пристроями, обробці та візуалізації даних, що є ключовими аспектами даної системи. Для цього були обрані сучасні, популярні та добре підтримувані інструменти, які дозволяють швидко реалізувати інтерактивний інтерфейс та гнучкий сервер даних.

Для реалізації веб-інтерфейсу користувача було обрано стандартний стек технологій HTML та JavaScript. HTML забезпечує структуру веб-сторінки, а JavaScript — динамічність, взаємодію з сервером та графіки. Особливу роль відіграє бібліотека Chart.js — вона дозволяє легко й гнучко створювати інтерактивні лінійні графіки, що візуалізують споживання енергії кожного пристрою в режимі реального часу. Візуальні індикатори, кольорове кодування ефективності (зелений, жовтий, червоний) та динамічне оновлення елементів інтерфейсу створюють зручний інструмент спостереження за пристроями.

На стороні сервера було вирішено використати середовище Node.js — це асинхронна, подієво-орієнтована платформа для серверної розробки, яка дозволяє обробляти численні запити без затримок і блокувань. Основна серверна логіка реалізована у файлі server.js за допомогою фреймворку Express.js, що значно спрощує створення RESTful API та маршрутизації. Сервер обслуговує статичні файли з каталогу public, обробляє запити до API, зберігає історію споживання у вигляді JSON-файлу, а також взаємодіє з симулятором мікроконтролера через TCP-сокет.

Для емуляції мікроконтролера STM32 створено окрему програму — stm32_emulator.py, написану на мові Python. Python обрано через його простоту,

гнучкість та наявність вбудованих засобів для роботи з мережею, генерації випадкових даних, логування та багатопоточності. Програма створює TCP-клієнт, який з'єднується із сервером Node.js, передає дані про стан пристроїв (струм, напруга, потужність, стан увімкнення/вимкнення), а також MCU-статус (температура, Vcc, сигнал, режими тощо). Симулятор також реалізує можливість примусових збоїв, перевантажень, аварій та випадкових змін сенсорів, що наближує роботу до реальних умов.

Обмін даними між сервером та емулятором відбувається через простий текстовий протокол з використанням ключових команд (STATUS, TOGGLE, FORCE_FAIL, SET_MODE тощо). Сервер надсилає команди, а емулятор відповідає повідомленнями про стан кожного пристрою. Таке розділення дозволяє реалізувати реальну модель «пристрій – сервер – користувач» із можливістю майбутнього підключення фізичного STM32 через той самий TCP-протокол або перенесення логіки в мікропрошивку.

Як середовище розробки було використано Visual Studio Code — зручний та функціональний редактор, який підтримує автодоповнення, налагодження та одночасну роботу з кількома мовами програмування. Для роботи з HTML, CSS та JavaScript використовувались розширення Live Server і Prettier, а для розробки серверної частини — підтримка Node.js і плагіни для налагодження Express-додатків. Емулятор писався та відлагоджувався у середовищі Python IDLE та запуском через командний рядок.

3.3 Створення програмного забезпечення для мікроконтролера та веб-інтерфейсу

Розробка програмного забезпечення комп’ютерних засобів для моніторингу та керування енергоспоживанням побутових пристроїв передбачала створення трьох взаємопов’язаних компонентів: веб-інтерфейсу користувача, серверної частини для обробки запитів і збереження даних, а

також програмного симулятора мікроконтролера STM32, який відтворює поведінку реального пристрою. Кожен з цих компонентів виконує важливу функцію в архітектурі системи та взаємодіє з іншими за допомогою чітко визначених інтерфейсів.

На рисунку 3.1 представлено вміст папки energy-monitoring, яка містить в собі файли веб-інтерфейсу, емулятора та серверу.

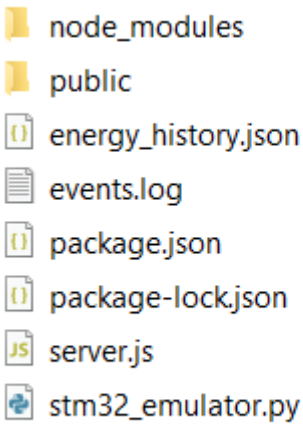

Рисунок 3.1 – Вміст файлової системи розробленої програми

Файл package.json є конфігураційним файлом проекту. Він описує проєкт: назва, версія, короткий опис, автор; вказує основний файл запуску; містить список залежностей, які потрібні для роботи проекту; описує npm-скрипти для запуску в JSON форматі. На рисунку 3.2 наведений вміст файлу package.json.

```
{
  "name": "energy-monitoring",
  "version": "1.0.0",
  "description": "Комп'ютерний прототип моніторингу та управління енергоспоживанням на Node.js та STM32",
  "main": "server.js",
  ▷ Отладка
  "scripts": {
    "start": "node server.js"
  },
  "author": "fantomas00",
  "license": "ISC",
  "dependencies": {
    "express": "^4.21.2"
  },
  "keywords": []
}
```

Рисунок 3.2 – Вміст конфігураційного файлу проєкту

Веб-інтерфейс реалізований за допомогою HTML і JavaScript й відображає в реальному часі інформацію про стан кожного пристрою — його назву, потужність, напругу, струм, стан увімкнення, наявність аварії, температурні показники, спрацювання датчиків руху та історію навантаження. Завдяки бібліотеці Chart.js кожен пристрій має власний графік, що дозволяє користувачеві відстежувати динаміку споживання енергії. Крім того, користувач має змогу перемикати режими роботи системи, вмикати або вимикати пристрої, переглядати загальну потужність та сумарне споживання енергії. Додано також візуальні індикатори ефективності (низьке/середнє/високе споживання), що дозволяє інтуїтивно оцінювати стан системи.

Файл index.html має такі основні елементи:

- — відображає інформацію про мікроконтролер STM32;
- <select id="modeSelect"> — дозволяє змінювати режим роботи;
- — контейнер, у який динамічно вставляються картки пристроїв;
- <canvas id="chart_X"> — елементи для графіків Chart.js;
- та — показники загальної потужності та спожитої енергії.

Основні функції JavaScript:

1. fetchMcuStatus()

   Запитує /api/mcu_status і оновлює інформацію про температуру, Vcc, живлення, сигнал та режим.

2. setMode(mode)

   Надсилає команду зміни режиму /api/set_mode?mode=... і блокує повторне оновлення select протягом 1 секунди.

3. fetchData()

   Основна функція оновлення інтерфейсу. Запитує /api/data, перебирає при-

строї, створює або оновлює блоки HTML для кожного, генерує графіки Chart.js, підраховує загальну потужність, додає до історії.

4. toggleDevice(name)
   Відправляє запит /api/toggle?device=..., щоб перемкнути стан пристрою.

Усередині тегу <style> прописано адаптивне та інформативне оформлення:

- елементи мають закруглені краї, тіні, згладжене оформлення;
- device-alert застосовується у випадках аварії (використовується @keyframes alert-blink);
- кольорові класи high, medium, low виділяють споживання.

На рисунку 3.3 зображено CSS-правило для класу .mcu-block з файлу index.html.

```
.mcu-block {
  max-width: 700px;
  margin: 0 auto 20px auto;
  background: #e0f2f1;
  border-radius: 12px;
  padding: 18px;
  font-size: 17px;
  color: #01443e;
}
```

Рисунок 3.3 – Класс для стилізації блоку мікроконтролера

Опис характеристик:

- max-width: 700px; — максимальна ширина блоку 700 пікселів.
- margin: 0 auto 20px auto; — зовнішній відступ: зверху 0, злів і справа автоматично (що центрує блок по горизонталі), знизу 20px.
- background: #e0f2f1; — світлий бірюзовий фон блоку.
- border-radius: 12px; — закруглені кути блоку (радіус 12 пікселів).
- padding: 18px; — внутрішній відступ усередині блоку — 18 пікселів.
- font-size: 17px; — розмір шрифту тексту в блоці — 17 пікселів.
- color: #01443e; — темно-зелений колір тексту.

Файл server.js, який побудований на Node.js, має реалізовані основні маршрути API. Обробляються такі запити:

- /api/data — повертає структуровану інформацію про всі пристрої, їх стан, аварійність, поточні параметри, статуси сенсорів тощо.
- /api/mcu_status — повертає поточний стан контролера (температура, напруга живлення, якість живлення, останній reset, рівень сигналу, режим роботи).
- /api/toggle — дозволяє увімкнути або вимкнути вибраний пристрій через надсилання команди MCU.
- /api/set_mode — змінює режим роботи всієї системи (економний, діагностика, нічний, ручний).
- /api/history — повертає масив історичних записів енергоспоживання для побудови графіків у веб-інтерфейсі.
- /api/force_fail — ініціює аварійну ситуацію на вибраному пристрої з заданим типом та тривалістю.
- /api/reset — скидає аварійний стан чи перевантаження на пристрої.
- /api/set_power_lvl — встановлює рівень споживаної потужності для пристрою.
- /api/status — отримує розширений поточний статус пристрою (сенсори, аварії, температура).

Основні бібліотеки, які використовуються у файлі server.js:

- express — для створення HTTP-сервера та обробки REST API-запитів;
- net — для організації TCP-з'єднання з емулятором STM32;
- fs — для зберігання та завантаження історії енергоспоживання у файл;
- path — для коректної роботи з файловою системою.

Сервер підтримує TCP-з’єднання з симулятором, отримуючи від нього регулярні пакети даних у вигляді рядків, які розділяються на назви пристроїв і числові параметри. Після обробки ці дані зберігаються у JSON-структурах та надсилаються клієнту у форматі, зручному для обробки у браузері.

Також сервер реалізує логіку збереження історії. Усі записи, що надходять від емулятора, періодично додаються до масиву, який серіалізується у файл energy_history.json. Це дозволяє підтримувати короткотривалу історію (до 500 записів), яку згодом можна використовувати для побудови графіків або аналізу змін навантаження. Функція addHistoryRecord() викликається після кожного оновлення даних, і вся логіка відділена від основної взаємодії API.

Емулятор, написаний на Python, зчитує конфігурацію пристроїв з вбудованого словника, імітує їхню поведінку з урахуванням випадкових коливань напруги, струму, внутрішніх збоїв, сенсорних помилок і реакцій на команди з сервера. Кожен пристрій має свої номінальні характеристики, межі споживання, ймовірності збоїв та унікальні сценарії. Крім того, реалізовано сенсори температури, руху, статус сенсора (помилка/відсутній/ок) тощо. Емулятор передає значення у вигляді тексту, що аналізується сервером у реальному часі.

У файлі stm32_emulator.py використовуються стандартні бібліотеки Python:

- socket – для TCP-з'єднання з сервером;
- time, random – для генерації випадкових подій, часових затримок;
- threading – для паралельної обробки команд від сервера;
- datetime – для формування міток часу та логування.

Також у цьому файлі використані такі ключові константи та початкові налаштування:

- HOST, PORT – адреса та порт сервера для підключення.
- LOG_FILE – файл для збереження журналу подій.
- MCU_MODES – список можливих режимів роботи контролера (eco, diagnostic, manual, night).
- DEVICE_PARAMS – словник з параметрами кожного пристрою (номінальна напруга, струм, потужність, ліміт потужності).

- FAIL_SCENARIOS – набір можливих аварійних сценаріїв для кожного пристрою: тип аварії, тривалість, імовірність настання.

На рисунку 3.4 зображено словник з параметрами кожного пристрою.

```
DEVICE_PARAMS = {
    "fridge":       {"limit": 170, "nominal": (220, 0.7, 120)},
    "microwave":    {"limit": 220, "nominal": (220, 1.2, 180)},
    "tv":           {"limit": 65,  "nominal": (220, 0.3, 55)},
    "coldchamber":  {"limit": 190, "nominal": (220, 0.9, 150)},
    "heater":       {"limit": 1800,"nominal": (220, 8.5, 1600)},
    "pc":           {"limit": 350, "nominal": (220, 1.7, 250)},
}
```

Рисунок 3.4 – Перелік усіх пристроїв та їх параметрів

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

У третьому розділі розглянуто процес розробки програмного забезпечення для системи моніторингу та управління енергоспоживанням побутових пристроїв на базі STM32. На основі аналізу варіантів інтерфейсів користувача було обґрунтовано вибір веб-інтерфейсу як найбільш доступного та гнучкого рішення, що не потребує спеціального програмного забезпечення для запуску й забезпечує кросплатформеність. Веб-інтерфейс реалізовано за допомогою HTML, JavaScript та бібліотеки Chart.js для побудови графіків у реальному часі, що дозволяє користувачеві ефективно відстежувати стан і роботу підключених пристроїв.

Вибір мови програмування та інструментів розробки був зумовлений необхідністю одночасно моделювати роботу мікроконтролера STM32 та створити повноцінний сервер обміну даними. Для цього було застосовано Python (для емулятора STM32) та Node.js з Express (для серверної частини). Симулятор передає дані в режимі реального часу через TCP, а сервер забезпечує обробку, маршрутизацію, збереження історії та API-взаємодію.

Також, було представлено повну реалізацію ключових програмних компонентів, зокрема: серверної частини (server.js), що обробляє запити, зберігає історію споживання, підтримує TCP-з’єднання з емулятором; клієнтського інтерфейсу (index.html), що забезпечує динамічну візуалізацію, перемикання режимів, керування пристроями; емулятора (stm32_emulator.py), що моделює поведінку пристроїв і сенсорів, реагує на команди.

Розроблене програмне забезпечення дозволяє повністю змоделювати роботу реальної системи моніторингу, забезпечуючи всі ключові функції: збір даних, візуалізацію, аналіз, керування та діагностику.

4. ТЕСТУВАННЯ ТА ОПИС РОБОТИ СИСТЕМИ

Перед тим, як запустити програму, слід відкрити папку, в якій знаходяться усі необхідні файли проєкту. Потім, у цій папці необхідно натиснути комбінацію клавіш “Win + R”, прописати команду “cmd” та натиснути клавішу ENTER для того, щоб відкрився командний рядок у системі Windows. Важливим кроком буде прописати шлях до папки, де знаходяться файли проекту. Далі, слід запустити сервер командою “node server.js” та, в разі успіху, отримати відповідне повідомлення.

На рисунку 4.1 представлений приклад командного рядку:

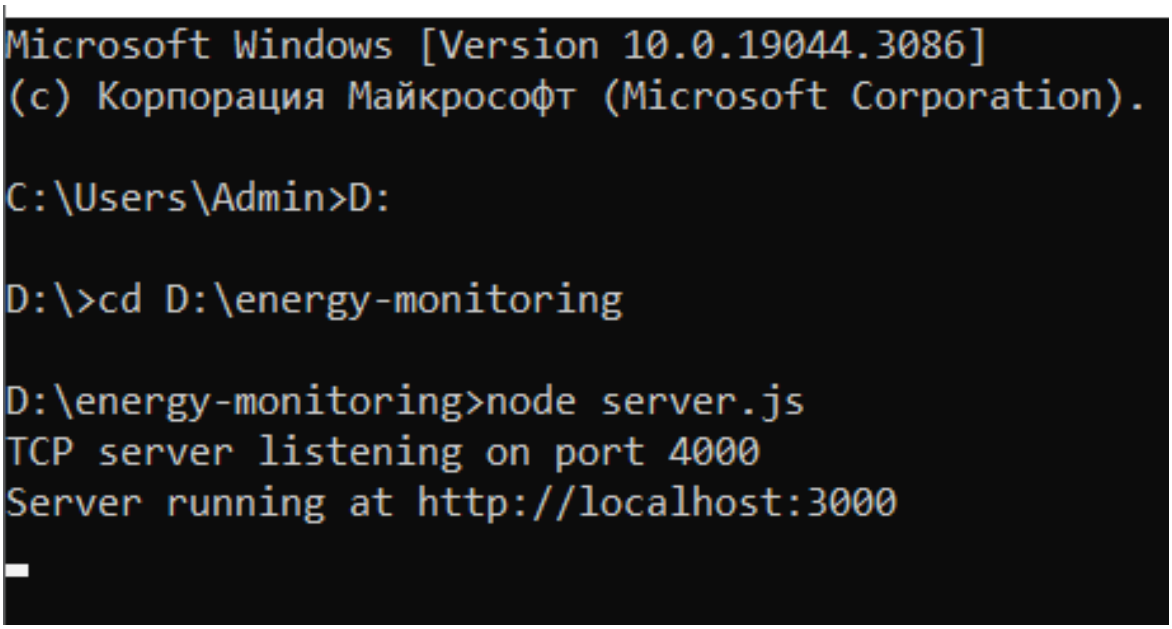

Рисунок 4.1 – Приклад успішного запуску файлу server.js через командний рядок

Виконавши попередні дії, слід відкрити ще один командний рядок у цій же папці для того, щоб запустити емулятор мікроконтролера STM32. Якщо все буде працювати вірно, то в першому командному рядку з’явиться повідомлення, що емулятор STM32 був успішно підключений.

На рисунку 4.2 представлений набір команд для підключення емулятора до сервера.

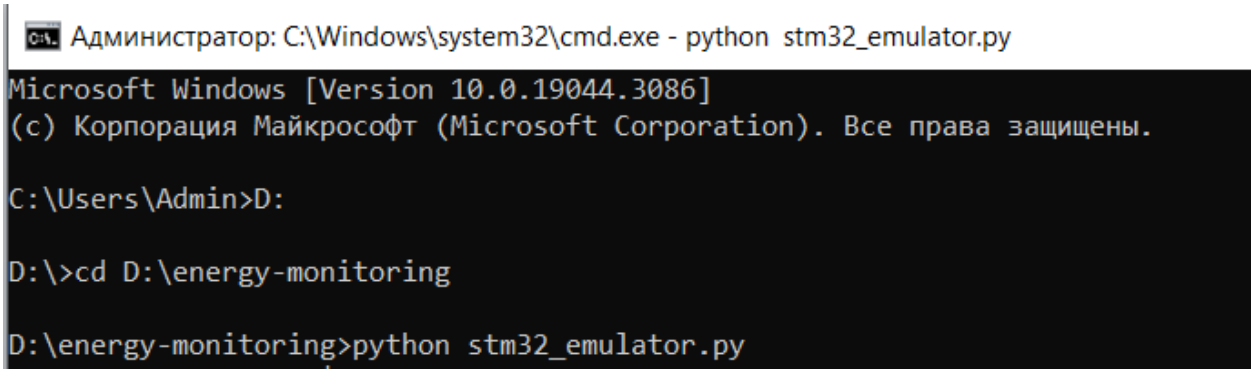

Рисунок 4.2 – Запуск емулятора STM32

Тепер можна запустити сам веб-сайт. Основні елементи інтерфейсу:

– блок інформації про стан MCU (температура, напруга, стан живлення, дата останнього скидання, режим роботи);
– випадаючий список для зміни режиму роботи;
– список пристроїв у вигляді окремих карток;
– графік потужності для кожного пристрою;
– блок загальної статистики (загальна потужність, сумарна енергія);
– повідомлення про помилки або аварійні стани.

На рисунках 4.3.1 та 4.3.2 показаний початковий стан користувацького інтерфейсу.

Рисунок 4.3.1 – Загальний вигляд панелі моніторингу енергоспоживання (перша частина)

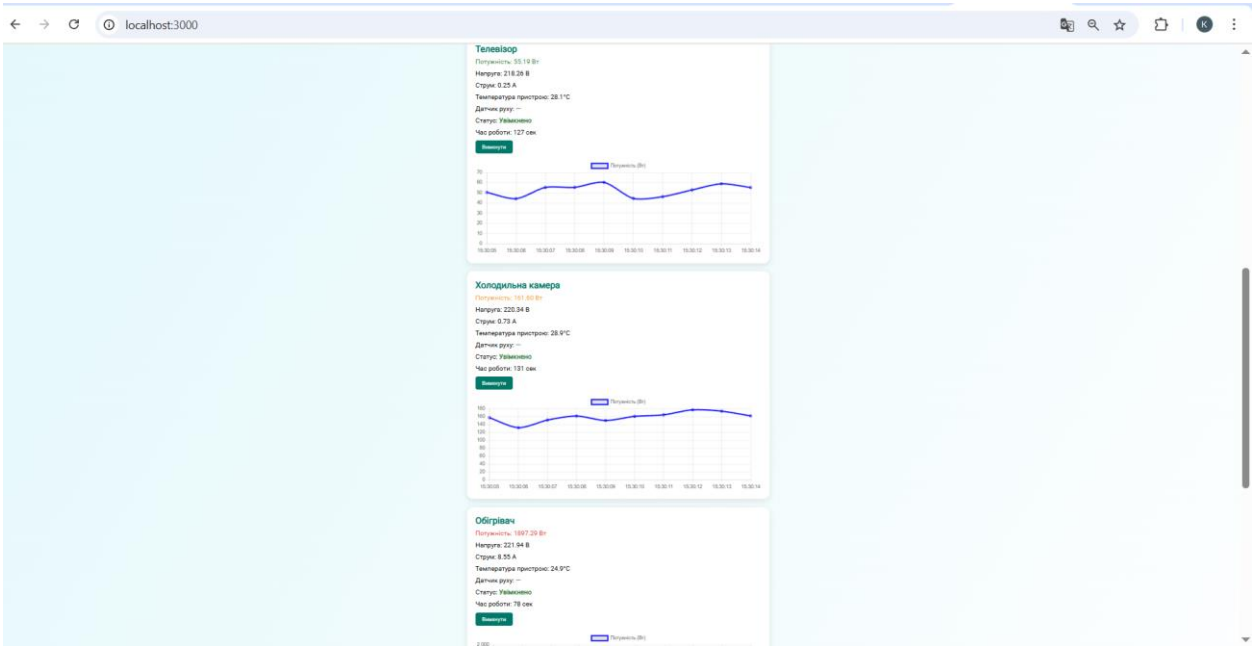

Рисунок 4.3.2 – Загальний вигляд панелі моніторингу енергоспоживання (друга частина)

Веб-інтерфейс включає в себе сторінку, на якій відображаються 6 різних побутових приладів. Кожний прилад має такі характеристики:

- Потужність (у Ватах)
- Напруга (у Вольтах)
- Струм (в Амперах)
- Температура пристрою (за Цельсієм)
- Датчик руху (Рух є або відсутній)
- Статус (увімкнути/вимкнути)
- Час роботи (у секундах)

На рисунку 4.4 показано приклад зображення пристрою та його характеристик.

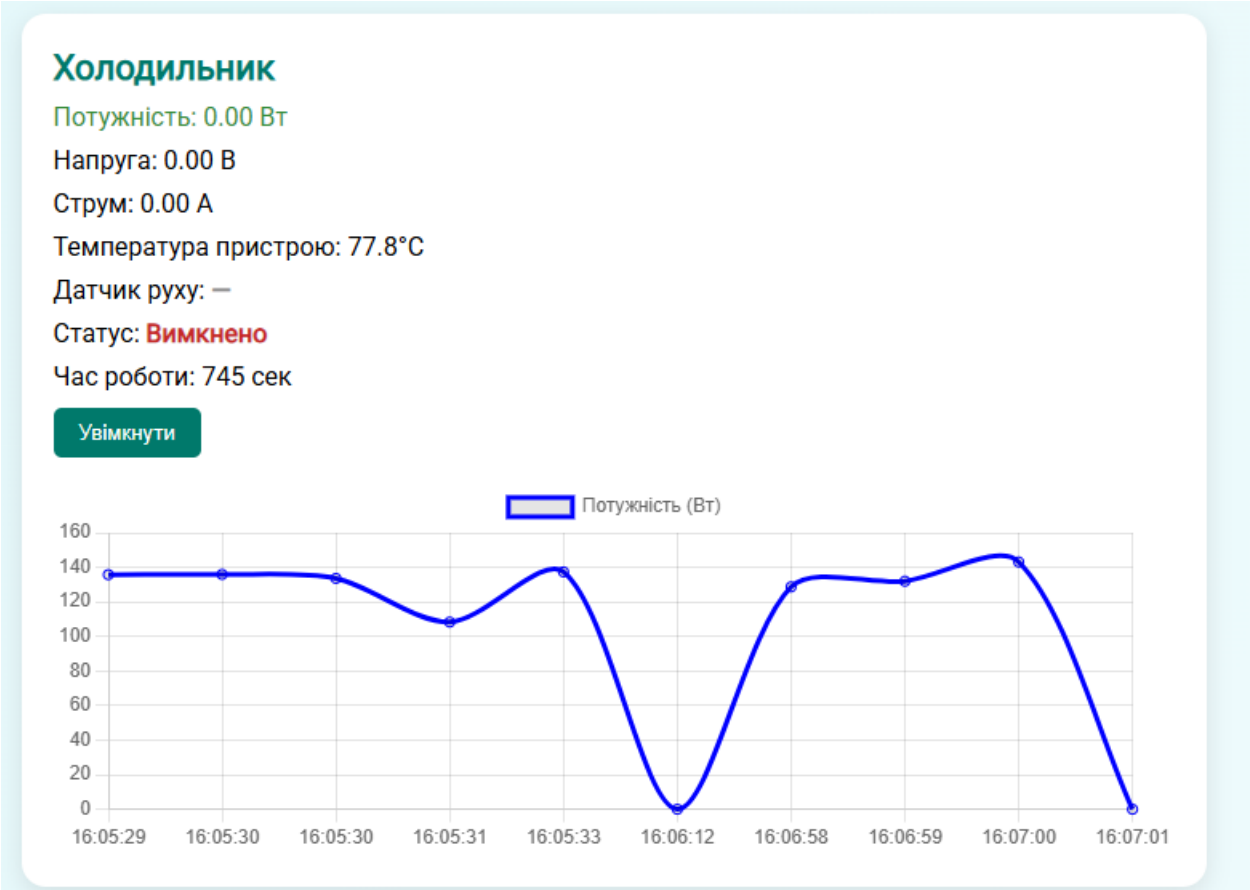

Рисунок 4.4 – Характеристики обраного приладу

У реалізації цієї програми також передбачені критичні та аварійні ситуації. На рисунку 4.5 можна побачити автоматичне вимкнення обігрівача на певний час через перегрів самого пристрою. На рисунку 4.6 демонструється втрата сигналу на деякий час для телевізора через несправність сенсорів. На рисунку 4.7 зображено критичну ситуацію, коли двері холодильника були відкриті занадто довго і він був вимкнутий після цього. На рисунку 4.8 пристрій має перевищення допустимої потужності й ця ситуація вважається аварійною.

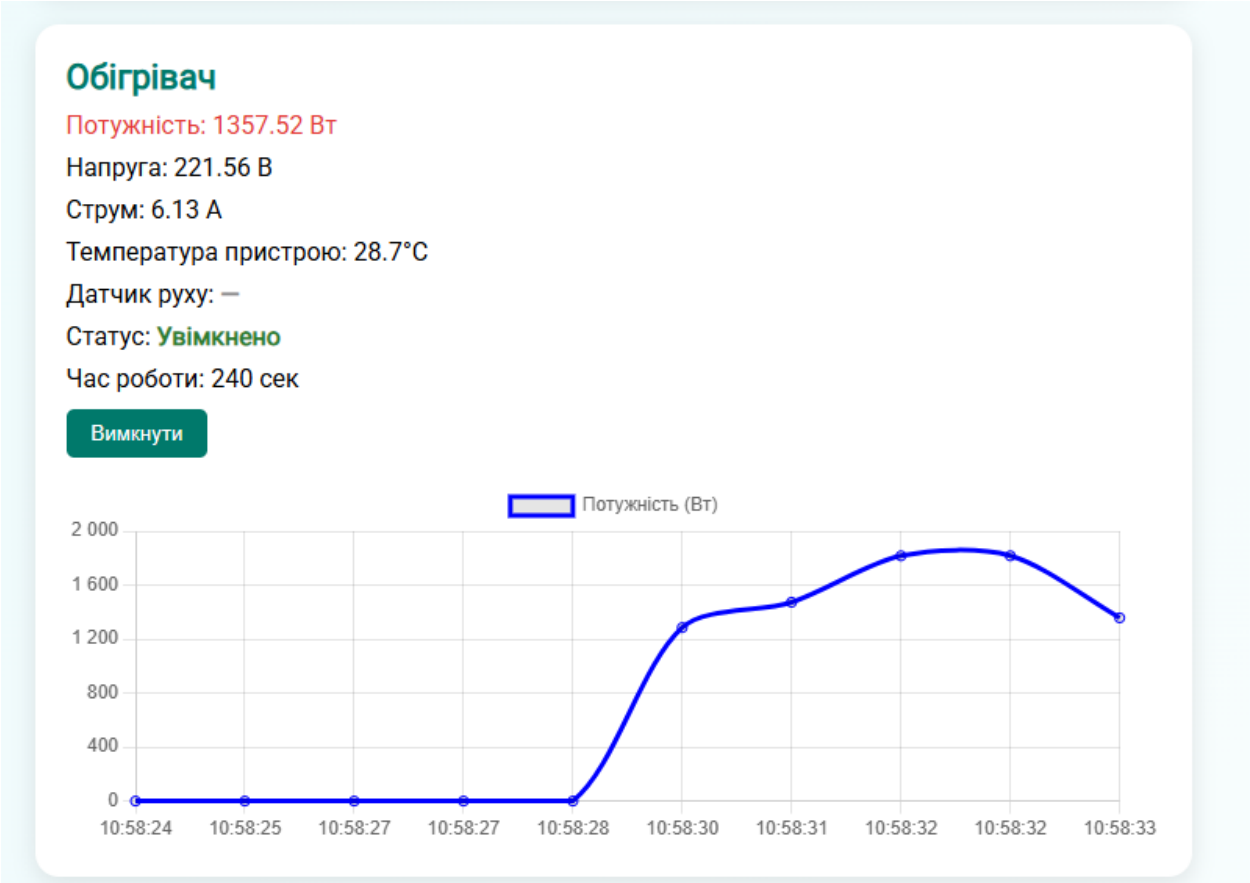

Рисунок 4.5 – Перегрів пристрою та подальше його вимкнення

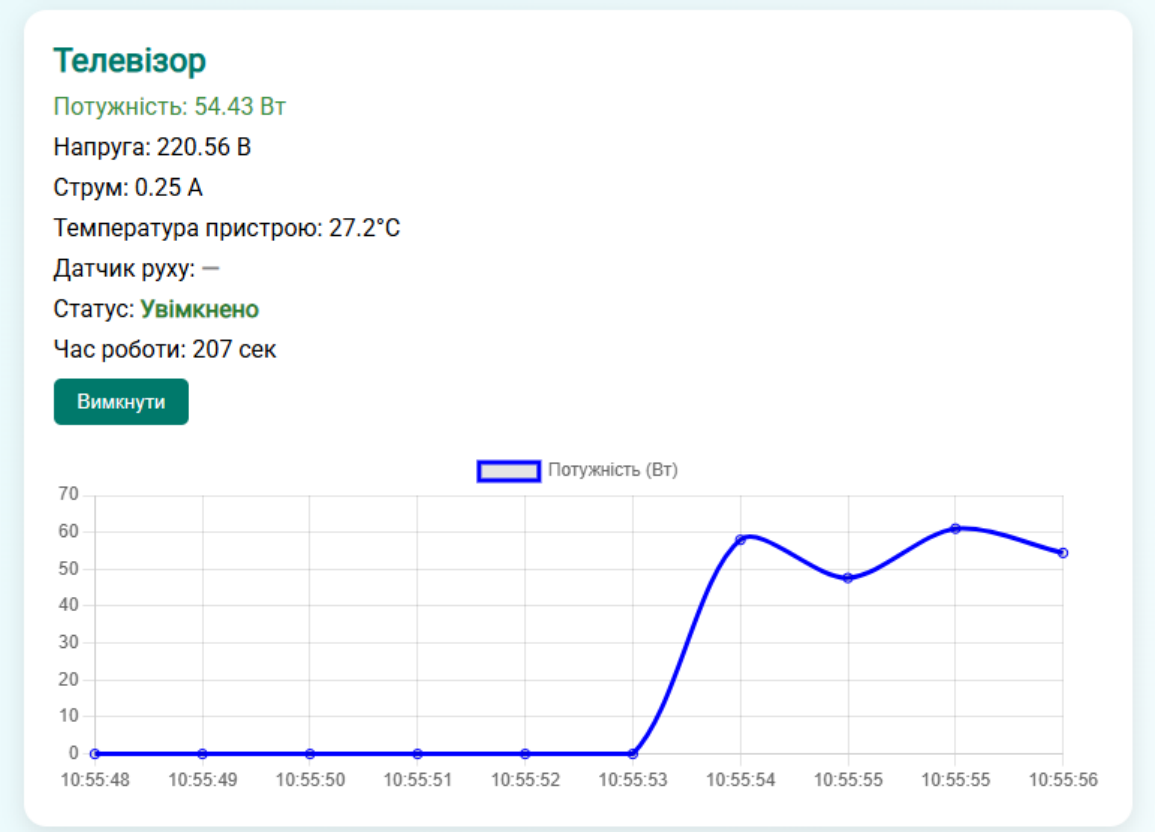

Рисунок 4.6 – Втрата сигналу приладу через несправність сенсорів

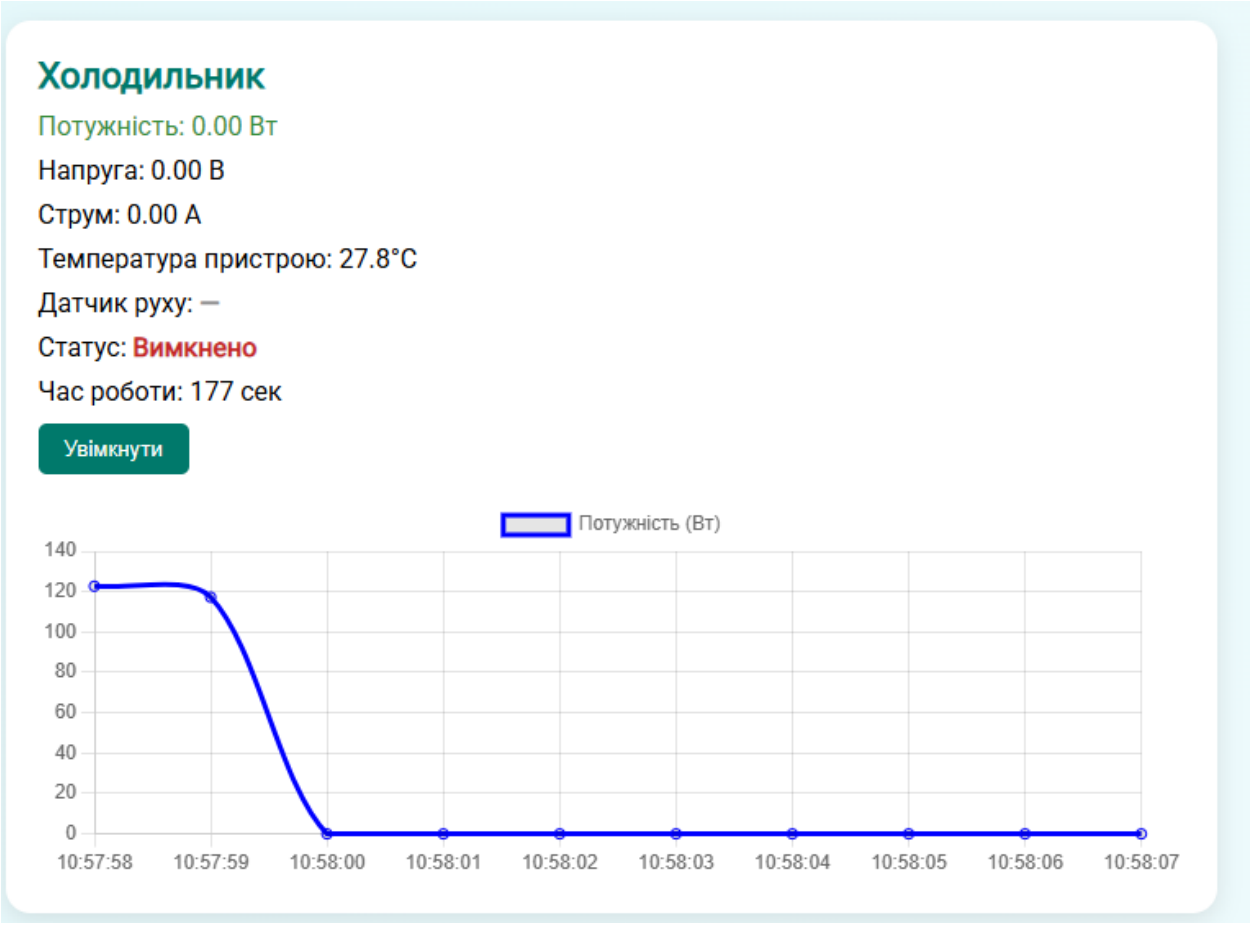

Рисунок 4.7 – Вимкнення пристрою холодильник у аварійній ситуації

Рисунок 4.8 – Перевищення допустимої потужності для побутового пристрою

Усі критичні ситуації автоматично прописуються в командному рядку та дублюються у файл events.log. У цих повідомленнях можна побачити дату та точний час аварії, пристрій, який зазнав ураження, назва аварії, яка сталася та час, на який автоматично вимикається пристрій. Після проходження цього часу побутовий пристрій автоматично повертається до роботи.

На рисунку 4.9 представлено частину повідомлень запущеної програми у командному рядку. На рисунку 4.10 файл дублювання повідомлень events.log.

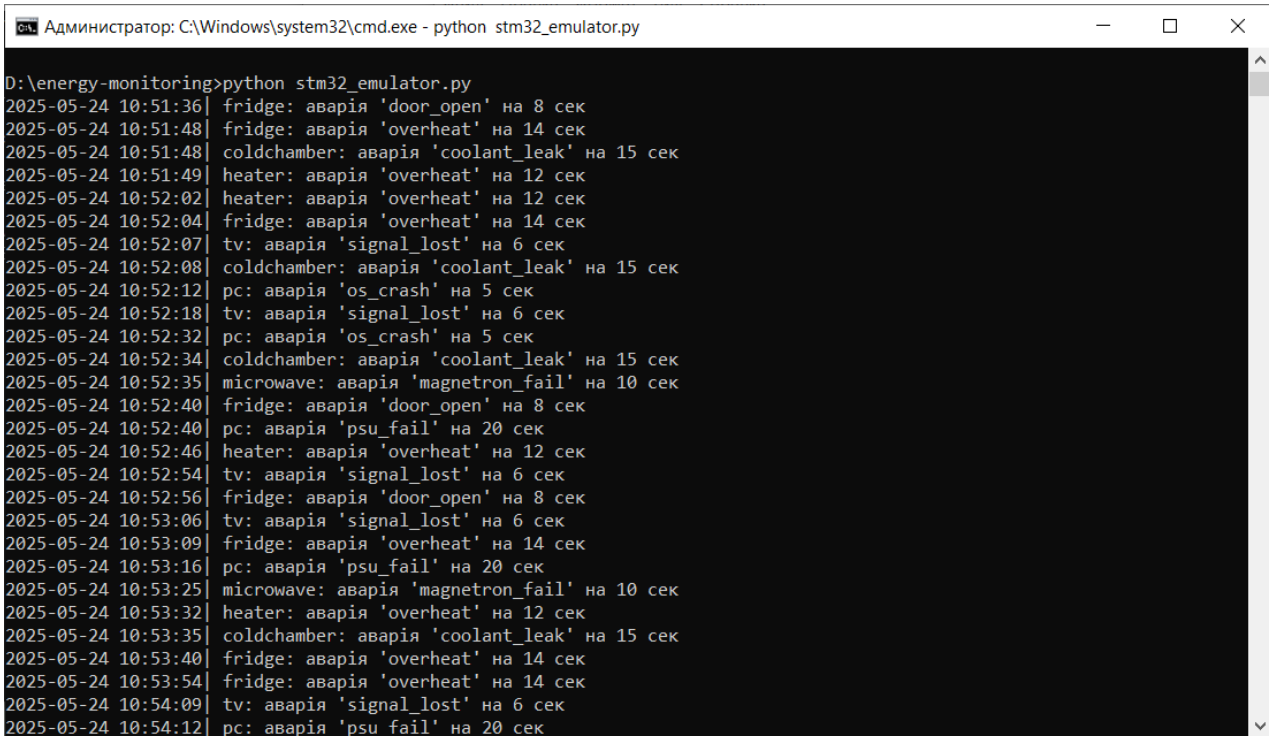

Рисунок 4.9 – Командний рядок з детальними повідомленнями про критичні та аварійні ситуації побутових пристроїв

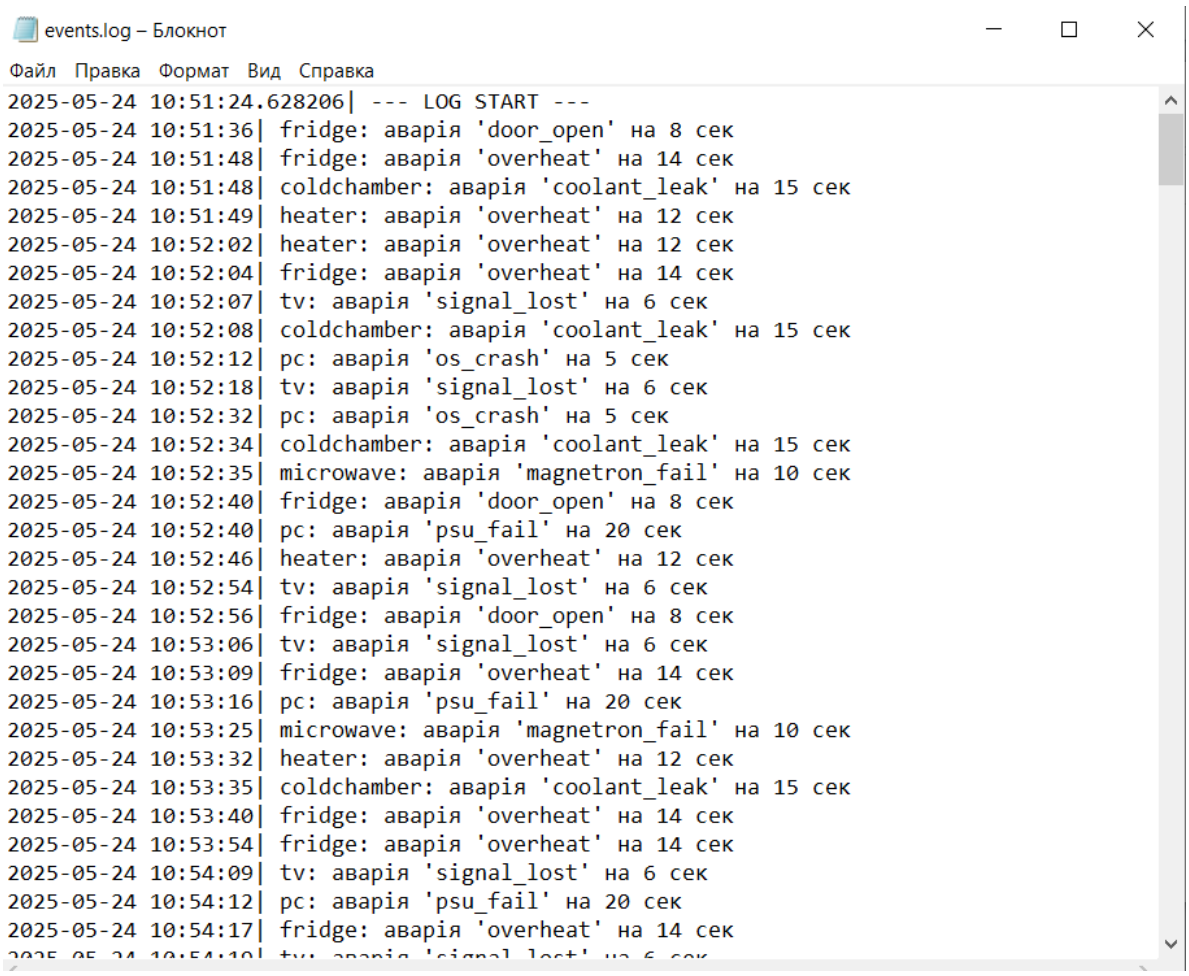

Рисунок 4.10 – Повідомлення усіх подій у файлі events.log

ВИСНОВКИ ДО ЧЕТВЕРТОГО РОЗДІЛУ

Перевірка роботи програми показала її спроможність моніторити та управляти енергоспоживанням побутових пристроїв у різних ситуаціях.

Наведено інструкцію для початкового запуску програми та показано зручний інтерфейс для користувача, як користуватися веб-сайтом. Були наведені приклади різних пристроїв та їх головні характеристики. Продемонстровані приклади різних аварійних та критичних ситуацій, а також, подальша поведінка самих пристроїв. Окрім цього, було наведено приклад виведення повідомлень подій побутових пристроїв.

Зроблена перевірка продемонструвала, що програма працює коректно, а створений веб-інтерфейс доволі простий для розуміння та зручний у використанні для користувачів.

ВИСНОВОК

У рамках цієї роботи було розроблено прототип комп'ютерного засобу на базі мікроконтролера STM32 для моніторингу та керування енергоспоживанням побутових пристроїв. Основною метою проєкту було створення системи, що дозволяє користувачу контролювати роботу електричних приладів, оцінювати рівень їх споживання в режимі реального часу, виявляти неефективне використання електроенергії та здійснювати базове керування навантаженням.
Було реалізовано архітектуру програмного комплексу, що складається з веб-інтерфейсу, серверної частини на Node.js та емулятора мікроконтролера на Python, який імітує поведінку фізичного STM32. Даний підхід сприяв створенню кросплатформеної системи з можливістю повного тестування без підключення фізичного обладнання.

Веб-інтерфейс забезпечує зручну візуалізацію даних про напругу, струм, потужність, температуру та стан пристроїв, а також, дозволяє вмикати або вимикати пристрої, перемикати режими роботи системи, переглядати графіки енергоспоживання та отримувати діагностичну інформацію. Було впроваджено зручні візуальні індикатори ефективності та модуль підрахунку сумарного споживання енергії. Збереження історії роботи системи реалізовано у вигляді JSON-файлу з можливістю аналізу попередніх значень.

Емулятор мікроконтролера виконує роль постачальника даних для серверної частини та моделює поведінку сенсорів (температури, струму, напруги, руху), а також дозволяє задавати збої, перевантаження або несправності.

Запропоноване рішення показало хорошу масштабованість і перспективи подальшого розвитку. Зокрема, його можна адаптувати для фізичних сенсорів, інтегрувати з хмарними сервісами або розширити кількість пристроїв. Побудована система має низький поріг входження для користувача, прозору логіку роботи та зручну структуру програмного коду.

Результати роботи демонструють, що поставлені завдання були виконані повною мірою, а розроблена система відповідає сучасним вимогам до енергоефективних рішень у сфері розумного дому. Обрана галузь проектування є доволі перспективною, тому слід надалі покращувати свої знання і працювати у цьому напрямку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. STM32F103C8T6 Datasheet // [Електронний ресурс]. — Режим доступу: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html — Дата доступу: травень 2025
2. Official Node.js Documentation // [Електронний ресурс]. — Режим доступу: https://nodejs.org/en/docs — Дата доступу: травень 2025
3. HTML Living Standard — WHATWG // [Електронний ресурс]. — Режим доступу: https://html.spec.whatwg.org/ — Дата доступу: травень 2025
4. CSS Reference (MDN Web Docs) // [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference — Дата доступу: травень 2025
5. Chart.js Documentation // [Електронний ресурс]. — Режим доступу: https://www.chartjs.org/docs/latest/ — Дата доступу: травень 2025
6. WebSocket vs TCP vs HTTP // [Електронний ресурс]. — Режим доступу: https://ably.com/concepts/websockets-tcp-http — Дата доступу: травень 2025
7. Python socket — Networking Programming // [Електронний ресурс]. — Режим доступу: https://docs.python.org/3/library/socket.html — Дата доступу: травень 2025