

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

"На правах рукопису"
УДК 004.4

«ДО ЗАХИСТУ ДОПУЩЕНО»

В.о. завідувача кафедри
_____ Олександр КОВАЛЬ

_____ 2024 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»
зі спеціальності 121 Інженерія програмного забезпечення
на тему: «Система прогнозування та оптимізації споживання енергоресурсів
EnMonCo»

Виконав: студент 2 курсу, групи ТВ-21мп

Нестеров Микита Андрійович

(прізвище, ім'я, по батькові)

_____ (підпис)

Науковий керівник професор, доц., д.т.н. Федорова Н.В.

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

_____ (підпис)

Рецензент

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

_____ (підпис)

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2024

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

Рівень вищої освіти другий (магістерський)

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

Олександр КОВАЛЬ

(підпис)

_____ 2024 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Нестерову Микиті Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема дисертації Система прогнозування та оптимізації споживання енергоресурсів EnMonCo

Науковий керівник Федорова Н.В., доц., д.т.н.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “06” листопада 2023 року № 5152-с

2. Строк подання студентом дисертації 10 січня 2024 року

3. Об'єкт дослідження процеси використання та оптимізації споживання енергоресурсів в різних сферах.

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за ОПП) Розробка та впровадження системи для прогнозування споживання енергії та ідентифікації можливостей її оптимізації на прикладі системи EnMonCo.

5. Перелік питань, які потрібно розробити Аналіз існуючих методів та систем прогнозування споживання енергоресурсів: дослідження та оцінка сучасних підходів та технологій у цій галузі. Збір та аналіз даних: збір історичних даних про споживання енергії, включаючи сезонні та годинні варіації, та аналіз цих даних для виявлення закономірностей та трендів. Розробка математичної моделі прогнозування: створення та калібрування математичної моделі, яка може передбачати споживання енергії з високою точністю. Інтеграція моделі з алгоритмами оптимізації: розробка алгоритмів, які використовують дані прогнозування для ідентифікації потенційних можливостей оптимізації

споживання енергії. Розробка програмного забезпечення системи: створення програмного забезпечення, яке інтегрує модель прогнозування та алгоритми оптимізації для реального часу чи планового використання. Тестування та валідація системи: проведення експериментів і тестувань для перевірки точності прогнозів та ефективності оптимізаційних стратегій

6. Орієнтований перелік ілюстративного матеріалу діаграми системи, інтерфейс програмного забезпечення.

7. Дата видачі завдання «01» листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.11.2022	виконано
2	Дослідження предметної області	01.11.2022 - 01.12.2022	виконано
3	Постановка вимог до проектування системи	01.12.2022 - 15.12.2022	виконано
4	Дослідження існуючих рішень	15.12.2022 - 01.01.2023	виконано
5	Розробка програмного продукту	06.02.2023 - 08.10.2023	виконано
6	Тестування	09.10.2023 - 22.10.2023	виконано
7	Захист програмного продукту	23.10.2023 - 27.10.2023	виконано
8	Підготовка магістерської дисертації	30.10.2023 - 20.11.2023	виконано
9	Передзахист	11.12.2023 - 15.12.2023	виконано
10	Захист	15.01.2024 - 19.01.2024	виконано

Студент

(підпис)

Нестеров М.А

(ім'я, прізвище)

Науковий керівник

(підпис)

Федорова Н.В

(ім'я, прізвище)

РЕФЕРАТ

Структура й обсяг дипломної роботи. Магістерська дисертація складається зі вступу, шести розділів, висновків та двох додатків. Робота містить посилання на 25 джерел та 10 ілюстрацій. Основна частина роботи викладена на 72 сторінках.

Актуальність. Актуальність цієї роботи обґрунтовується зростаючою потребою в ефективному використанні енергоресурсів в умовах сучасної глобалізації. Збільшення обсягів промислового виробництва, широке застосування інформаційних технологій та розширення громадської інфраструктури посилюють потребу у енергії. Паралельно із цим, проблеми зміни клімату наголошують на необхідності відповідального використання енергоресурсів. Таким чином, розробка ефективних систем прогнозування та оптимізації споживання енергії стає ключовою для зниження економічних витрат та охорони навколишнього середовища. Застосування методів прогнозування електроспоживання у системі EnMonCo дозволяє більш раціонально розподіляти енергетичні ресурси, крім того, адекватне прогнозування споживання електроенергії важливе для оптимізації виробничих процесів.

Метою магістерської роботи є створення ефективної системи, яка б дозволяла своєчасно виявляти можливості для оптимізації споживання енергоресурсів, а також робити точні прогнози щодо потреби в енергії на певний період часу.

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

- аналіз існуючих методів та систем прогнозування споживання енергоресурсів: дослідження та оцінка сучасних підходів та технологій у цій галузі;
- збір та аналіз даних: збір історичних даних про споживання енергії, включаючи сезонні та годинні варіації, та аналіз цих даних для виявлення закономірностей та трендів;

- розробка математичної моделі прогнозування: створення та калібрування математичної моделі, яка може передбачати споживання енергії з високою точністю;
- інтеграція моделі з алгоритмами оптимізації: розробка алгоритмів, які використовують дані прогнозування для ідентифікації потенційних можливостей оптимізації споживання енергії;
- розробка програмного забезпечення системи: створення програмного забезпечення, яке інтегрує модель прогнозування та алгоритми оптимізації для реального часу чи планового використання;
- тестування та валідація системи: проведення експериментів і тестувань для перевірки точності прогнозів та ефективності оптимізаційних стратегій;
- аналіз впливу системи на зменшення енергоспоживання та витрат: оцінка ефективності системи шляхом порівняння енергоспоживання та витрат до і після її впровадження;

Об’єктом дослідження є процеси використання та оптимізації споживання енергоресурсів в різних сферах.

Предметом дослідження є розробка та впровадження системи для прогнозування споживання енергії та ідентифікації можливостей її оптимізації. Це включає аналіз існуючих методів прогнозування споживання енергоресурсів, розробку математичних моделей для точного прогнозування, створення алгоритмів для виявлення оптимізаційних можливостей, а також розробку відповідного програмного забезпечення та його тестування.

Методи дослідження: сукупність теоретичних та експериментальних методів, а саме аналітичного методу, літературного огляду, методу вивчення випадків тощо.

Практичне значення одержаних результатів цієї роботи може мати широкий спектр позитивних наслідків, які стосуються економічної ефективності та екологічної стійкості, а саме: система дозволить підприємствам та організаціям ефективніше використовувати енергію, знижуючи тим самим витрати на

енергоресурси; точні прогнози споживання енергії допоможуть у плануванні потужностей та запобіганні перевантажень енергосистем.

Ключові слова: алгоритми оптимізації енергоспоживання, обробка сигналів, мікроконтролери, веб додаток, NodeJS, NestJS, IoT, C++, ESP32, розподілені системи, енергоспоживання, алгоритми прогнозування.

ABSTRACT

Structure and scope of the thesis. The master's thesis consists of an introduction, six chapters, conclusions and two appendices. The work contains references to 25 sources and 10 illustrations. The main part of the work is set out on 72 pages.

Relevance. The relevance of this work is justified by the growing need for efficient use of energy resources in the context of modern globalization. Increased industrial production, widespread use of information technology, and expansion of public infrastructure increase the need for energy. At the same time, climate change issues emphasize the need for responsible use of energy resources. Thus, the development of effective systems for forecasting and optimizing energy consumption is becoming key to reducing economic costs and protecting the environment. The use of electricity consumption forecasting methods in the EnMonCo system allows for a more rational distribution of energy resources, and adequate forecasting of electricity consumption is important for optimizing production processes.

The aim of the work is to create an effective system that would allow timely identification of opportunities for optimizing energy consumption, as well as making accurate forecasts of energy demand for a certain period of time.

To achieve this goal, the following tasks need to be implemented:

- analysis of existing methods and systems for forecasting energy consumption: research and evaluation of modern approaches and technologies in this area;
- data collection and analysis: collecting historical data on energy consumption, including seasonal and hourly variations, and analyzing this data to identify patterns and trends;
- development of a mathematical forecasting model: creation and calibration of a mathematical model that can predict energy consumption with high accuracy;
- Integration of the model with optimization algorithms: development of algorithms that use forecasting data to identify potential opportunities for optimizing energy consumption;

- system software development: creating software that integrates the forecasting model and optimization algorithms for real-time or scheduled use;
- testing and validation of the system: conducting experiments and tests to verify the accuracy of forecasts and the effectiveness of optimization strategies;
- analysis of the system's impact on reducing energy consumption and costs: evaluating the system's effectiveness by comparing energy consumption and costs before and after its implementation;

The object of the study is the processes of using and optimizing energy consumption in various fields.

The subject of the study is the development and implementation of a system for forecasting energy consumption and identifying opportunities for its optimization. This includes analyzing existing methods for forecasting energy consumption, developing mathematical models for accurate forecasting, creating algorithms for identifying optimization opportunities, as well as developing appropriate software and testing it.

Research methods: a combination of theoretical and experimental methods, namely the analytical method, literature review, case study method, etc.

The practical significance of the results of this work can have a wide range of positive consequences related to economic efficiency and environmental sustainability, namely: the system will allow enterprises and organizations to use energy more efficiently, thereby reducing energy costs; accurate forecasts of energy consumption will help in capacity planning and preventing overloads of energy systems.

Keywords: energy optimization algorithms, signal processing, microcontrollers, web application, NodeJS, NestJS, IoT, C++, ESP32, distributed systems, energy consumption, forecasting algorithms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ ..	10
ВСТУП.....	11
1 ПОСТАНОВКА ЗАВДАНЬ МАГІСТЕРСЬКОЇ РОБОТИ	13
1.1 Мета та завдання магістерської роботи.....	13
1.2 Призначення системи	14
1.3 Основний функціонал системи	14
Висновки до розділу 1	15
2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ.....	16
2.1 Загальний огляд	16
2.2 Традиційні методи прогнозування споживання	17
2.3 Інтелектуальні системи прогнозування	18
2.4 Системи оптимізації споживання	19
2.5 Інтегровані системи	20
2.6 Методології розробки IoT систем.....	21
2.7 Behavioral Driven Development.....	23
Висновки до розділу 2	25
3 ЗАСОБИ РОЗРОБКИ	26
3.1 Arduino.....	26
3.2 C++	27
3.3 MQTT.....	28
3.4 HTTPS.....	29
3.5 ESP32 як Веб-Сервер	31

3.6 Git.....	32
3.7 Docker	33
3.8 Javascript.....	34
3.9 HTML.....	35
3.10 CSS.....	36
Висновки до розділу 3	37
4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ	39
4.1 Розробка структурної схеми.....	39
4.2 Сценарії роботи системи	40
4.3 Взаємодія мікроконтролера з сервером	42
4.4 Взаємодія сенсорів у одній мережі із контролером.....	45
4.5 Прогнозування енергоспоживання за допомогою ARIMA.....	46
Висновки до розділу 4	47
5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ.....	48
5.1 Системні вимоги.....	48
5.2 Взаємодія користувача з мікроконтролером	48
5.3 Графічний інтерфейс.....	50
5.4 Графічний інтерфейс.....	53
Висновки до розділу 5	54
6 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ	56
6.1 Опис ідеї проекту	56
6.2 Технологічний аудит ідеї проекту	58
6.3 Аналіз ринкових можливостей запуску стартап-проекту	60
6.4 Розроблення ринкової стратегії проекту	67

Висновки до розділу 6	69
ВИСНОВКИ.....	70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	71
ДОДАТКИ.....	73
ДОДАТОК А Лістинг розробленої програми	73
ДОДАТОК Б Презентація роботи.....	96

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

JS — JavaScript;

CI/CD — Continuous Integration/Continuous Deployment;

IDE — Integrated Development Environment, інтегроване середовище розробки;

API — Application programming interface

SaaS — Software as a service

API — Application Programming Interface

HTTP — HyperText Transfer Protocol

IoT — Internet of Things

БД — база даних

REST — Representational state transfer

XML — Extensible Markup Language

HTML — HyperText Markup Language

JSON — JavaScript Object Notation

IDE — інтегроване середовище розробки

VCS — система контролю версій

CI — Continuous Integration, неперервна інтеграція

CD — Continuous Deployment

URL — Uniform Resource Locator

MQTT — Message Queuing Telemetry Transport

HTTPS — Hypertext Transfer Protocol Secure

UUID — Universally Unique Identifier

GSM — Global System for Mobile Communications

IoT — Internet of Things

BDD — Behavioral Driven Development

TDD — Test Driven Development

ВСТУП

В епоху глобалізації та технологічного прогресу, актуальність ефективного використання енергоресурсів стає все більш важливою. За останні роки ми спостерігаємо значне зростання промислового виробництва, широке впровадження інформаційних технологій та розширення інфраструктури громадських та комерційних об'єктів. Усе це веде до збільшення споживання енергоресурсів. Водночас, з огляду на глобальні виклики, такі як зміна клімату, суспільство стикається з необхідністю переосмислення та оптимізації використання енергії.

В цьому контексті, розробка систем прогнозування та оптимізації споживання енергоресурсів стає критично важливою. Такі системи можуть виступати як ключовий інструмент для зменшення витрат на енергоресурси у підприємствах і організаціях, а також сприяти збереженню природних ресурсів і зниженню негативного впливу на довкілля.

Магістерська робота "Система прогнозування та оптимізації споживання енергоресурсів EnMonCo" зосереджена на розробці методології та інструментів для детального вивчення та оптимізації процесів споживання енергії. Ця робота охоплює декілька ключових аспектів.

В першому розділі дослідження встановлюється мета та визначаються основні завдання дослідження.

У другому розділі проводиться глибокий аналіз існуючих систем і методів прогнозування та оптимізації енергоспоживання. Важливо розглянути як класичні підходи, так і сучасні технології, включаючи машинне навчання та штучний інтелект.

Третій розділ зосереджується на виборі та розробці засобів для створення прогнозуючих моделей. Тут розглядаються такі методи як аналіз часових рядів, регресійний аналіз, нейронні мережі та інші передові методики.

Четвертий розділ присвячений безпосередньо розробці нової системи та алгоритмів прогнозування. Тут важливо інтегрувати вивчені методики та перевірити їх ефективність на практиці.

У п'ятому розділі вивчається взаємодія користувачів із системою, включаючи інтерфейси, зручність використання та можливості кастомізації під конкретні потреби користувачів.

У шостому розділі описується розробка стартап проекту.

Ціллю роботи є створення не просто теоретичного інструменту, а практично застосовної системи, яка дозволить не лише прогнозувати потреби в енергії, а й виявляти потенційні точки для оптимізації її споживання. Результати такої роботи матимуть важливе значення для різних секторів економіки, від промисловості до громадських служб, і сприятимуть сталому розвитку.

1 ПОСТАНОВКА ЗАВДАНЬ МАГІСТЕРСЬКОЇ РОБОТИ

1.1 Мета та завдання магістерської роботи

Метою магістерської роботи є створення ефективної системи, яка б дозволяла своєчасно виявляти можливості для оптимізації споживання енергоресурсів, а також робити точні прогнози щодо потреби в енергії на певний період часу.

Для досягнення поставленої мети необхідно реалізувати наступні завдання:

- аналіз існуючих методів та систем прогнозування споживання енергоресурсів: дослідження та оцінка сучасних підходів та технологій у цій галузі;
- збір та аналіз даних: збір історичних даних про споживання енергії, включаючи сезонні та годинні варіації, та аналіз цих даних для виявлення закономірностей та трендів;
- розробка математичної моделі прогнозування: створення та калібрування математичної моделі, яка може передбачати споживання енергії з високою точністю;
- інтеграція моделі з алгоритмами оптимізації: розробка алгоритмів, які використовують дані прогнозування для ідентифікації потенційних можливостей оптимізації споживання енергії;
- розробка програмного забезпечення системи: створення програмного забезпечення, яке інтегрує модель прогнозування та алгоритми оптимізації для реального часу чи планового використання;
- тестування та валідація системи: проведення експериментів і тестувань для перевірки точності прогнозів та ефективності оптимізаційних стратегій;
- аналіз впливу системи на зменшення енергоспоживання та витрат: оцінка ефективності системи шляхом порівняння енергоспоживання та витрат до і після її впровадження;

Користувачами розробленого програмного продукту можуть бути звичайні домогосподарства або підприємства.

1.2 Призначення системи

Система "EnMonCo" призначена для прогнозування та оптимізації споживання енергоресурсів. Вона дозволяє точно визначати майбутні потреби у енергії та ідентифікувати ключові точки для зниження витрат і підвищення енергетичної ефективності. Ця система спрямована на підтримку сталого розвитку та економічно ефективного управління енергетичними ресурсами.

1.3 Основний функціонал системи

Програмний продукт представляє собою систему мікроконтролерів пов'язаних між собою та веб додаток, який має веб інтерфейс для взаємодії з користувачем. Кожен мікроконтролер має власний веб інтерфейс для його конфігурації користувачем

Застосунок зі сторони лічильників (мікроконтролерів) має містити такий функціонал:

- Реєстрація та авторизація лічильника у єдиній системі та видача йому унікального ідентифікатору.
- Можливість об'єднання пристроїв у локальній або різних мережах для підключення до головного лічильника для передачі інформації на сервер.
- Можливість головного лічильника контролювати стан під'єднаних до нього пристроїв та керувати ним.
- Передача даних про використання енергії протягом дня на головний сервер.

Застосунок зі сторони веб-додатку має містити такий функціонал:

— Відображення статистики споживання відповідного лічильника за весь час його роботи.

— Прогнозувати майбутнє споживання електроенергії спираючись на попередні дані.

Висновки до розділу 1

У даному розділі були чітко окреслені мета та завдання дослідження, які направлені на створення ефективної системи "EnMonCo" для прогнозування та оптимізації споживання енергоресурсів. Реалізація цього проекту передбачає детальний аналіз існуючих методів прогнозування, збір та аналіз великої кількості даних, розробку високоточної математичної моделі прогнозування, інтеграцію цієї моделі з алгоритмами оптимізації та розробку відповідного програмного забезпечення для ефективної реалізації системи. Також, вагомою частиною роботи є тестування та валідація системи, а також аналіз її впливу на зменшення енергоспоживання та витрат. Було окреслено основний функціонал системи, що включає в себе мережу мікроконтролерів та веб-додаток з інтуїтивно зрозумілим інтерфейсом, дозволяє користувачам не тільки стежити за своїм споживанням енергії, але й прогнозувати майбутнє споживання на основі історичних даних.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Загальний огляд

Розумні лічильники електроенергії, що також відомі під назвами "інтелектуальні" або "смарт" лічильники, становлять вирішальний компонент у сучасних системах керування енергоспоживанням. Ці пристрої надають споживачам можливість глибшого розуміння їхнього використання електроенергії, дозволяючи їм не тільки більш ефективно управляти своїми енергетичними потребами, але й сприяючи зниженню загальних енерговитрат та підвищенню енергозбереження.

На ринку існує кілька видів розумних лічильників, кожен з яких має свої унікальні особливості та функції:

- традиційні розумні лічильники: Більшість енергетичних компаній пропонують ці лічильники з вбудованими комунікаційними модулями. Ці модулі забезпечують зв'язок з серверами постачальника через різноманітні технології, включаючи GSM, Wi-Fi або Zigbee, дозволяючи користувачам відстежувати своє споживання через веб-портالي або мобільні додатки;
- лічильники зі смарт-розетками: Ці лічильники оснащені розетками або роз'ємами, які дозволяють користувачам підключати побутові пристрої та керувати ними дистанційно через мобільні додатки або голосові асистенти, такі як Amazon Alexa або Google Assistant;
- сонячні лічильники: Особливо актуальні для власників сонячних панелей, ці лічильники спеціалізуються на моніторингу виробництва сонячної енергії та її використання, надаючи можливість детально відстежувати ефективність сонячних панелей та управляти їх продуктивністю;
- енергозберігаючі рішення: Деякі лічильники включають додаткові функції енергозбереження, допомагаючи користувачам ідентифікувати

періоди пікового споживання та надаючи рекомендації щодо оптимізації використання енергії;

— розумні системи керування енергоспоживанням: Ці системи інтегрують розумні лічильники в комплексні рішення для автоматичного керування освітленням, опаленням та іншими системами в будинку чи на підприємстві, що сприяє значному підвищенню енергоефективності;

— додатки для моніторингу: Сучасні розумні лічильники часто супроводжуються мобільними додатками, які дозволяють користувачам відслідковувати своє споживання електроенергії у реальному часі, аналізувати історичні дані та отримувати персоналізовані рекомендації;

— дистанційний збір даних: Функція автоматичної передачі даних про споживання енергії на сервери постачальника спрощує процес обліку та фактурування, роблячи його більш зручним і точним;

Ринок розумних лічильників електроенергії активно розвивається, вносячи значний внесок у досягнення глобальних цілей енергоефективності та сталого розвитку. Різноманітність доступних рішень дозволяє кожному споживачу вибрати оптимальний варіант відповідно до своїх індивідуальних потреб та умов.

2.2 Традиційні методи прогнозування споживання

На початковому етапі розвитку систем прогнозування споживання електроенергії використовувалися переважно традиційні статистичні методи. Ці методи базувалися на аналізі історичних даних та патернів споживання, але часто не враховували комплексний вплив сезонних, погодних та інших зовнішніх факторів, що могли суттєво вплинути на споживання енергії.

Два ключові методи, що використовувалися в цих ранніх системах, включали:

— метод екстраполяції: Цей метод передбачає використання діапазону історичних даних для передбачення майбутніх тенденцій. Заснований на

припущенні, що минулі тенденції будуть продовжуватися в майбутньому, цей метод ефективний для короткострокових прогнозів, але може не враховувати раптові зміни або нетипові ситуації;

— регресійний аналіз: Цей метод використовує математичні рівняння для визначення залежності між споживанням енергії та одним або декількома прогнозуючими показниками. Регресійний аналіз дозволяє з'ясувати, як зміни в певних змінних (наприклад, температура, час доби, тип діяльності) впливають на споживання енергії;

Проте, з плином часу і розвитком технологій, з'явилася потреба в більш складних та точних методах прогнозування. Сучасні системи використовують більш комплексні алгоритми, які можуть враховувати широкий спектр змінних і забезпечувати більш точні та довгострокові прогнози. На відміну від ранніх методів, сучасні підходи здатні адаптуватися до змін у споживацьких звичках та враховувати непередбачувані зміни в енергоспоживанні.

2.3 Інтелектуальні системи прогнозування

Завдяки стрімкому розвитку обчислювальних технологій та прогресу у сфері штучного інтелекту, з'явилися інтелектуальні системи прогнозування, які застосовують передові техніки, такі як нейронні мережі, машинне навчання, глибоке навчання та аналіз часових рядів. Ці методи не лише забезпечують високу точність прогнозів, але й дозволяють системам адаптуватися до змін у патернах споживання, враховуючи різноманітні внутрішні та зовнішні фактори.

— нейронні мережі: Ці технології імітують роботу людського мозку, здійснюючи навчання на основі аналізу великих наборів даних. Вони ефективні для виявлення складних патернів і трендів у споживанні енергії;

— машинне навчання: Застосування алгоритмів, таких як випадковий ліс або градієнтний бустінг, дозволяє системам "вчитися" з історичних даних і

вдосконалювати свої прогнози. Такі системи постійно самовдосконалюються, отримуючи нові дані;

— глибоке навчання: Підходи глибокого навчання здатні обробляти великі обсяги даних, виявляючи при цьому надзвичайно складні закономірності, які можуть бути недоступними для традиційних методів аналізу;

— аналіз часових рядів (Time Series Analysis): Використання історичних даних для прогнозування майбутнього споживання, з урахуванням сезонних варіацій, погодних умов, свят і інших чинників, які можуть вплинути на споживання;

— моделі змішаного типу: Комбінація різних підходів і методів дозволяє створити гнучкі та адаптивні системи, здатні досягати максимальної точності у прогнозуванні;

Ці інтелектуальні системи прогнозування відкривають нові можливості для енергетичних компаній та споживачів у плануванні та оптимізації споживання енергії, що сприяє не тільки економії витрат, але й зниженню впливу на довкілля.

2.4 Системи оптимізації споживання

В рамках оптимізації споживання енергоресурсів розроблені системи, які не тільки фокусуються на прогнозуванні, але й забезпечують ефективне керування ресурсами. Особлива увага приділяється використанню даних із датчиків для моніторингу споживання в реальному часі та їхнього використання для автоматичного регулювання системних параметрів, таких як температура у приміщенні, для забезпечення максимальної енергоефективності.

Для оптимізації електроспоживання застосовуються наступні алгоритми:

— моніторинг реального часу: Системи використовують датчики для постійного відстеження поточного споживання енергії. Це дозволяє оперативно реагувати на зміни в патернах споживання та відповідно коригувати роботу систем;

- прогнозування попиту: Визначення періодів пікового та низького споживання енергії дозволяє системам адаптувати використання ресурсів, мінімізуючи витрати у часи високого попиту та використовуючи переваги низького попиту;
- адаптивний контроль: Автоматичне регулювання системних параметрів на основі поточних умов, наприклад, зміна режимів опалення чи охолодження відповідно до зовнішніх температур;
- розклад завантаження: Інтелігентне перенесення енергоємних завдань на періоди з нижчим тарифом на електроенергію або з низьким рівнем споживання, що забезпечує зниження вартості електроенергії;

Ці підходи сприяють не тільки зниженню вартості електроенергії для кінцевих споживачів, але й зменшенню навантаження на електромережі, підвищуючи при цьому загальну ефективність енергосистем. Оптимізація використання електроенергії також відіграє важливу роль у зменшенні впливу на довкілля, оскільки зниження споживання енергії веде до відповідного зниження викидів парникових газів.

2.5 Інтегровані системи

Сучасні інтегровані системи управління енергоресурсами ефективно поєднують елементи прогнозування та оптимізації, використовуючи комплексний підхід для забезпечення максимальної ефективності управління. Вони втілюють собою синтез різноманітних технологій та методів, спрямованих на адаптивне реагування на зміни умов та потреб, а також на інтеграцію різних джерел даних для точнішого аналізу та прогнозування.

- динамічна адаптація: Сучасні системи здатні оперативно реагувати на зміни в умовах оточуючого середовища, варіаціях попиту на енергію, а також на внутрішні зміни в споживанні енергії. Це досягається шляхом

застосування алгоритмів, які автоматично коригують роботу систем, змінюючи параметри на основі отриманих даних;

— об'єднання джерел даних: Інтеграція інформації з різних джерел, таких як датчики споживання енергії, метеорологічні дані, інформація про роботу обладнання та інші, дозволяє створити більш повне та точне уявлення про поточний стан системи. Це сприяє кращому розумінню потреб у енергії та можливостям для її економії;

— комплексне моделювання: Системи використовують передові математичні та комп'ютерні моделі для відтворення реальних процесів споживання та виробництва енергії. Це дозволяє точно прогнозувати майбутні сценарії та оптимізувати роботу системи відповідно до прогнозованих умов;

— автоматизоване вирішення конфліктів: Системи здатні автоматично розпізнавати та вирішувати конфліктні ситуації, наприклад, коли існує протиріччя між потребами в енергії різних відділів або пристроїв;

Застосування таких інтегрованих систем веде не тільки до оптимізації використання енергії та зниження витрат, але й до підвищення загальної ефективності, надійності та стійкості енергосистем. Вони стають незамінним інструментом в управлінні енергоресурсами на різних рівнях - від окремих будівель до цілих промислових комплексів.

2.6 Методології розробки IoT систем

Сучасні методології розробки систем Internet of Things (IoT) та систем прогнозування спираються на комплексний підхід, що поєднує передові технології, алгоритми та принципи проектування. Ці методології забезпечують розробку надійних, масштабованих та ефективних систем, здатних збирати, аналізувати та використовувати великі обсяги даних для прийняття обґрунтованих рішень.

- модульний підхід у розробці: Це передбачає створення системи з взаємозамінних модулів, що спрощує масштабування та оновлення системи. Кожен модуль відповідає за свою частину функціоналу, що дозволяє легко інтегрувати нові технології та оновлення;
- інтеграція з хмарними технологіями: Використання хмарних платформ для зберігання та обробки даних дозволяє знизити витрати на інфраструктуру та підвищити масштабованість та доступність систем. Хмарні рішення також забезпечують високу швидкість обробки даних та гнучкість в управлінні ресурсами;
- застосування машинного навчання та штучного інтелекту: Алгоритми машинного навчання та ШІ використовуються для аналізу даних, отриманих з IoT-пристроїв. Це дозволяє не тільки прогнозувати потреби в ресурсах, але й автоматизувати процеси прийняття рішень, заснованих на аналізі великих обсягів даних;
- крос-платформна сумісність та стандартизація: Розробка IoT-систем з урахуванням сумісності з різними платформами та пристроями є ключовим аспектом для забезпечення їх широкого використання. Стандартизація протоколів комунікації та форматів даних сприяє інтеграції різних систем та пристроїв;
- безпека та конфіденційність даних: Оскільки IoT-системи збирають та обробляють великі обсяги даних, важливо забезпечити високий рівень безпеки та захисту даних. Це включає застосування шифрування, безпечних протоколів передачі даних та політик конфіденційності;
- інтерактивність та користувацький інтерфейс: Розробка інтуїтивно зрозумілих і зручних інтерфейсів для кінцевих користувачів дозволяє легко взаємодіяти з системою, контролювати її роботу та аналізувати отримані дані;

Застосування цих методологій в розробці IoT-систем та систем прогнозування дозволяє створювати ефективні, надійні та масштабовані рішення, які можуть адаптуватися до змінюваних потреб користувачів та умов ринку.

2.7 Behavioral Driven Development

Behavioral Driven Development (BDD) є методологією розробки програмного забезпечення, яка фокусується на співпраці між розробниками, тестувальниками та не-технічними учасниками проекту (наприклад, бізнес-аналітиками). Основна ідея BDD полягає у визначенні поведінки програми через сценарії, написані зрозумілою для всіх учасників проекту мовою.

Коли краще використовувати BDD:

- у комплексних проектах, де важлива чітка комунікація між технічною командою та бізнес-стейкхолдерами;
- в проектах, де важливі високий рівень зрозумілості вимог та їх точна реалізація;
- коли потрібно забезпечити, що функціональність відповідає бізнес-потреbam і є корисною для кінцевого користувача;

Плюси BDD:

- поліпшена комунікація: BDD сприяє зрозумілості та ясності вимог серед усіх членів команди;
- фокус на користувача: Врахування потреб кінцевого користувача під час формулювання вимог;
- зменшення непорозумінь: Зменшує ризик неправильного трактування вимог;
- документація: Сценарії BDD можуть слугувати як жива документація системи;

Недоліки BDD:

- вимога часу та ресурсів: Розробка сценаріїв BDD може бути часомісткою;

- потреба у залученні всіх стейкхолдерів: Іноді може бути складно залучити всіх необхідних учасників;

- складність: Для нетехнічних учасників може бути складно розуміти технічні аспекти BDD;

Порівняння із Test Driven Development (TDD):

- фокус: TDD фокусується на розробці тестів до написання коду, що гарантує, що програма відповідає технічним вимогам. BDD, в свою чергу, орієнтований на розробку на основі поведінкових вимог, з акцентом на функціональність з точки зору кінцевого користувача;

- спілкування: BDD сприяє кращому розумінню та комунікації між розробниками, тестувальниками та бізнес-стейкхолдерами. TDD більше фокусується на комунікації в межах технічної команди;

- тестування: У TDD тести в основному є одиничними та орієнтовані на технічні аспекти, тоді як у BDD тести засновані на поведінкових сценаріях, які часто включають більш комплексні інтеграційні перевірки;

- документація: BDD надає більш зрозумілу та доступну для бізнес-стейкхолдерів документацію, у той час як TDD зосереджений на технічній документації та тестовому покритті;

Виходячи із вищезазначеного було обрано модель розробку BDD оскільки вона більше підходить під вимоги проекту через фокус на поведінкових аспектах системи оскільки вона розрахована на кінцевого користувача, а також неможливості або надмірної проблематичності використання TDD через складність системи, яка включає різні компоненти, такі як IoT пристрої, датчики та алгоритми прогнозування, BDD дозволить ефективніше інтегрувати та тестувати всі ці елементи.

Висновки до розділу 2

У розділі "Аналіз існуючих рішень" розглянуто широкий спектр технологій та методів, які застосовуються в сучасних системах управління споживанням енергії. Від традиційних розумних лічильників, що дозволяють споживачам моніторити та оптимізувати використання енергії, до сучасних інтелектуальних систем прогнозування, які використовують нейронні мережі, машинне навчання та аналіз часових рядів для точнішого та адаптивного управління енергоресурсами. Акцент зроблено на необхідності інтеграції різних підходів і технологій для створення комплексних, гнучких та ефективних систем. Така інтеграція включає динамічну адаптацію до змін умов, об'єднання джерел даних, розширене моделювання та автоматизацію процесів. У підсумку, інтеграція різноманітних технологій та методів, а також застосування відповідної методології розробки, є ключовими для створення ефективних систем управління енергоресурсами, здатних відповідати викликам сучасності та сприяти сталому розвитку.

3 ЗАСОБИ РОЗРОБКИ

3.1 Arduino

Arduino - це платформа для розробки апаратного і програмного забезпечення, яка спрощує створення і прототипування різноманітних електронних пристроїв і систем. Основними компонентами Arduino є мікроконтролери, середовище програмування та бібліотеки.

Arduino використовує різні мікроконтролери, такі як ATmega328 для Arduino Uno, ESP8266 для Arduino Wi-Fi та багато інших. Ці мікроконтролери мають різні характеристики та функціональність, що дозволяє розробникам вибирати платформу, яка найкраще підходить для їх конкретного проекту.

Arduino постачається з багатьма стандартними бібліотеками, які спрощують роботу з певними сенсорами, модулями та іншими пристроями. Також має велику та активну спільноту розробників. Це означає, що можна знайти безліч документації, прикладів коду та порад на форумах, що допоможуть вирішувати проблеми та здійснювати вдосконалення у проектах. Це дозволяє розробникам використовувати готовий код для взаємодії з різними пристроями, що економить час і зусилля. Arduino сумісний з безліччю датчиків, модулів та розширень, які дозволяють додавати різні функції до вашого проекту, такі як датчики температури, вологості, GPS, Bluetooth, Wi-Fi та інші.

Arduino можна легко інтегрувати з іншими електронними компонентами та платформами, що робить його ідеальним вибором для розробки інтерактивних систем та пристроїв.

Arduino - це дуже популярний і доступний засіб для розробки програмного забезпечення для різних проектів, від простих світлодіодних вправ до складних систем автоматизації та IoT пристроїв.

3.2 C++

C++ - це потужна мова програмування, яка використовується для розробки різноманітних програмних застосунків і систем. Вона дозволяє розробникам створювати програмне забезпечення для різних цілей, починаючи від простих консольних програм і закінчуючи складними додатками для великих проєктів.

C++ підтримує об'єктно-орієнтований підхід, що дозволяє структурувати код у вигляді об'єктів та класів, що полегшує розробку і обслуговування програм. Вона також включає в себе інші парадигми програмування, такі як процедурне програмування і шаблони, що розширює її можливості.

C++ має великий екосистему стандартних бібліотек, які включають різноманітні функції та інструменти для взаємодії з різними аспектами програмної розробки. Ці бібліотеки полегшують роботу з рядками, контейнерами, мережами, файлами та багатьма іншими аспектами програмного забезпечення.

C++ також відома своєю продуктивністю і можливістю оптимізації коду для ефективного використання ресурсів комп'ютера. Вона є кросплатформеною мовою, тобто код, написаний на C++, може бути компільований і виконаний на різних операційних системах і архітектурах.

Завдяки активній спільноті розробників, доступності ресурсів та документації, C++ залишається однією з найпопулярніших мов програмування для великих і складних проєктів. Ця мова дозволяє розробникам втілювати свої ідеї у програмному забезпеченні, надаючи при цьому високий рівень контролю і продуктивності.

Зокрема, C++ знайшла широке застосування в наступних областях:

— розробка ігор: Це одна з основних областей використання C++. Багато відомих відеоігор, включаючи AAA-проєкти, були створені з використанням C++ через його високу продуктивність і можливості оптимізації;

- системне програмування: C++ дозволяє розробляти операційні системи, драйвери, системи керування ресурсами та інші системні програми, оскільки вона надає контроль над ресурсами комп'ютера;
- вбудовані системи: Вбудовані системи, такі як мікроконтролери і пристрої IoT, часто програмуються на C++ через його можливості мінімізувати використання ресурсів і контролювати апаратне забезпечення;
- наукові обчислення: Велика частина наукового програмного забезпечення, включаючи математичні обчислення, обробку сигналів та інші області, реалізується на C++ через його високу швидкодію;
- фінанси та фінансовий аналіз: У фінансовому секторі C++ використовується для розробки торгових систем, аналізу ринку, обробки даних та великих фінансових додатків;
- десктопні застосунки: C++ дозволяє створювати десктопні програми для різних платформ, включаючи Windows, macOS і Linux;
- ігри в реальному часі (Real-time games): Від створення 3D-графіки до обробки великої кількості даних у реальному часі, C++ залишається популярним вибором для розробки ігор в реальному часі;

Загалом, C++ - це мова програмування, яка надає великий контроль і широкі можливості для розробників, що робить її важливим інструментом для створення різноманітних програмних застосунків і систем у багатьох галузях.

3.3 MQTT

Протокол MQTT (Message Queuing Telemetry Transport) є ключовим інструментом у світі IoT (Internet of Things), надаючи засоби для ефективної комунікації між пристроями та додатками через мережу. Створений у 1999 році в IBM, MQTT тепер є відкритим стандартом, який отримав широку підтримку серед розробників і компаній.

Протокол MQTT розроблено з урахуванням обмежених ресурсів, що робить його ідеальним для пристроїв IoT, які часто характеризуються обмеженим обчислювальним потенціалом та пам'яттю. Однією з ключових особливостей MQTT є його модель "опублікувати/підписатися", що дозволяє пристроям публікувати повідомлення на задані теми, а інші пристрої можуть підписуватися на ці теми для отримання відповідних повідомлень.

MQTT також підтримує різні рівні якості обслуговування (QoS), що дає можливість користувачам вибирати баланс між швидкістю доставки та надійністю. Повідомлення в MQTT організовані за темами, дозволяючи легко структурувати та керувати потоками даних в системі. Для випадків, коли з'єднання з пристроєм переривається, MQTT пропонує механізм "останнього зберігача", що забезпечує автоматичну відправку повідомлень у таких ситуаціях.

Центральним елементом в MQTT є брокер, який відповідає за розподіл повідомлень між пристроями. Забезпечення безпеки даних в MQTT здійснюється за допомогою шифрування та аутентифікації, що критично важливо для IoT-застосунків.

Завдяки своїй легкості, гнучкості та високій адаптивності, MQTT став популярним вибором для розробників IoT-пристроїв, знаходячи застосування в різних сферах від моніторингу та управління пристроями до складних автоматизованих систем.

3.4 HTTPS

HTTPS (Hypertext Transfer Protocol Secure) являє собою протокол передачі гіпертексту, який забезпечує захищене з'єднання у Всесвітній павутині. Розроблений на базі стандартного HTTP з додаванням шару безпеки за допомогою SSL/TLS (Secure Sockets Layer / Transport Layer Security), цей протокол відіграє ключову роль у забезпеченні конфіденційності, цілісності та аутентифікації даних в Інтернеті.

HTTPS був розроблений в середині 1990-х років з метою забезпечення безпеки транзакцій в Інтернеті, особливо в контекстах, де безпека передачі даних є критичною, наприклад, в онлайн банківництві та електронній комерції. З того часу його використання значно розширилося, переходячи в стандарт де-факто для більшості веб-сайтів.

HTTPS використовує асиметричне шифрування для обміну секретним симетричним ключем між клієнтом і сервером. Це забезпечує конфіденційність і цілісність даних, що передаються між веб-браузером і сервером. Застосування сертифікатів SSL/TLS, які видаються сертифікаційними органами (CA), допомагає у встановленні довіри до ідентичності веб-сайтів, запобігаючи атакам типу "людина посередині".

У сучасному цифровому світі, де питання кібербезпеки і захисту особистих даних набувають все більшого значення, HTTPS є фундаментальним компонентом для забезпечення безпечного обміну даними. Він не тільки шифрує дані, але й забезпечує їх цілісність та захищає від численних кібератак.

Зі зростанням потреби в безпеці даних, виникають нові виклики та питання щодо управління сертифікатами, продуктивності серверів та сумісності протоколів. Постійна еволюція криптографічних методів і стандартів TLS сприяє підвищенню безпеки, однак вимагає від фахівців у сфері ІТ постійного оновлення знань та навичок. Окрім того, зростаюча увага до особистої конфіденційності в Інтернеті ставить нові вимоги до шифрування та анонімності в мережі.

HTTPS є ключовою технологією в сучасному Інтернеті, що забезпечує безпеку веб-трафіку. Незважаючи на свою важливість, він постійно розвивається і адаптується до нових викликів у галузі кібербезпеки, вимагаючи постійного удосконалення та адаптації з боку веб-розробників та адміністраторів систем.

3.5 ESP32 як Веб-Сервер

ESP32, інтегрований мікроконтролер з функціоналом Wi-Fi та Bluetooth, розроблений компанією Espressif Systems, зарекомендував себе як значущий елемент в сфері Інтернету речей (IoT). Використання даного мікроконтролера в ролі веб-сервера відкриває нові горизонти для розробників, дозволяючи організовувати взаємодію між ESP32 та іншими пристроями через веб-інтерфейс, базуючись на HTTP протоколах.

Інтегрований модуль Wi-Fi дозволяє ESP32 функціонувати у якості точки доступу (AP) або підключатися до існуючих мереж у режимі станції (STA). Програмування зазвичай ведеться через Arduino IDE або ESP-IDF, використовуючи мови C++ або Python.

Реалізація Веб-Сервера на ESP32:

- налаштування Мережі: Конфігурація ESP32 для роботи як AP або STA є необхідною для інтеграції у мережу;
- розвиток HTTP Сервера: Кодування для запуску HTTP сервера на ESP32, який оброблятиме запити та надаватиме відповіді;
- розробка Веб-Інтерфейсу: Створення веб-сторінок з використанням HTML/CSS для інтерактивного користувацького інтерфейсу, доступного через веб-браузер.
- логіка Управління та Взаємодії: Програмна логіка для координації між веб-інтерфейсом та апаратними ресурсами ESP32, такими як датчики та світлодіоди;

Застосування: Керування домашніми системами, такими як освітлення, клімат-контроль, тощо. Моніторинг показників з різних сенсорів та їх відображення у веб-інтерфейсі. Інтеграція в Інтернет Речей (IoT) Використання в якості вузла для зв'язку та контролю у широкомаштабних IoT-системах.

Незважаючи на високу функціональність, ESP32 має обмеження за обсягом пам'яті та обчислювальною потужністю. Також можливі випадки нестабільності Wi-Fi з'єднання, що впливає на надійність системи.

Як висновок можна зазначити що використання ESP32 в якості веб-сервера є потужним інструментом для розробників, який сприяє ефективній інтеграції апаратного забезпечення у складні мережеві структури, зокрема в сферах IoT та автоматизації. Проте, необхідно враховувати обмеження апаратних ресурсів, питання безпеки та стабільності з'єднання для оптимізації проектів на основі ESP32.

3.6 Git

Git є відкритим програмним забезпеченням для контролю версій, розробленим Лінусом Торвальдсом, основним творцем ядра Linux. Це інструмент, який використовується програмістами для керування змінами в коді, особливо в контексті багатокористувацьких та великих проектів. Git дозволяє кільком користувачам працювати над спільними проектами одночасно, не перешкоджаючи один одному і дає можливість швидко переключатися між різними версіями проектів.

Однією з ключових особливостей Git є його система розподілених контрольних версій, яка означає, що кожен користувач працює з копією кодової бази, включаючи повну історію змін. Це підвищує надійність, оскільки немає єдиного місця збою, і дозволяє користувачам працювати офлайн.

В основі роботи з Git лежать коміти - фіксації змін, які робить користувач у коді. Коміти можуть бути організовані в гілки, що дозволяє розділяти розробку нових функцій від основної роботи коду. Гілки можуть бути згодом злиті, що забезпечує інтеграцію нового коду з основною кодовою базою.

Git також має потужні засоби для виявлення та вирішення конфліктів між змінами, внесеними різними користувачами. Це забезпечує ефективне управління кодовою базою і допомагає запобігати помилкам у процесі розробки.

Інтеграція з веб-сервісами, такими як GitHub та GitLab, розширює можливості Git, надаючи графічний інтерфейс, інструменти для відстеження проблем, перегляду коду та інші функції, які покращують співпрацю та управління проектами.

В цілому, Git є ключовим інструментом в сучасній розробці програмного забезпечення, який значно спрощує процес управління змінами та співпраці між розробниками.

3.7 Docker

Docker – це платформа для розробки, доставки та запуску програмного забезпечення у вигляді легких та переносних контейнерів. Він дозволяє пакувати додаток із всіма його залежностями в стандартний блок, що забезпечує консистентність середовища на всіх етапах розробки, тестування та розгортання.

Ключовою перевагою Docker є те, що він ізолює додаток від інфраструктури. Це означає, що розробники можуть зосередитися на написанні коду, не турбуючись про те, чи правильно він працюватиме на іншій машині або в іншому середовищі, оскільки контейнер містить усе необхідне для запуску програми.

Docker використовує концепцію образів (images), які представляють собою шаблони, і контейнерів, що є екземплярами цих образів у виконанні. Образи створюються за допомогою Dockerfile, текстового файлу, який містить інструкції щодо того, як зібрати образ. Коли Docker запускає образ, він створює контейнер, який виконує процеси в ізольованому середовищі, забезпечуючи таким чином ізоляцію та безпеку.

Docker також включає систему управління версіями для образів, що дозволяє зберігати, версіонувати та перерозподіляти контейнери, навіть у

великомасштабних системах. Це сприяє спрощенню процесу неперервної інтеграції та неперервного розгортання (CI/CD).

Однією з важливих характеристик Docker є його портативність. Контейнери Docker можна легко переміщати між локальними середовищами розробки, системами неперервної інтеграції, тестовими та виробничими середовищами. Це значно спрощує процес управління інфраструктурою та розгортання додатків.

Завдяки цим особливостям, Docker став ключовим інструментом у сучасному світі розробки програмного забезпечення, дозволяючи розробникам та системним адміністраторам ефективно вирішувати проблеми ізоляції додатків, залежностей та розгортання, забезпечуючи більш швидке впровадження та високу надійність систем.

3.8 Javascript

JavaScript є високорівневою, часто компілюємою та багатопарадигмальною мовою програмування, що відрізняється динамічною типізацією, об'єктно-орієнтованою та прототипною структурою. Вперше з'явившись у вигляді мови для створення інтерактивних веб-сторінок у браузерях, JavaScript швидко розвинувся і тепер використовується не тільки для клієнтської, але й серверної розробки завдяки середовищам, таким як Node.js.

Ключовими особливостями JavaScript є його гнучкість і здатність до інтеграції з іншими технологіями. Ця мова легко взаємодіє з HTML і CSS для створення динамічних веб-сторінок і застосунків. Завдяки цьому, JavaScript є неодмінною складовою більшості веб-проектів, і це одна з причин, чому мова відіграє ключову роль у веб-розробці.

Інша важлива риса JavaScript - асинхронність, особливо у вигляді обіцянок (Promises) та асинхронних функцій (async/await). Це дозволяє ефективно виконувати відкладені або тривалі часом операції, такі як запити до сервера або

будь-які важкі обчислення, не блокуючи головного потоку виконання та інтерфейсу користувача.

JavaScript також характеризується великим співтовариством і широким спектром доступних бібліотек і фреймворків, які спрощують розробку складних додатків із багатими функціональними можливостями. Від відомих бібліотек для створення користувацьких інтерфейсів, таких як React, до потужних інструментів для серверної розробки - усе це робить JavaScript надзвичайно гнучкою мовою для розробників різного рівня.

З огляду на всі ці аспекти, JavaScript є однією з найпопулярніших мов програмування в світі. Він постійно розвивається і адаптується, щоб задовольнити сучасні вимоги розробки, забезпечуючи потужні і гнучкі засоби для створення широкого спектру програм - від простих веб-сторінок до складних корпоративних рішень.

3.9 HTML

HTML (HyperText Markup Language) є стандартом у світі веб-розробки, використовуваним для структурування вмісту на веб-сторінках та веб-додатках. Він дозволяє розмістити текст, зображення та інші елементи, використовуючи теги для визначення різних частин веб-документа, таких як заголовки, параграфи, і списки. HTML забезпечує функціональність гіпертекстових посилань, що дозволяє легко з'єднувати веб-сторінки одна з одною.

Важливим аспектом HTML є його співпраця з іншими технологіями, як-от CSS для стилізації та JavaScript для додавання інтерактивності. Це робить HTML вкрай гнучким і потужним інструментом в руках веб-розробників. Він також підтримує створення форм для збору даних від користувачів, що є важливою функцією для багатьох веб-додатків.

Однією з особливостей HTML є його семантична розмітка, яка включає теги, що описують зміст сторінки, забезпечуючи краще розуміння і доступність

контенту. Такі елементи як заголовки, статті та розділи допомагають у структуруванні інформації та її інтерпретації.

HTML також адаптується до різних пристроїв та браузерів, забезпечуючи адаптивність та доступність. Це означає, що веб-сторінки, створені за допомогою HTML, можуть ефективно відображатися на різних пристроях, від мобільних телефонів до настільних комп'ютерів. До того ж, HTML підтримує вбудовування мультимедійних елементів, таких як відео та аудіо, без необхідності використання додаткових плагінів.

Завдяки своїй ефективності, простоті використання та здатності працювати на різних платформах і операційних системах, HTML є незамінним інструментом для розробників, які створюють інтерактивні та динамічні веб-сайти та додатки.

3.10 CSS

CSS (Cascading Style Sheets) є фундаментальним інструментом у веб-розробці, який використовується для стилізації веб-сторінок. Він дозволяє контролювати візуальний вигляд HTML-документів, визначаючи, як елементи будуть відображатися в браузері. CSS відокремлює зміст від його візуального представлення, що дозволяє легко змінювати дизайн без впливу на саму структуру сторінки.

Однією з ключових особливостей CSS є його здатність створювати адаптивні та реактивні веб-дизайни. Це означає, що веб-сторінки можуть автоматично адаптуватися до різних розмірів екранів та пристроїв, включаючи мобільні телефони, планшети та настільні комп'ютери. Це досягається за допомогою медіа-запитів та гнучких макетів.

CSS також сприяє поліпшенню доступності веб-сторінок, дозволяючи розробникам керувати такими аспектами, як розміри шрифтів, контрастність кольорів та інші фактори, які можуть впливати на зручність використання сайту людьми з різними обмеженнями.

Використання CSS дає можливість розробникам ефективно управляти стилями для різних елементів веб-сторінки, забезпечуючи узгодженість дизайну через весь сайт. Оскільки стилі визначаються окремо від HTML-коду, вони можуть бути легко змінені або оновлені, що робить процес розробки більш гнучким та ефективним.

CSS також підтримує велику кількість стилістичних ефектів, таких як тіні, градієнти, переходи, анімації та інші, що дозволяє створювати більш динамічний та привабливий дизайн веб-сторінок.

Загалом, CSS є невід'ємною частиною сучасної веб-розробки, яка забезпечує естетичне оформлення та функціональність веб-сайтів. Він дозволяє розробникам створювати візуально привабливі, адаптивні та доступні веб-сторінки, що сприяють забезпеченню кращого користувацького досвіду.

Висновки до розділу 3

У третьому розділі було детально розглянуто різноманітні інструменти та технології, які застосовуються в розробці програмного забезпечення для прогнозування та оптимізації споживання. Використання платформи Arduino та мови програмування C++ виявилось фундаментальним для розробки низькорівневих компонентів системи, забезпечуючи високу продуктивність та надійність.

Протокол MQTT і технологія HTTPS відіграють ключову роль у забезпеченні безпечного та ефективного обміну даними між мікроконтролерами та сервером. Використання ESP32 як веб-сервера демонструє інноваційний підхід до створення мережових з'єднань і взаємодії компонентів системи, підвищуючи її гнучкість та масштабованість.

Внесок систем контролю версій, зокрема Git, та контейнеризації, представленої Docker, є важливим для забезпечення стабільності розробки та легкості впровадження змін. Javascript, з іншого боку, відіграє значну роль у розробці клієнтської частини, забезпечуючи інтерактивний та користувацьки привабливий інтерфейс.

Цей розділ наголошує на важливості вибору правильних інструментів та технологій у розробці складних систем. Описані в цьому розділі інструменти та методології лягають в основу сучасної практики розробки програмного забезпечення та відіграють критичну роль у досягненні цілей проект.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Розробка структурної схеми

Підготовка до процесу розробки включає важливий крок створення детальної структурної схеми, яка є ключовою для оцінки складності та планування різних фаз розробки проекту. У проекті розумного лічильника EnMonCo існують три головні аспекти роботи: інтеракція з користувачем, комунікація з сервером та процес збору та обробки даних мікроконтролером від лічильника електроенергії.

З метою глибшого розуміння та візуалізації взаємодій між складовими системи, було створено DFD-діаграму (діаграму потоку даних), яка ілюструє взаємозв'язки між елементами системи через обмін даними. Діаграма, зображена на рисунку 4.1, надає зрозуміле зображення взаємодії між користувачем та мікроконтролером.

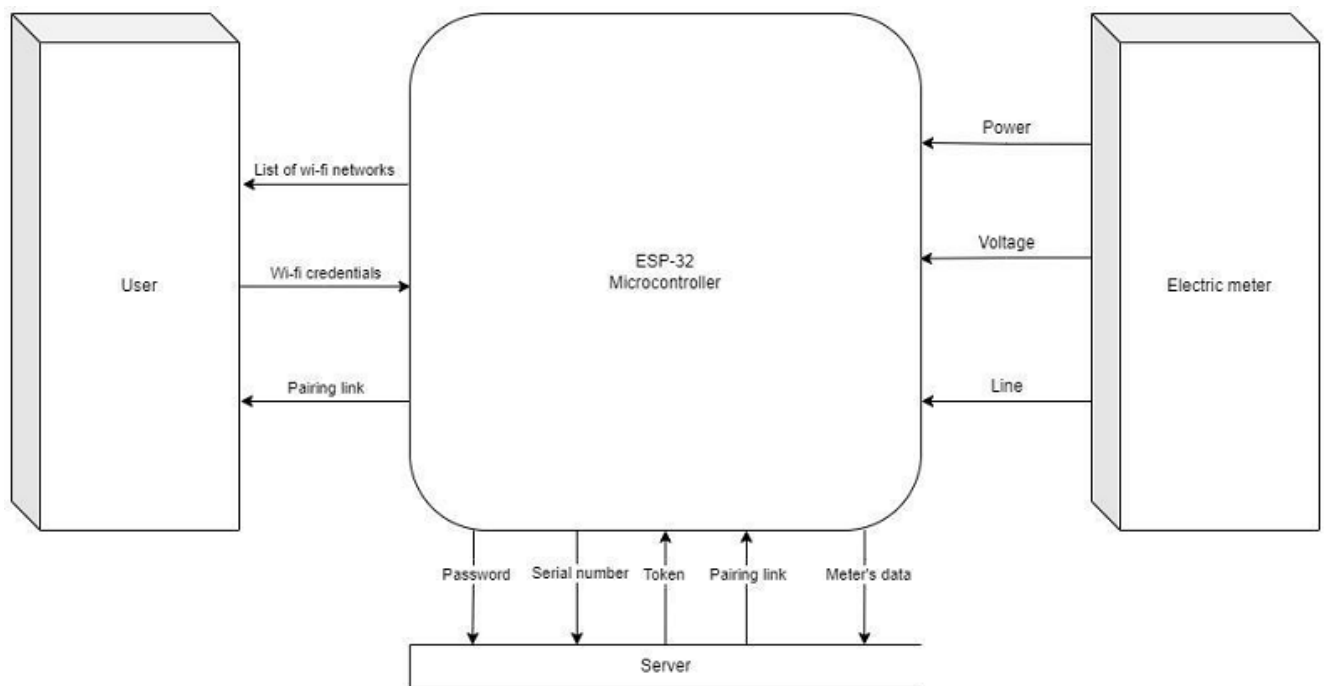


Рисунок 4.1 — діаграма потоку даних проекту

На цій діаграмі видно, як користувач за допомогою Bluetooth і смартфона проводить первинну конфігурацію та реєстрацію свого лічильника в системі

сервера. Це є критичним для інтеграції лічильника в екосистему інтернету речей та його підключення до мережі.

Мікроконтролер, у свою чергу, безперервно займається збором та обробкою даних, незалежно від того, підключений він до мережі чи ні. Дані, які збираються, включають показники напруги та споживання енергії для кожної з фаз лічильника.

Інтерація між сервером і мікроконтролером заснована на принципах захисту даних користувачів. З метою забезпечення безпеки та автентичності даних використовуються методи шифрування та авторизація з унікальними токенами, генерованими сервером. Це дозволяє уникнути ризику несанкціонованого доступу та маніпуляції з даними про споживання енергії.

Таким чином, структурна схема та DFD-діаграма надають цінне уявлення про всі ключові елементи проекту, дозволяючи здійснювати кроки розробки систематично та з урахуванням усіх аспектів взаємодії між компонентами системи.

4.2 Сценарії роботи системи

Як типова для сучасних IoT-систем, система мікроконтролера EnMonCo працює за різними сценаріями в залежності від конкретних умов. Це може бути введення даних користувачем, наявність підключення до інтернету, перший запуск системи або вже існуюча реєстрація лічильника в системі. Ці фактори визначають логіку виконання програмного коду.

Різні моделі поведінки системи демонструються на діаграмі послідовностей, представлений на рисунку 4.2.

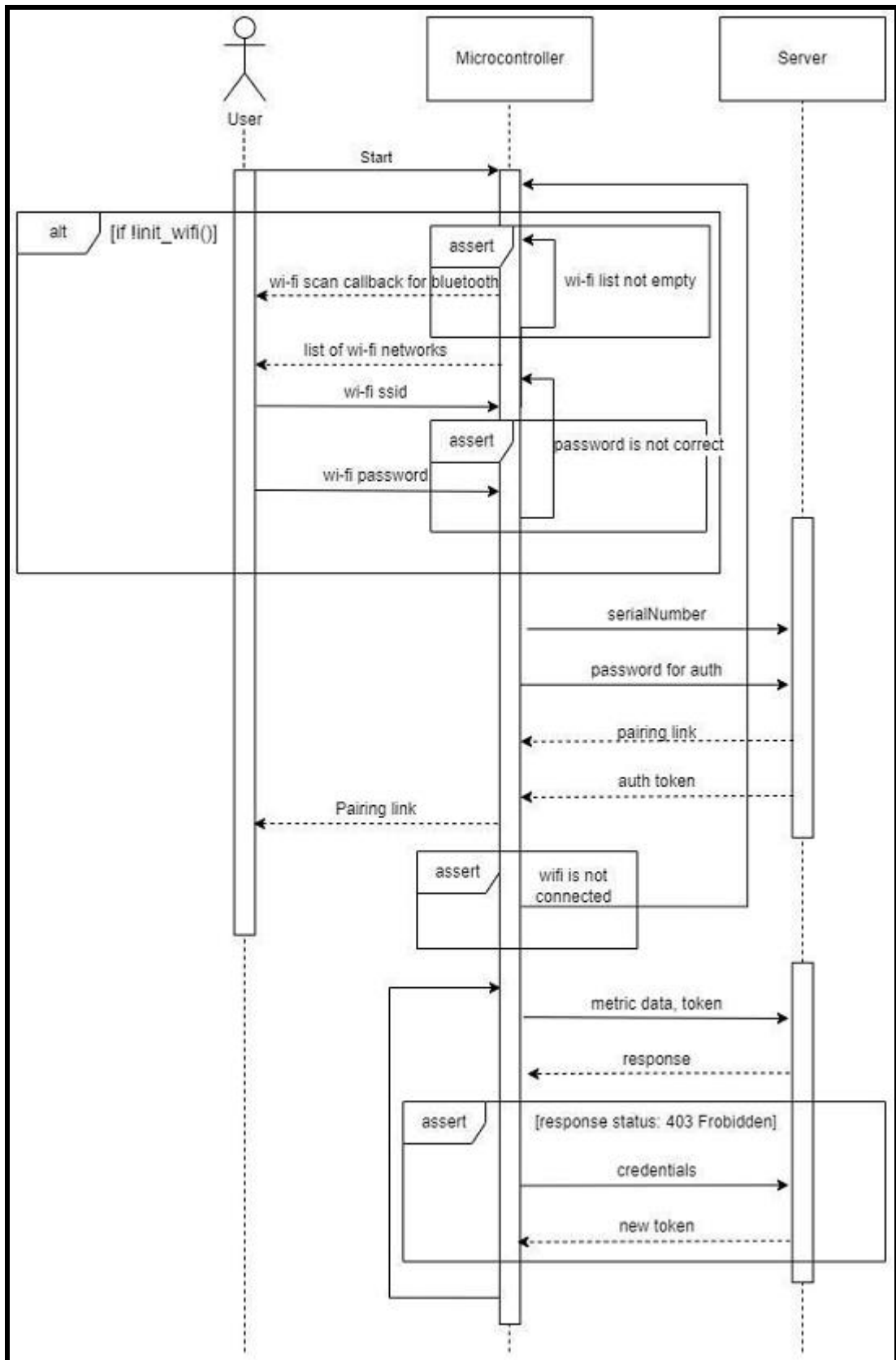


Рисунок 4.2 — Діаграма послідовностей системи

На основі аналізу цієї діаграми можна зробити висновок про ступінь автономності системи. EnMonCo розроблено таким чином, що він здатний працювати незалежно від безпосереднього втручання користувача. При первинному запуску необхідно лише під'єднати мікроконтролер до домашньої мережі та зареєструвати новий продукт на вебсайті за допомогою отриманого посилання. Далі EnMonCo функціонує автономно, хіба що потрібно змінити мережеві налаштування за бажанням користувача.

4.3 Взаємодія мікроконтролера з сервером

У проєкті розумного лічильника EnMonCo, який є частиною сучасного IoT-світу, реалізована система взаємодії з сервером, яка підтримує роботу як через протокол HTTPS для безпечної передачі даних, так і через звичайний HTTP протокол у разі необхідності зміни серверу. Ця гнучкість в комунікації досягається завдяки використанню API, який визначає специфічні шляхи та методи для взаємодії, включаючи отримання та відправку інформації.

Сервер EnMonCo містить два окремих API - один для запитів, пов'язаних з користувачами, та інший для запитів, що стосуються лічильників, причому мікроконтролером використовується тільки останній. Для наглядного представлення інтерфейсів програмування була розроблена графічна документація, зображена на рисунках 4.3 і 4.4.

Auth		^
POST	/api/v1/auth	✓
POST	/api/v1/auth/register	✓
GET	/api/v1/auth/me	✓ 🔒
Home		^
GET	/	✓
UserMeters		^
GET	/api/v1/users/{userId}/meters	✓ 🔒
GET	/api/v1/users/{userId}/meters/{meterId}	✓ 🔒
PATCH	/api/v1/users/{userId}/meters/{meterId}	✓ 🔒
DELETE	/api/v1/users/{userId}/meters/{meterId}	✓ 🔒
POST	/api/v1/users/{userId}/meters/{hash}/pair	✓ 🔒

Рисунок 4.3 — API для взаємодії мікроконтролера із сервером. Частина 1

Data		^
POST	/api/v1/meters/{meterId}/data	✓ 🔒
GET	/api/v1/meters/{meterId}/data	✓ 🔒
GET	/api/v1/meters/{meterId}/data/live	✓ 🔒

Рисунок 4.4 — API для взаємодії мікроконтролера із сервером. Частина 2

Ці рисунки демонструють, що основний зв'язок мікроконтролера з сервером відбувається за допомогою JSON формату, який спрощує процес формування запитів та аналізу відповідей сервера. Використання бібліотеки "ArduinoJSON" полегшує роботу з форматом JSON, надаючи зручні інструменти для створення та обробки JSON документів. Зокрема, для зберігання та обробки даних використовується тип `JsonDocument`, який може бути налаштований на певний розмір та заповнений за допомогою перевантаженого оператора індексування.

Ключовим елементом безпеки в системі є використання унікальних токенів для ідентифікації лічильника, які сервер надає під час процесу авторизації. Авторизація потребує ID та пароля, збережених у пам'яті пристрою, і вимагає

регулярного оновлення токена кожні 24 години для забезпечення безперервної взаємодії з сервером.

Для забезпечення захищеної комунікації через протокол HTTPS та надійного шифрування передаваних даних, було використано спеціалізовану бібліотеку "WiFiClientSecure". Ця бібліотека є інтегрованою частиною ширшого фреймворку "Esp IDF", розробленого та підтримуваного компанією Espressif Systems. Важливо відмітити, що ця бібліотека відрізняється від "ArduinoJSON", остання повністю розробляється та підтримується компанією Adafruit і не є природно призначеною для використання з цим типом мікроконтролерів.

У контексті взаємодії з сервером, ключовим аспектом є наявність сертифіката сервера та здатність шифрувати дані за допомогою SSL. Ці завдання вирішуються автоматично за допомогою бібліотеки "WiFiClientSecure". Проте, створення HTTP запитів, які мають уніфіковану структуру незалежно від процесу шифрування, було здійснено без залучення додаткових бібліотек. Для формування таких запитів використовується функція "form_http_request", до якої необхідно передати метод, шлях запиту, сервер, а також, за необхідності, токен та тіло запиту.

Відповіді від сервера отримуються у вигляді масиву байтів, які потім конвертуються у строковий формат. Ці відповіді містять як заголовки, так і тіло повідомлення, форматоване у JSON. Спеціально розроблені методи дозволяють обробляти ці відповіді, витягуючи ключову інформацію з них. Наприклад, метод "parse_response_status" повертає статус відповіді, на основі якого можна визначити, чи було запит оброблено успішно.

Всі можливі статус-коди, які може повернути сервер, зберігаються у програмі мікроконтролера як константи. Отриманий від сервера код порівнюється з цими константами, що дозволяє визначити подальші дії. Для забезпечення безпеки, разом із статус-кодами зберігається також сертифікат сервера. Цей електронний підпис використовується для перевірки, чи відповідь надійшла саме від того сервера, на який було спрямовано запит, запобігаючи тим самим можливості передачі даних неуповноваженим третім сторонам.

4.4 Взаємодія сенсорів у одній мережі із контролером

Принцип взаємодії сенсорів із мікроконтролером у мережі за допомогою MQTT протоколу базується на моделі публікації та підписки. У цьому процесі, кожен сенсор функціонує як видавець (publisher), який збирає дані з навколишнього середовища та пересилає їх до мікроконтролера. Мікроконтролер, в свою чергу, діє як клієнт, що підключається до MQTT брокера - централізованого вузла, який керує потоками даних між різними пристроями у мережі. На рисунку 4.5 зображено процес роботи протоколу.

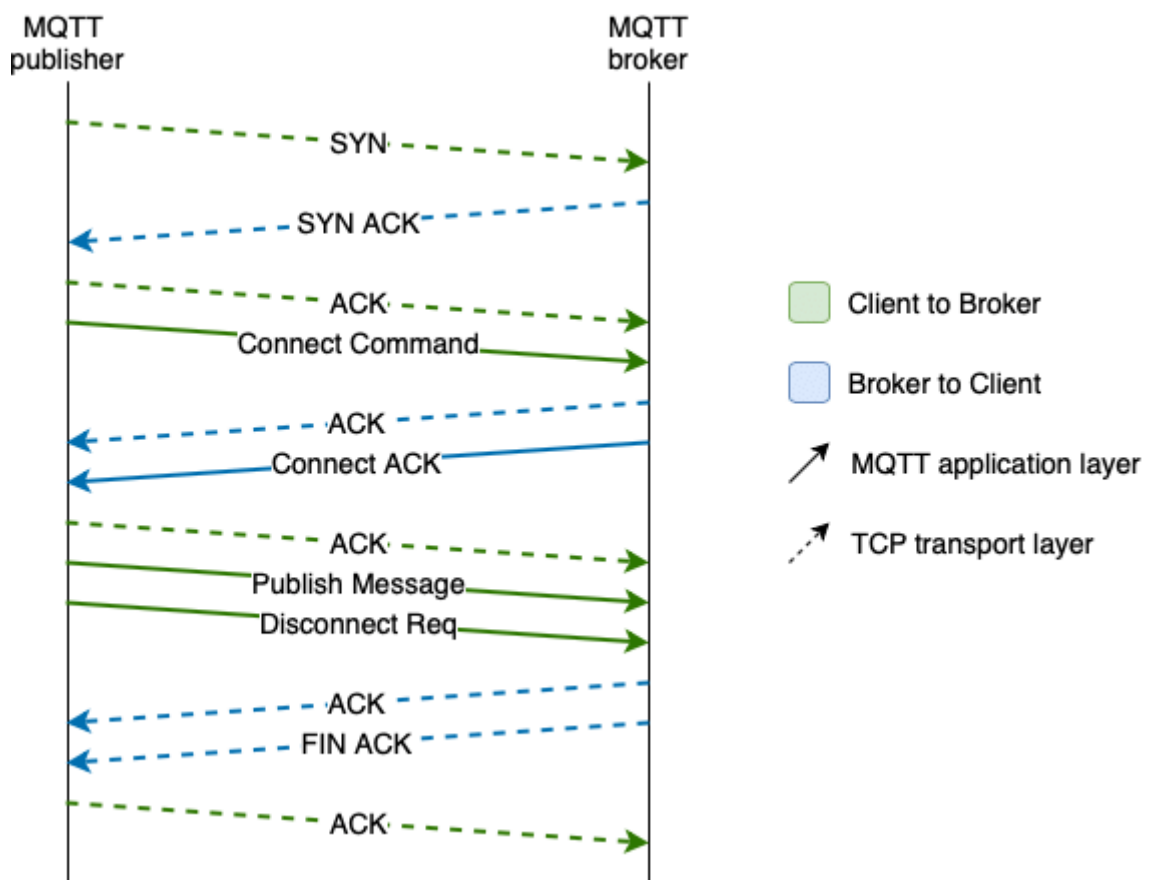


Рисунок 4.5 — Діаграма передачі даних по протоколу MQTT

Коли сенсор виявляє будь-які зміни в навколишньому середовищі, він генерує повідомлення з відповідними даними. Це повідомлення містить тему (topic) та власне інформацію, яка відображає стан чи показники сенсора. Тема виступає як адреса, за якою MQTT брокер визначає, куди слід направити ці дані.

Мікроконтролер, який підписаний на відповідні теми, отримує ці дані від брокера і може виконувати на їх основі певні дії або приймати рішення.

Протокол MQTT дуже ефективний у ситуаціях, де потрібна висока надійність та мінімізація затримок у передачі даних, а також у випадках, коли ресурси мережі та пристроїв обмежені. Така модель забезпечує надійну та ефективну передачу даних між сенсорами та мікроконтролером, дозволяючи системі оперативно реагувати на зміни у навколишньому середовищі та забезпечувати автоматизацію процесів у реальному часі.

4.5 Прогнозування енергоспоживання за допомогою ARIMA

Прогнозування часових рядів енергоспоживання за допомогою ARIMA моделі є складним процесом, що включає кілька ключових етапів. ARIMA, що розшифровується як Авторегресійна Інтегрована Модель Ковзного Середнього, є популярним статистичним методом для аналізу та прогнозування часових рядів.

Уявімо, що ми маємо історичні дані про споживання електроенергії. Перший крок у використанні ARIMA моделі - це перевірка часового ряду на стаціонарність. Стаціонарний часовий ряд має постійне середнє та дисперсію з часом. Якщо дані не є стаціонарними, вони повинні бути перетворені, наприклад, за допомогою диференціювання, щоб досягти стаціонарності.

ARIMA модель характеризується трьома параметрами: p (порядок компоненти авторегресії), d (ступінь диференціювання), q (порядок компоненти ковзного середнього). Визначення цих параметрів є ключовим для точності моделі. Параметр ' p ' визначає кількість лагів часового ряду, що використовуються як змінні прогнозування. Параметр ' d ' вказує, скільки разів часовий ряд був диференційований. Нарешті, ' q ' визначає розмір вікна ковзного середнього.

Після визначення параметрів, модель ARIMA "навчається" на історичних даних. Вона аналізує шаблони у споживанні електроенергії та використовує ці знання для прогнозування майбутнього споживання. Наприклад, модель може

виявити періодичні піки в енергоспоживанні, що повторюються щотижня або залежать від сезону, і використовувати цю інформацію для вирахування майбутніх значень.

Завершальний етап - це перевірка точності моделі. Це зазвичай робиться шляхом порівняння прогнозів моделі з фактичними даними, які не використовувалися при навчанні. Цей процес допомагає зрозуміти, наскільки надійними є прогнози моделі, та робить можливим її подальше налаштування для підвищення точності.

Таким чином, використання ARIMA для прогнозування часових рядів енергоспоживання дозволяє розробити модель, яка може адекватно передбачати майбутні тренди на основі минулих даних, що є особливо корисним для планування та оптимізації ресурсів..

Висновки до розділу 4

Розділ 4 дає детальний огляд програмної реалізації проекту "EnMonCo", акцентуючи увагу на важливості структурної схеми та DFD-діаграми для візуалізації та розуміння взаємодії між компонентами системи. Описано важливість первинної конфігурації та реєстрації лічильника, що відіграє ключову роль у інтеграції пристрою в інтернет речей. Також, деталізовано процес безперервного збору та обробки даних мікроконтролером, включаючи захист даних та використання методів шифрування. Взаємодія між сенсорами та мікроконтролером через MQTT протокол та прогнозування енергоспоживання за допомогою ARIMA моделі демонструють технічну складність та інноваційний підхід у розробці проекту. Цей розділ підкреслює глибоке технічне розуміння та системний підхід, який був застосований для реалізації проекту "EnMonCo".

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1 Системні вимоги

Для роботи з веб-додатком на настільному комп'ютері необхідні системні вимоги:

- операційна система Windows 7,8,10, Ubuntu ≥ 16 ;
- процесор з 1 ядром, від 2.0 ГГц;
- оперативна пам'ять — 2 Гб або більше;
- браузер та доступ до інтернету;

Для роботи з веб-додатком на смартфоні

- оперативна пам'ять — 2 Гб або більше;
- браузер та доступ до інтернету;

Для роботи із мікроконтролером на смартфоні або комп'ютері потрібен лише браузер та можливість під'єднання до мережі Wi-Fi

5.2 Взаємодія користувача з мікроконтролером

Після включення мікроконтролера в мережу, відбувається взаємодія з користувачем через мобільний додаток "Serial Bluetooth Terminal", який можна завантажити з "Google Play". Запустивши додаток, користувач вибирає "EnMonCo" на панелі керування, що забезпечує з'єднання смартфона з мікроконтролером через Bluetooth. Якщо мікроконтролер ще не підключений до Wi-Fi мережі, в терміналі додатка з'явиться перелік доступних Wi-Fi мереж. Користувачу потрібно обрати мережу та ввести пароль для підключення. У випадку невдачі процедура повторюється. Успішне підключення до інтернету відображає на екрані локальну IP-адресу роутера та посилання для реєстрації лічильника на сайті.

Авторизаційні дані Wi-Fi мережі зберігаються у flash-пам'яті пристрою, що дозволяє мікроконтролеру автоматично підключатися до мережі після перезапуску без участі користувача. Водночас, користувач може переглядати інформацію про стан підключення та активацію лічильника через свій акаунт EnMonCo у додатку.

Комунікація з користувачем реалізована через такі бібліотеки як "Bluetooth_serial" для передачі даних через Bluetooth, "WiFi" для підключення до мережі та обміну даними, і "Preferences" для збереження важливих даних у flash-пам'яті. Ці бібліотеки є частиною стандартного набору бібліотек для чіпу esp32 у рамках IDF (IoT Development Framework). "Bluetooth_serial" відповідає за ініціацію зворотного виклику Bluetooth, в залежності від стану підключення до мережі. "WiFi" використовується для з'єднання з Wi-Fi мережею та передачі даних, а "Preferences" грає ключову роль у збереженні налаштувань і статусу підключення, забезпечуючи стабільну та автономну роботу пристрою навіть при нестабільному з'єднанні чи перезавантаженні.

Також користувач може використати принцип першої настройки пристрою через підключення до його відкритої точки доступу. Коли пристрій вперше включається, він автоматично створює відкриту Wi-Fi. Ця мережа не захищена паролем, що дозволяє користувачам легко до неї підключитися.

Користувач має знайти цю відкриту мережу на своєму смартфоні або комп'ютері та підключитися до неї. Після успішного підключення до цієї тимчасової мережі, користувач зазвичай використовує веб-браузер або спеціалізований додаток для доступу до інтерфейсу налаштувань пристрою.

У веб-інтерфейсі або через додаток користувач може налаштувати різні параметри пристрою, включаючи підключення до домашньої Wi-Fi мережі. Це включає введення SSID (назви мережі) та пароля домашньої Wi-Fi мережі, щоб пристрій міг підключитися до неї та інтегруватися в домашню мережу.

Після завершення налаштувань, пристрій автоматично перезавантажується та намагається підключитися до вказаної домашньої Wi-Fi мережі. Якщо підключення

вдається, відкрита точка доступу вимикається, і пристрій починає нормально функціонувати в рамках домашньої мережі.

Цей метод дуже зручний для користувачів, оскільки не вимагає додаткових пристроїв або складних налаштувань для ініціалізації нових пристроїв у мережі. На рисунку 5.1 зображено веб-інтерфейс мікроконтролера.

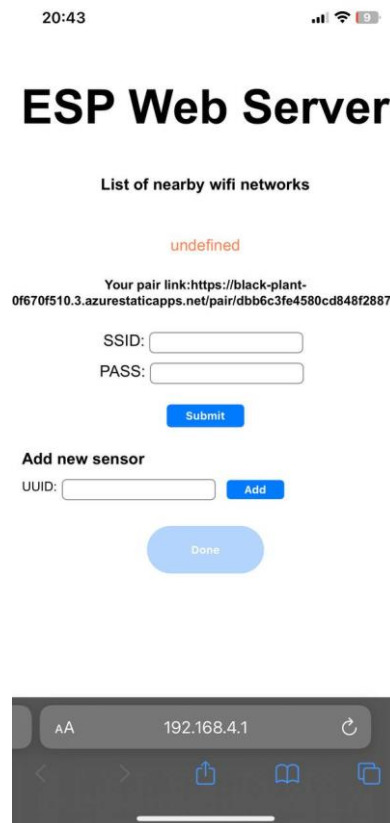


Рисунок 5.1 — Веб-інтерфейс мікроконтролера

Після початкового налаштування, у користувача є можливість додати всі сенсори у своїй локальній мережі до пристрою. Робиться це через веб-інтерфейс мікроконтролера де треба вписати серійний номер сенсору до поля “add new sensor”.

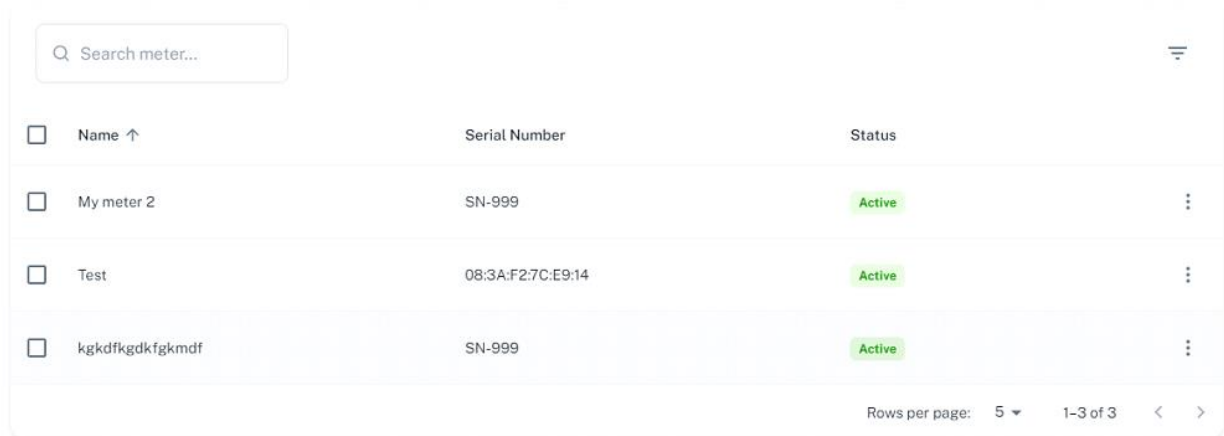
5.3 Графічний інтерфейс

Графічний інтерфейс цього проекту був уміло реалізований у формі веб-сайту, створеного з метою забезпечення ефективної взаємодії користувачів з їхніми

електроенергетичними лічильниками. В основу розробки було покладено використання мови програмування Javascript разом із фреймворком React, відомим своєю гнучкістю та ефективністю у створенні інтерактивних веб-інтерфейсів. Однією з ключових функцій цього інтерфейсу стала візуалізація даних про споживання енергії та напруги у вигляді лінійних графіків, які дозволяють користувачам чітко бачити тенденції споживання за різні часові періоди.

Було також створено зручну таблицю лічильників, яка спрощує процес управління та моніторингу різних пристроїв користувача. На даний момент, ця таблиця в основному надає інформацію про перший доданий лічильник, але в майбутніх оновленнях планується розширення функціональності для включення даних з інших пристроїв. На рисунку 5.2 відображено список зареєстрованих у системі лічильників.

Meters



☐ Name ↑	Serial Number	Status	
☐ My meter 2	SN-999	Active	⋮
☐ Test	08:3A:F2:7C:E9:14	Active	⋮
☐ kgkdfkgdkfgkmdf	SN-999	Active	⋮

Rows per page: 5 1-3 of 3 < >

Рисунок 5.2 — Таблиця зареєстрованих лічильників

У таблиці, представлений на цій ілюстрації, можна побачити три доданих пристрої. Варто зазначити, що лічильники із серійним номером SN-999 є не реальними, а були створені виключно для цілей тестування через базу даних. Зокрема, лічильник з назвою "Test", який був створений як перший, демонструє всі дані на головній сторінці сайту.

Важливою частиною розробки було використання бібліотеки "ReactApexChart" для створення графіків. Ця бібліотека надає широкий спектр інструментів для налаштування графіків, що дозволяє користувачам модифікувати їх відповідно до своїх індивідуальних потреб. У процесі розробки EnMonCo було приділено особливу увагу створенню інтуїтивно зрозумілого, мінімалістичного, але при цьому функціонального та інформативного інтерфейсу. Наприклад, графіки споживання енергії розміщені таким чином, щоб займати достатньо місця на веб-сторінці, при цьому не перевантажуючи її надмірною кількістю інформації, що є явною перевагою в порівнянні з деякими аналогічними застосунками, такими як SmartMAIC. На ілюстрації 5.3 зображено графік споживання енергії.



Рисунок 5.3 — Графік споживання енергії за даними лічильника "Test"

Гнучкість інтерфейсу бібліотеки "ReactApexChart" також відкриває додаткові можливості для взаємодії з графіком, зокрема збільшення масштабу для детального аналізу споживання енергії (зображений на рисунку 5.4).

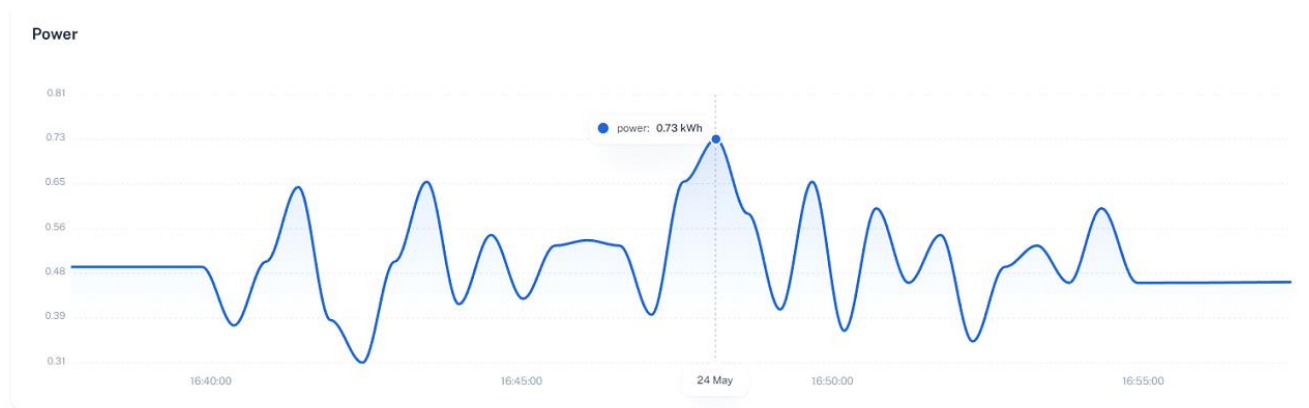


Рисунок 5.4 — Графік споживання енергії за даними лічильника “Test”.

Наближена версія

Це дозволяє користувачам легко виявляти важливі тенденції, такі як пікові навантаження або спади енергоспоживання протягом дня чи окремих годин.

5.4 Графічний інтерфейс

З огляду на відсутність вбудованого дисплея на лічильнику для відображення даних у реальному часі, була розроблена альтернативна функція у веб-застосунку. Ця інновація дозволяє користувачам візуально спостерігати за поточними показниками своїх лічильників через інтерфейс веб-сайту, забезпечуючи зручний та доступний моніторинг. Для реалізації цього функціоналу, веб-застосунок регулярно, кожні десять секунд, надсилає запити на сервер для отримання останніх доступних даних від обраного лічильника. Це забезпечує постійне оновлення інформації, дозволяючи користувачам бути в курсі поточних показників енергоспоживання.

Інтерфейс моніторингу, зображений на рисунку 5.5, включає інтуїтивно зрозумілі віджети, які відображають найновіші дані, отримані з лічильників. Особливість цих віджетів полягає в їх здатності змінювати колір в залежності від показників, які вони відображають. Наприклад, вони стануть червоного кольору, якщо значення напруги у мережі перевищать 240 Вольт або якщо споживання

енергії складе більше 0.5 кВт/год. Такий підхід не тільки забезпечує більш динамічне та візуально привабливе відображення даних, але й підвищує увагу користувача до критичних змін в енергоспоживанні.



Рисунок 5.5 — Відображення параметрів у реальному часі

Ця система моніторингу в реальному часі є ключовою для ефективного управління споживанням енергії та може бути особливо корисною для ідентифікації незвичних витрат або несправностей у системі. Включення цього функціоналу в веб-застосунок відкриває нові можливості для користувачів, дозволяючи їм отримувати повний контроль та прозорість у використанні електроенергії.

Висновки до розділу 5

Розділ 5 описує взаємодію користувача з програмною системою, вказуючи на необхідні системні вимоги для ефективного використання веб-додатку як на настільних комп'ютерах, так і на смартфонах. Ці вимоги включають наявність оперативної пам'яті, відповідної операційної системи та доступу до інтернету. Особливий акцент робиться на взаємодії користувача з мікроконтролером через мобільний додаток "Serial Bluetooth Terminal", що дозволяє здійснювати первинну конфігурацію та підключення до Wi-Fi мережі. Підкреслюється, що авторизаційні

дані Wi-Fi мережі зберігаються у flash-пам'яті пристрою для забезпечення автоматичного підключення після перезапуску.

Для більш ефективної взаємодії з мікроконтролером користувач може використовувати принцип першої настройки через відкриту точку доступу Wi-Fi. Це дозволяє легко підключити пристрій до домашньої мережі через веб-браузер або спеціалізований додаток, забезпечуючи гнучке налаштування та інтеграцію пристрою в домашню мережу.

Графічний інтерфейс проекту, створений за допомогою Javascript та React, пропонує користувачам інтуїтивно зрозумілий і функціональний спосіб моніторингу споживання енергії через візуалізацію даних за допомогою лінійних графіків та таблиць. Використання бібліотеки "ReactApexChart" для створення графіків надає можливість детального аналізу споживання енергії, включаючи виявлення пікових навантажень чи спадів. Це допомагає користувачам ефективно управляти своїм енергоспоживанням та ідентифікувати потенційні проблеми чи несправності у системі.

Таким чином, розділ 5 підсумовує ключові аспекти роботи користувача з програмною системою, підкреслюючи її гнучкість, зручність та функціональність у повсякденному використанні.

6 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

6.1 Опис ідеї проєкту

Ідея проєкту полягає в розробці інтегрованої системи обліку електроенергії, призначеної для вдосконалення процесів моніторингу та управління енергоспоживанням в розумних будинках. Система буде включати в себе функції збору даних, їх аналізу та візуалізації, забезпечуючи користувачам глибше розуміння їх енергетичного профілю.

Проект обіцяє значний потенціал завдяки застосуванню сучасних алгоритмів вейвлет-перетворення, які дозволяють ефективно обробляти великі об'єми даних та швидко ідентифікувати важливі тенденції у споживанні енергії. Розробка спрямована на створення легкого в користуванні інтерфейсу, що включає в себе інтуїтивно зрозумілі інструменти для аналізу та контролю за споживанням.

В ході дослідження були проаналізовані існуючі рішення на ринку, виявлені їх сильні та слабкі сторони. В результаті було встановлено, що запропонована система може значно спростити процеси обліку енергоспоживання та забезпечити користувачам зручні інструменти для управління енергоресурсами.

Деталі ідеї проєкту, його застосування та переваги для користувачів представлені в таблиці 6.1, яка систематизує основні аспекти розробки та її потенціал на ринку.

Таблиця 6.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення інтегрованої системи обліку електроенергії призначеної для вдосконалення процесів	1. Використання системи домогосподарствами та підприємствами	Доступ до даних лічильників в одному місці, можливість передбачити дані з лічильників

Продовження таблиці 6.1

1	2	3
моніторингу та управління енергоспоживанням в розумних будинках.	2. Використання системи компаніями з постачання електроенергії	Можливість відслідковувати дані з лічильників електроенергії, можливість перевіряти користувачів та їх лічильники

У якості аналогів було розглянуто програмні продукти, які також дають змогу переглядати дані з лічильників електроенергії, а саме Tesla та SmartMike. Аналіз систем наведено в таблиці 6.2.

Таблиця 6.2 — Визначення сильних, слабких та нейтральних характеристик ідеї систем обліку електроенергії.

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			Слабкі (W), нейтральні (N) та сильні (S) сторони		
		Проект EnMonCo	Tesla	SmartMaic	W	N	S
1.	Доступність	Всюди, де є інтернет	Всюди, де є інтернет	Всюди, де є інтернет		+	
2.	Основні технології	Інтегровані бази даних	Інтегровані бази даних	Інтегровані бази даних		+	

Продовження таблиці 6.2

1	2	3	4	5	6	7	8
3.	Повнота функціоналу	Моніторинг Алгоритми аналізу даних Алгоритми передбачення витрат	Моніторинг Керування Аналіз	Моніторинг			+
4.	Надійність	Висока	Висока	Висока			+
5.	Залежність від інтернету	Залежить	Залежить	Залежить		+	
6.	Кількість проведених тестувань	Дуже мала	Дуже велика	Дуже велика	+		

6.2 Технологічний аудит ідеї проєкту

Був проведений аудит технологій, за допомогою яких можна поєднати кращі практики систем конкурентів в один багатфункціональний застосунок, що реалізує ідею проєкту, було визначено доступність технологій, а також чи потребують вони подальшого доопрацювання для реалізації ідеї. Технологічний аудит представлений в таблиці 6.3.

Таблиця 6.3 — Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Розробка системи моніторингу та оптимізації споживання енергоресурсів EnMonCo	- використання HTML, CSS, JavaScript та фреймворку React для розробки фронт-енд частини веб-додатку.	Наявна	Доступна
		- використання бібліотеки стилів Material UI для створення зручного інтерфейсу програмної системи.	Наявна	Доступна
		- використання фреймворку NestJS для реалізації бекенд частини програмного застосунку	Наявна	Доступна
		- використання бібліотеки ReactApexChart для побудови графіків.	Наявна	Доступна
		- IDF від Espressif Systems та Paho від Eclipse для комунікації з сенсорами	Наявна	Доступна

Обрані технології реалізації проєкту: HTML, CSS, JavaScript, React, NestJs розробка велась у середовищі WebStorm, Clion, Arduino. Бібліотеки з відкритим програмним кодом: Material UI, ReactApexChart, ArimaJs, IDF, Paho, Arduino.

6.3 Аналіз ринкових можливостей запуску стартап-проєкту

Аналіз наявних ринкових можливостей для ринкової реалізації проєкту та ринкових загроз, які можуть перешкоджати реалізації проєкту, дозволяє спланувати напрям розвитку продукту з урахуванням стану ринкового середовища та потреб. Оцінити пропозиції, які надають конкуренти та дізнатись потреби потенційних клієнтів.

Потенційні користувачі та їхня характеристика, а також сформований перелік орієнтовних вимог до пропозиції представлено в таблиці 6.4.

Таблиця 6.4 — Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності в поведінці різних цільових груп	Вимоги споживачів до товару
1.	Зменшення загального споживання енергоресурсів	Великі підприємства, домогосподарств а та звичайні користувачі	Використання в якості навчального середовища для отримання знань	Наявність оптимізаційних алгоритмів для контролю за споживанням
2.	Забезпечення зручного та повнофункціонального середовища для моніторингу споживання електроенергії	Великі підприємства, домогосподарств а та звичайні користувачі	Використання в якості системи для контролю виробничого процесу/ведення власної статистики	Наявність детальних графіків використання та зручного та інтуїтивного інтерфейсу у веб-додатку

Після проведеного аналізу різних груп клієнтів було проаналізовано поточний ринок. За результатами даних досліджень створено таблицю факторів, які сприяють або перешкоджають впровадженню проекту на ринок. Фактори загроз та фактори можливостей представлені в таблиці 6.5 та таблиці 6.6 відповідно.

Таблиця 6.5 — Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Інертність суспільства	Небажання переходити на новий комплексний продукт	Проведення маркетингової кампанії всередині університету.
2.	Конкуренція	Наявність конкурентів зі схожим функціоналом	Вдосконалення уже наявного та введення нового функціоналу

Таблиця 6.6 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Висока якість продукту	Витіснення конкурентів з ринку	Розширення підтримки програмного продукту
2.	Нові закони Євросоюзу про зберігання енергії	Примушення людей на перехід до більшої економії енергоресурсів	Збільшення маркетингової компанії нашого продукту для отримання максимальної кількості клієнтів у даній ситуації

Продовження таблиці 6.6

1	2	3	4
3.	Велика кількість користувачів	Модернізація програмного забезпечення для виходу за рамки навчання лише аспірантів	Впровадження нового функціоналу програмного продукту.

Далі проведено аналіз пропозицій, за яким визначаються загальні риси конкуренції ринку (табл. 6.7).

Таблиця 6.7 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоби бути конкурентоспроможною)
1. Тип конкуренції: - оліпологія	На ринку існує не багато різних готових рішень, більшість з них не надають повний необхідний функціонал	Створення продукту високої якості та постійне вдосконалення
2. Галузева ознака: - міжгалузева	Використання продукту для зменшення витрат на використання енергоресурсів	Покращення функціоналу та популяризація продукту серед підприємств

Продовження таблиці 6.7

1	2	3
3. Рівень конкурентної боротьби: - національний	Компанії-конкуренти націлені на український ринок	Розвиток продукту з подальшим виходом на міжнародний ринок
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між програмним забезпеченням одного виду	Покращення програмного продукту за рахунок випуску новіших версій
5. Характер конкурентних переваг: - нецінова	Функціональні можливості програмних засобів	Збільшення функціональних можливостей
6. Інтенсивність: - не марочна	Важливим є якість продукту, а не бренд	Покращення продукту, реалізація нових алгоритмів

Після вивчення конкурентного середовища, проведено глибокий аналіз умов конкуренції у галузі за допомогою моделі п'яти сил М. Портера (табл. 6.8).

Таблиця 6.8 — Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Tesla, SmartMAIC та схожі програмні продукти	Відносно невисокий бар'єр через неповноту функціоналу, обмеженості доступу до навчальних матеріалів університету	Залежність від постачальників відсутня, система є незалежною	Споживачі є вагомим фактором	Відсутні. Наявні лише конкуренти зі схожими розробками деякого функціоналу
Висновки	Активна конкурентна боротьба вимагає створення застосунку з повним, зручним та необхідним функціоналом.	Вхід на ринок є невисоким, можливість появи конкурентів мало ймовірна	Залежність від постачальників відсутня, система є незалежною	Сильний вплив на продукт мають споживачі, отже, подальший розвиток продукту залежить від їхніх вподобань та вимог.	Для утримання позицій на ринку необхідно підтримувати високу якість продукту.

На підставі розгляду конкурентного середовища згідно з моделлю М. Портера, який був проведений у таблиці 6.8, а також з урахуванням попередніх таблиць, визначено та обгрунтовано перелік факторів конкурентоспроможності у таблиці 6.9.

Таблиця 6.9 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Потреби споживачів	Необхідність розробки проекту обумовлена потребами споживачів.
2.	Повнота функціоналу	Програмний продукт надає необхідний обсяг користувацьких послуг
3.	Надійність	Програмний продукт працює коректно, непередбачувані випадки відсутні
4.	Не залежить від можливостей пристроїв та платформ	Продукт можна використовувати на багатьох пристроях та платформах.
5.	Ціна продукту	Нижча за рахунок невідомості бренду
6.	Технічне обслуговування	Постійне оновлення програмного забезпечення та впровадження нових версій

З врахуванням визначених факторів конкурентоспроможності, які представлені у таблиці 6.9, був здійснений аналіз сильних та слабких сторін стартап-проекту (табл. 6.10).

Таблиця 6.10 — Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним проектом						
			-3	-2	-1	0	+1	+2	+3
1.	Потреби споживачів	16						+	
2.	Повнота функціоналу	12					+		

Продовження таблиці 6.10

1	2	3	4	5	6	7	8	9	10
3.	Надійність	10				+			
4.	Не залежить від пристроїв та платформ	18							+
5.	Ціна продукту	15						+	
6.	Технічне обслуговування	9		+					

За результатами порівняльного аналізу, можна зробити висновки, що проект, який створюється, має сильні переваги в порівнянні з конкурентами.

Техніко-економічну доцільність реалізації проекту було проаналізовано шляхом створення матриці SWOT-аналізу (матриця, що визначає сильні (Strength) та слабкі (Weak) сторони, ризики (Troubles) та можливості (Opportunities)). SWOT-аналіз проекту наведено в таблиці 6.11.

Таблиця 6.11 — SWOT-аналіз стартап-проекту

Сильні сторони (S): – Повнота функціоналу – Доступність – Незалежність від пристроїв та платформ – Ціна продукту	Слабкі сторони (W): – Обмежений ринковий вплив на початковому етапі через наявність схожих продуктів. – Низька репутація в період запуску проекту.
Можливості (O): – Розширення функціоналу програми – Покращення роботи системи	Загрози (T): – Небажання користувачів переходити на новий програмний продукт

На основі проведеного SWOT-аналізу розроблено альтернативну стратегію ринкової поведінки для введення стартап-проекту на ринок. Також визначено приблизний оптимальний час для їхньої реалізації, враховуючи потенційні проекти конкурентів, які можуть бути представлені на ринку.

Альтернативи для введення на ринок визначено і представлено у таблиці 6.12, яка містить інформацію щодо ринкового впровадження стартап-проекту.

Таблиця 6.12 — Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Організація демонстраційної конференції для виходу на ринок.	25%	12-14 місяців
2.	Безкоштовне розповсюдження створеного продукту	70%	9-12 місяців
3.	Розповсюдження продукту для інших навчальних закладів	55%	6-10 місяців

В результаті проведеного аналізу було вирішено передусім впровадити систему для КПП, оскільки розробка велася для цього навчального закладу.

6.4 Розроблення ринкової стратегії проекту

Розробка ринкової стратегії починається з визначення стратегії охоплення ринку. Визначено групи потенційних користувачів розробленої системи для діяльності вибраних сегментів ринку, які показано в таблиці 6.13.

Була обрана стратегія охоплення ринку вибіркового розподілу, що застосовується до товарів, що користуються високим попитом у споживачів. Диференціація була обрана як стратегія розвитку, щоб представити новий підхід і отримати лояльних клієнтів через демонстрацію продукту.

Таблиця 6.13 — Опис потенційних груп

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Домогосподарства	готові	високий	невелика	легка
2.	Підприємства	готові	високий	невелика	легка
3.	Компанії з постачання електроенергії	неготові	низький	велика	важка
Які цільові групи обрано: домогосподарства та підприємства					

Для ведення діяльності в обраному сегменті ринку була розроблена основна стратегія розвитку, яку представлено у таблиці 6.14.

Таблиця 6.14 — Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Розробка системи обліку електроенергії	Вибірковий розподіл	Розширення функціоналу, нововведення	Стратегія диференціації

За результатами проведеної роботи було обрано стратегію диференціації як базову стратегію конкурентної поведінки. Дана стратегія була вибрана, оскільки дана ніша потребує розроблення нового функціоналу для охоплення цільової аудиторії.

Основними вимогами до програмного продукту цільовою аудиторією є забезпечення зручними та зрозумілими функціями для досягнення високого рівня навчання на аспірантурі як для аспірантів так і для викладачів. При цьому веб додаток має бути зручним у користуванні та зосередити усю необхідну інформацію та функціонал в одному місці.

Висновки до розділу 6

В результаті аналізу ринкових можливостей розробленої системи було створено концепційний опис проекту, ідентифіковані слабкі та сильні сторони. Технічний аудит, який включав вибір мови програмування Обрані технології реалізації проекту: NodeJS, TypeScript, NestJS, React, розробка велась у середовищі WebStorm, бібліотеки з відкритим програмним кодом: Material UI, axios. Створення програмного продукту здійснювалась у середовищі розробки Webstorm.

Даний аналіз ринкових можливостей дозволив визначити конкурентоспроможність системи та визначити цільову аудиторію, якою є домогосподарства та підприємства. Базуючись на цьому аналізі, була обрана стратегія диференціації, як основна стратегія розвитку. Для розробки була обрана стратегія охоплення ринку вибіркового розподілу.

Ключовими факторами успіху проекту є якість та повнота розробленого функціоналу для забезпечення високого рівня освіти, при збереженні комфортного та зручного облікових процесів як для домогосподарств так і для підприємств. Також визначено, що важливо слідкувати за відгуками користувачів і систематично удосконалювати функціонал проекту.

ВИСНОВКИ

Мета цієї магістерської роботи полягала у створенні системи для ефективної оптимізації та прогнозування споживання енергоресурсів. Це передбачало аналіз існуючих методів і систем, збір і аналіз даних, розробку точної математичної моделі прогнозування та її інтеграцію з алгоритмами оптимізації. Реалізація програмного забезпечення, що включає ці моделі та алгоритми, дала змогу забезпечити своєчасне виявлення можливостей для зниження енергоспоживання. Тестування та валідація системи демонстрували її ефективність у передбаченні енергетичних потреб та оптимізації витрат. Розроблена система може бути корисною як для домогосподарств, так і для підприємств, сприяючи зниженню енергоспоживання та економії коштів. Система пройшла ретельне тестування та валідацію, підтверджуючи її спроможність забезпечувати точні прогнози та ефективні стратегії оптимізації. Завдяки цьому вдалося демонструвати значне зниження споживання енергії та витрат, це вказує на велику практичну цінність розробленого рішення в контексті сучасних потреб у сталому управлінні енергоресурсами. Також, дослідження відкриває перспективи для подальшого вдосконалення і адаптації системи до різних умов експлуатації, що може сприяти її широкому впровадженню та інтеграції з іншими інтелектуальними системами управління енергетичними ресурсами. Таким чином, робота ставить основу для розвитку більш сталого та ефективного енергетичного менеджменту в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Meyers D. Front-End Developer. BCS Learning & Development Limited, 2020. 232 p.
2. Meyers S. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14 / Scott Meyers., 2014. – 451 с. – (1st edition).
3. Arduino MKR 1000 WiFi. Arduino docs. URL: <https://docs.arduino.cc/hardware/mkr-1000-wifi> (date of access: 11.04.2023).
4. Sutter H. Exceptional C++: 47 Engineering Puzzles, Programming Problems, and Solutions / Herb Sutter., 1999. – 240 с. – (1st edition).
5. Margolis M. Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects / M. Margolis, B. Jepson, N. Weldin., 2020. – 798 с. – (3rd Edition).
6. 5100 / S. Warner et al. The american mathematical monthly. 1964. Vol. 71, no. 4. P. 449. URL: <https://doi.org/10.2307/2313283> (date of access: 24.10.2023).
7. Tsitoara M. Version control systems. Beginning Git and GitHub. Berkeley, CA, 2019. P. 3–18. URL: https://doi.org/10.1007/978-1-4842-5313-7_1 (date of access: 24.10.2023).
8. W3Schools. Learn HTML and CSS with W3Schools. Wiley & Sons, Incorporated, John, 2010. 240 p.
9. Middaugh J. Styling material-ui components: syntax, apis, and detailed examples. Independently Published, 2020.
10. Godbolt M. Frontend architecture for design systems: a modern blueprint for scalable and sustainable websites. O'Reilly Media, Incorporated, 2016.
11. Allums S. Designing mobile payment experiences: principles and best practices for mobile commerce. O'Reilly Media, Incorporated, 2014. 242 p.
12. Architecting modern data platforms: a guide to enterprise hadoop at scale / P. Wilkinson et al. ; ed. by N. Tache, M. Cronin. Beijing : O'Reilly Media, 2019. 636 p.

13. Axelrod A. The value of test automation. *Complete guide to test automation*. Berkeley, CA, 2018. P. 3–11. URL: https://doi.org/10.1007/978-1-4842-3832-5_1 (date of access: 06.01.2024).
14. Beck K. Test driven development: by example (the addison-wesley signature series). Addison-Wesley Professional, 2002. 240 p.
15. Boduch A. React material-ui cookbook: build captivating user experiences using react and material-ui. Packt Publishing, Limited, 2019. 534 p.
16. Chang M. Adaptive design theory and implementation using SAS and R. Taylor & Francis Group, 2016.
17. Cowell C., Lotz N., Timberlake C. Automating DevOps with GitLab CI/CD Pipelines: Build powerful CI/CD pipelines to verify, secure, and deploy your code using real-life examples. Packt Publishing - ebooks Account, 2023. 315 p.
18. Hoffman A. Web application security: exploitation and countermeasures for modern web applications. O'Reilly Media, 2020. 330 p.
19. Merode H. V. Continuous integration / continuous delivery by design: practical guide to designing and developing pipelines. Apress L. P., 2023.
20. Parker T. Designing with progressive enhancement: Building the Web that works for everyone. Berkeley, CA : New Riders, 2010.
21. Saridakis T. Design patterns for graceful degradation. *Transactions on pattern languages of programming I*. Berlin, Heidelberg, 2009. P. 67–93.
22. Springer S. Node.js: the comprehensive guide. Rheinwerk Publishing Inc., 2022.
23. Vanderkam D. Effective typescript: 62 specific ways to improve your typescript. O'Reilly Media, Incorporated, 2019. 250 p.
24. Whitehead K. Component-Based development: principles and planning for business systems (addison-wesley object technology series). Addison-Wesley Professional, 2002. 224 p.
25. W U. Agile scrum crash course: A guide to agile project management and scrum master certification PSM 1. Independently Published, 2020. 137 p.

ДОДАТКИ

ДОДАТОК А Лістинг розробленої програми

Diploma.ino

```
#include <Arduino.h>
#include <ArduinoJson.h>
#include "bluetooth_handle.h"
#include "wifi_handle.h"
#include "data_collection.h"
#include "server_request.h"
#include "http_parse.h"
#include "consts.h"
#include "utils.h"

TaskHandle_t Task0;
utils::custom_lock lock = utils::CONTINUE;

void setup()
{
    Serial.begin(115200);
    server_request::preferences.begin("server_access", false);
    Serial.println(consts::mac);
    if(server_request::preferences.getString("password") == "")
    {
        server_request::preferences.putString("password",
utils::generate_password(utils::generate_number(8.0f, 16.0f)));
    }
    Serial.println(server_request::preferences.getString("password"));
```

```
wifi_handle::init_wifi();
```

```
Serial.println(server_request::preferences.getString("id"));
```

```
xTaskCreatePinnedToCore(  
    core0_task, /* Task function. */  
    "Task0",    /* name of task. */  
    10000,      /* Stack size of task */  
    nullptr,    /* parameter of the task */  
    2,          /* priority of the task */  
    &Task0,     /* Task handle to keep track of created task */  
    0);  
}
```

```
void core0_task(void* pvParameter)  
{  
    while(true)  
    {  
        delay(1000);  
        if(lock == utils::CONTINUE)  
        {  
            for(size_t i = 0; i < 12; i++)  
            {  
                data_collection::data_fetch_sim();  
                delay(5000);  
            }  
            lock = utils::WAIT;  
        }  
    }  
}
```

```

}

void loop()
{
  try
  {
    while(!wifi_handle::is_connected())
      delay(3000);
    if(lock == utils::WAIT)
    {
      if(server_request::preferences.getString("id") == "")
      {
        Serial.println("registering your meter");
        server_request::server_register();
      }

      std::vector<float> voltage_copy(*data_collection::voltage);
      auto power_map_copy(data_collection::sensors_data);
      data_collection::sensors_data.clear();
      data_collection::voltage->clear();
      lock = utils::CONTINUE;

      data_collection::power_med = utils::find_sum_in_map(power_map_copy);
      data_collection::voltage_med = utils::found_median_of_vec(voltage_copy);

      String request_body =
server_request::form_meter_data_json(data_collection::power_med,
data_collection::voltage_med);

```

```

String token = server_request::preferences.getString("token");
String meter_id = server_request::preferences.getString("id");
String path = consts::paths::meter + meter_id + "/data";
String request = server_request::form_http_request
(
    std::move(consts::POST),
    std::move(path),
    std::move(consts::host),
    std::move(request_body),
    std::move(token)
);
String response = server_request::send_data_https(std::move(consts::host),
consts::port_tls, std::move(request), consts::ca_cert);
String response_status = http_parse::parse_response_status(response);
Serial.println(request_body);
Serial.println(response);
if(response_status == consts::status_forbidden || response_status ==
consts::status_unauthorized)
{
    server_request::get_new_token();
}
}
} catch(std::exception &ex)
{
    Serial.println(ex.what());
}
}

```

server_request.cpp

```
#include <vector>
#include <WiFiClientSecure.h>
#include "server_request.h"
#include "utils.h"
#include "http_parse.h"
#include "consts.h"

namespace server_request
{
    Preferences preferences;
    static WiFiClientSecure client;

    String send_data(const String&& ip, const short port, const String&& request)
    {
        std::vector<char> buf(2048);

        const char* ip_cstr = ip.c_str();
        int sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

        struct sockaddr_in serverAddress;
        serverAddress.sin_family = AF_INET;
        inet_pton(AF_INET, ip_cstr, &serverAddress.sin_addr.s_addr);
        serverAddress.sin_port = htons(port);

        int rc = connect(sock, (struct sockaddr *)&serverAddress, sizeof(struct
sockaddr_in));
```

```

const char *data = request.c_str();
rc = send(sock, data, strlen(data), 0);

rc = recv(sock, buf.data(), buf.size(), 0);

rc = close(sock);

```

```

String response = utils::vec_of_char_to_str(buf);
response.trim();
return response;
}

```

```

String form_http_request(const String&& method, const String&& path, const
String&& host, const String&& body, const String&& token)
{
    String request = method + " " + path + " HTTP/1.1\r\n" + "Host: " + host + "\r\n";
    request += "Connection: close\r\n";

    if(token != "")
    {
        request = request + "Authorization: Bearer " + token + "\r\n";
    }

    if(body != "")
    {
        const char* body_cstr = body.c_str();
        request = request + "Content-Type: application/json\r\n" + "Content-Length: " +
strlen(body_cstr) + "\r\n";
        request = request + "\r\n" + body + "\r\n";
    }
}

```

```

    }
    request += "\r\n";
    return request;
}

```

```

String form_meter_data_json(const float power, const float voltage)
{
    const int capacity = JSON_ARRAY_SIZE(1) + JSON_OBJECT_SIZE(3);
    StaticJsonDocument<capacity> doc;
    doc[0]["power"] = power;
    doc[0]["voltage"] = voltage;
    doc[0]["line"] = 1;
    String output;
    serializeJson(doc, output);
    return output;
}

```

```

String form_auth_data_json(const String&& id, const String&& password)
{
    const int capacity = JSON_OBJECT_SIZE(2);
    StaticJsonDocument<capacity> doc;
    doc["id"] = id.c_str();
    doc["password"] = password.c_str();
    String output;
    serializeJson(doc, output);
    return output;
}

```

```

String form_register_data_json(const String password, const String mac)

```

```

{
    const int capacity = JSON_OBJECT_SIZE(2);
    StaticJsonDocument<capacity> doc;
    doc["password"] = password.c_str();
    doc["serialNumber"] = mac.c_str();
    String output;
    serializeJson(doc, output);
    return output;
}

```

```

String host_to_ip(const String& host)
{
    hostent* hostname = gethostbyname(host.c_str());
    if(hostname)
        return String(inet_ntoa(*(in_addr**)hostname->h_addr_list));
    return {};
}

```

```

String send_data_https(const String&& host, const short port, const String&& request,
const char* ca_cert)
{
    // client.setCACert(ca_cert);
    client.setInsecure();
    const char* server = host.c_str();
    auto conn = client.connect(server, port);
    String response;
    if(conn == 1)
    {
        client.print(request);

```

```

while (client.available() == 0); //wait for response

while(client.available())
{
    char c = client.read();
    response += c;
}

client.stop();
}
else
{
    Serial.print("failed to connect");
    client.stop();
}

return response;
}

void server_register()
{
    preferences.end();
    preferences.begin("server_access", false);
    // Serial.println(preferences.getString("password"));
    String register_data = form_register_data_json(preferences.getString("password"),
    consts::mac);
    // Serial.print(F("formed json"));
    // Serial.println(register_data);

```

```

    String request = form_http_request(std::move(consts::POST),
std::move(consts::paths::registration),
    std::move(consts::host), std::move(register_data));
//    Serial.println(F("formed request"));
    String response = send_data_https(std::move(consts::host), consts::port_tls,
std::move(request), consts::ca_cert);
//    Serial.println(F("sent data https"));
    String response_status = http_parse::parse_response_status(response);
//    Serial.println(F("parsed response status"));
    if(response_status == consts::status_created)
    {
        int content_length = http_parse::parse_response_length(response);
//        Serial.println("parsed response length");
        if (content_length != -1)
        {
            String response_body = http_parse::parse_response_body(response,
content_length);
            request.clear();
            response.clear();
            response_status.clear();
            register_data.clear();
//            Serial.println("parsed response body");
//            Serial.println(response_body);

            String output;
            utils::find_value_in_json(response_body, "pairUrl", output);
            preferences.putString("pair_url", output);
//            Serial.println(preferences.getString("pair_url"));
            utils::find_value_in_json(response_body, "token", output);

```

```

        preferences.putString("token", output);
//    Serial.println(preferences.getString("token"));
        utils::find_value_in_json(response_body, "id", output);
        preferences.putString("id", output);
//    Serial.println(preferences.getString("id"));
    }
}
}

void get_new_token()
{
    String request_body = form_auth_data_json(preferences.getString("id"),
preferences.getString("password"));
    String path = consts::paths::auth;
    String request = server_request::form_http_request
    (
        std::move(consts::POST),
        std::move(path),
        std::move(consts::host),
        std::move(request_body)
    );
    String response = send_data_https(std::move(consts::host), consts::port_tls,
std::move(request), consts::ca_cert);
    String response_status = http_parse::parse_response_status(response);
    if(response_status == consts::status_ok)
    {
        int content_length = http_parse::parse_response_length(response);
        if (content_length != -1)
        {

```

```
String response_body = http_parse::parse_response_body(response,
content_length);
String token;
utils::find_value_in_json(response_body, "token", token);
preferences.putString("token", token);
}
}
}

} // namespace server_request
```

wifi_handle.cpp

```
#include <WiFi.h>

#include <AsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include "wifi_handle.h"

#include "data_collection.h"

#include "server_request.h"

namespace wifi_handle

{

    AsyncWebServer* server = new AsyncWebServer(80);

    const unsigned short wifi_timeout = 10000;

    String client_wifi_ssid = "";

    String client_wifi_password = "";

    std::vector<String> ssids_array;

    bool webserver_active = false;

    const char* PARAM_INPUT_1 = "output";

    const char* PARAM_INPUT_2 = "state";

    const char index_html[] PROGMEM = R"rawliteral(

        <!DOCTYPE HTML>

        <html>

        <head>

            <title>ESP Web Server</title>

            <meta name="viewport" content="width=device-width, initial-scale=1">

            <link rel="icon" href="data:,">

            <style>

                html { font-family: Arial; display: inline-block; text-align: center; }
```

```

    h2 {font-size: 3.0rem;}
    p {font-size: 3.0rem;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
</style>
</head>
<body>
<h2>ESP Web Server</h2>
%wifi_ssids%
<div style="margin-top:10px">
    <span>
        <text>SSID:</text>
        <input type="text" id="wifi_ssid" name="wifiname" maxlength="64"/>
    </span>
    <br><span>
        <text>PASS:</text>
        <input style="margin-top:10px;margin-right:3px" type="text" id="wifi_pass"
name="wifipass" maxlength="64"/>
    </span></br>
</div>
    <br><button style="border-radius: 5px; border-width: 1px; height: 25px; width:
80px" type="submit" id="buttonsend" name="buttonsend"
onclick="send_wifi_auth()">Submit</button></br>
    <h4 style="display: flex; align-self: flex-start; margin-left: 10px; margin-bottom:
10px">
        Add new sensor
    </h4>
    <div style="display: flex; align-content: flex-start; margin-left: 10px; justify-content:
flex-start">

```

```

        <text style="font-size:10pt; margin-right: 5px; vertical-align:
center">UUID:</text>

        <input type="text" id="sensor_uuid" name="sensor_uuid" maxlength="64"/>

        <button style="margin-left:10px; width: 60px; border-radius: 5px; border-width:
1px;" type="submit" id="sensor_add" name="sensor_add"
onclick="add_sensor()">Add</button>

    </div>

    <div style="display: flex; align-content: center; justify-content: center; margin-
top:25px">

        <button style="border-radius: 25px; border-width: 1px; height: 50px; width:
120px; background-color: #b3d4fc" type="submit" id="done_button"
name="done_button" onclick=process_done()>Done</button>

    </div>

    <script>

        function send_wifi_auth()
        {
            var xhr = new XMLHttpRequest();
            var params = "ssid=" + document.getElementById('wifi_ssid').value;
            params += "&";
            params += "password=" + document.getElementById('wifi_pass').value;
            xhr.open("POST", "/wifi?" + params, true);
            xhr.send();
        }

        function add_sensor()
        {
            var xhr = new XMLHttpRequest();
            var params = "uuid=" + document.getElementById('sensor_uuid').value;
            xhr.open("POST", "/sensors?" + params, true);

```

```

        xhr.send();
    }

    function process_done()
    {
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "/done", true);
        xhr.send();
    }
</script>
</body>
</html>
)rawliteral";

String main_page(const String& var){
    String response = "<h4>List of nearby wifi networks</h4>";
    if(var == "wifi_ssids"){
        for(auto s: ssids_array)
        {
            response += "<br><text style=\"color:coral\">";
            response += s;
            response += "</text></br>";
        }
        String link = server_request::preferences.getString("pair_url");
        Serial.print("link:");
        Serial.println(link);
        if(link != "")
        {
            response += "<h5>Your pair link:";

```

```

        response += link;
        response += "</h5>";
    }
}
return response;
}

```

```

void init_wifi()
{
    const char* ssid    = "ESP32-Access-Point";
    const char* password = "123456789";
    scan_wifi_networks(ssids_array);

```

```

    WiFi.softAP(ssid, password);

```

```

    Serial.println(WiFi.softAPIP());

```

```

server->on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, main_page);
});

```

```

server->on("/wifi", HTTP_POST, [] (AsyncWebServerRequest *request) {

    if (request->hasParam("ssid") && request->hasParam("password"))
    {
        client_wifi_ssid = request->getParam("ssid")->value();
        client_wifi_password = request->getParam("password")->value();
    }
    Serial.print("SSID: ");

```

```

Serial.println(client_wifi_ssid);
Serial.print("PASSWORD: ");
Serial.println(client_wifi_password);
request->send(200, "text/plain", "OK");
});

server->on("/sensors", HTTP_POST, [](AsyncWebServerRequest *request){
    if (request->hasParam("uuid"))
    {
        data_collection::sensor_uuids.push_back(request->getParam("uuid")->value());
        data_collection::mqtt_subscribe();
    }
    request->send_P(200, "text/plain", "OK");
});

server->on("/done", HTTP_POST, [&](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", "OK");
    WiFi.mode(WIFI_STA);
    unsigned int timestamp = time(0);
    WiFi.begin(client_wifi_ssid, client_wifi_password);
    while(WiFi.status() != WL_CONNECTED && (time(0)-timestamp) < 10)
    {
        delay(1000);
    }
    if(WiFi.status() == WL_CONNECTED)
    {
        Serial.print("Connected! : ");
        Serial.println(WiFi.localIP());
    }
}

```

```

else
{
    Serial.println("Couldn't connect to wifi, opening access point...");
    WiFi.softAP(ssid, password);
}
});

// IPAddress IP = WiFi.softAPIP();
// Serial.print("AP IP address: ");
// Serial.println(IP);
// ssids_array = scan_wifi_networks();

server->begin();

webserver_active = true;
}

std::vector<String> scan_wifi_networks(std::vector<String>& ssids_array)
{
    WiFi.mode(WIFI_STA);
    int networks = WiFi.scanNetworks(); //returns number of networks found

    for (unsigned i = 0; i < networks; i++) {
        ssids_array.push_back(WiFi.SSID(i));
    }
    WiFi.disconnect();
    return ssids_array;
}

```

```
bool is_connected()
{
    return WiFi.status() == WL_CONNECTED;
}
} // namespace wifi_handle
```

data_collection.cpp

```
#include "data_collection.h"
#include "utils.h"
#include <WiFi.h>
#include "consts.h"

namespace data_collection
{
    std::vector<String> sensor_uuids;
    std::vector<float>* voltage = new std::vector<float>();
    std::map<String, std::vector<float>> sensors_data;
    float voltage_previous = 220.0;
    volatile float power_med = 0;
    volatile float voltage_med = 0;
    WiFiClient wifiClient;
    MqttClient mqttClient(wifiClient);

    void data_fetch_sim()
    {
        float voltage_num = voltage_previous + utils::generate_number(-5.0f, 5.0f);
        voltage->push_back(voltage_num);
        voltage_previous = voltage_num;

        if(!mqttClient.connected())
        {
            Serial.print(F("mqtt_init(): "));
            Serial.println(mqtt_init())
        }
    }
}
```

```

    }

    if(mqttClient.connected())
    {
        mqttClient.poll();
    }

}

bool mqtt_init()
{
    mqttClient.setUsernamePassword(consts::mqtt_username, consts::mqtt_password);
    mqttClient.onMessage(onMqttMessage);
    if(mqttClient.connect(consts::broker, consts::mqtt_port))
        return true;
    return false;
}

void mqtt_subscribe()
{
    for(String s: sensor_uuids)
    {
        String topic = "sensors_data/";
        topic += s;
        mqttClient.subscribe(topic);
    }
}

void onMqttMessage(int messageSize)

```

```

{
  if(!mqttClient.messageDup())
  {
    unsigned char* buf = new unsigned char[messageSize];
    mqttClient.read(buf, messageSize);
    buf[messageSize] = '\0';
    String msg (reinterpret_cast<char*>(buf));
    String uuid, value;
    utils::find_value_in_json(msg, "uuid", uuid);
    utils::find_value_in_json(msg, "value", value);
    sensors_data[uuid].push_back(value.toFloat());
    if(value.toFloat() < consts::lower_barrier)
    {
      String topic = "off/";
      topic += uuid;
      mqttClient.beginMessage(topic);
      mqttClient.print("{\"msg\":\"off\"}");
      mqttClient.endMessage();
    }
  }
}

```

```

} // namespace data_collection

```

ДОДАТОК Б Презентація роботи



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Кафедра Інженерії програмного забезпечення в енергетиці.



Система прогнозування та оптимізації споживання енергоресурсів EnMonCo

Виконав: студент магістерського рівня 2-го року навчання, групи
ТВ-21мп

Нестеров Микита Андрійович

Керівник: д.т.н., доцент, професор кафедри ІПЗЕ Федорова Н.В.



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

АКТУАЛЬНІСТЬ РОБОТИ

- Актуальність роботи обґрунтовується стрімким розвитком IoT технологій та систем “розумного будинку”, що свідчить про набуття особливої актуальності у світі питання ефективного використання енергоресурсів.
- Зростання промислового виробництва, поширення інформаційних технологій, збільшення кількості об'єктів громадської інфраструктури супроводжується посиленням потреби в енергоресурсах.
- Глобальні проблеми, такі як зміна клімату, вимагають від суспільства більш відповідального ставлення до споживання енергії.



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

МЕТА РОБОТИ

- Метою дипломної роботи було проведення дослідження моделей та методів розробки IoT систем і розробка власного IoT застосунку "EnMonCo".
- Об'єктом дослідження були моделі, методи та процес розробки IoT застосунків.
- Предметом дослідження був аналіз існуючих моделей та методів розробки IoT систем, їхніх переваг, недоліків, актуальності та вибір найкращих з них для розробки IoT застосунку "EnMonCo" для розумного будинку.



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ЗАВДАННЯ РОБОТИ

- Провести аналіз існуючих моделей та методів розробки IoT систем.
- Провести аналіз існуючих систем обліку електроенергії.
- Розробка алгоритму програмної системи. Формування архітектури програмної системи, створення базових модулів та їх підключення до системи.



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

МЕТОДИ ТА МОДЕЛІ РОЗРОБКИ СИСТЕМ ПРОНОЗУВАННЯ

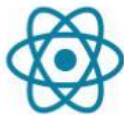
- Нейронні мережі
- Машинне навчання
- Глибоке навчання
- Time Series Analysis
- Метод екстраполяції
- Регресійний аналіз
- Моделі змішаного типу



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

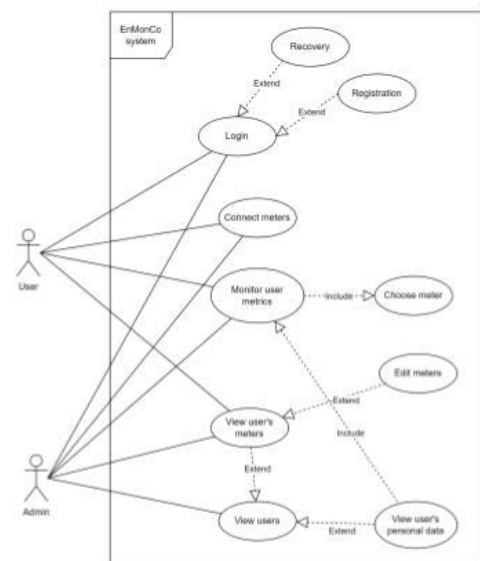
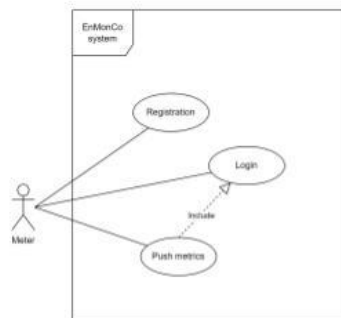
ЗАСОБИ РОЗРОБКИ

- HTML, CSS, Javascript, React, Material UI.
- Docker.
- C++, Arduino.
- Бібліотеки Paho, Boost, ESP-IDF
- ESP32
- Система контролю версій Git.



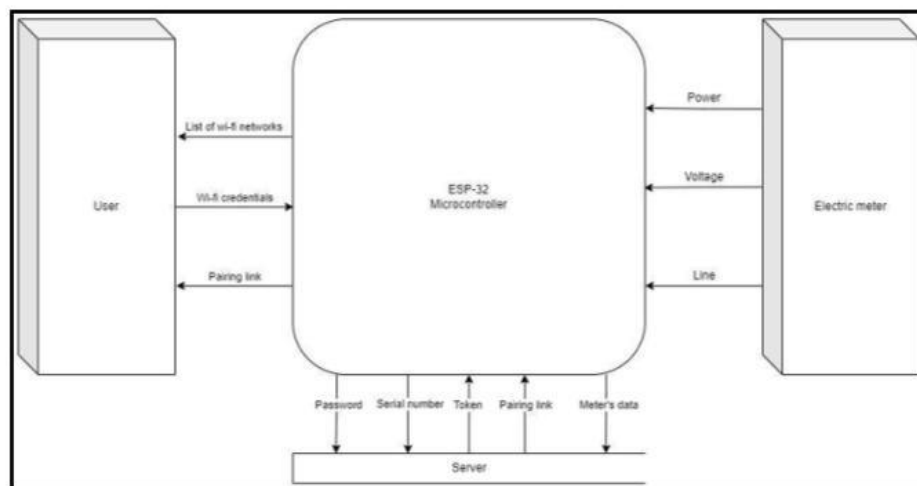
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ФУНКЦІОНАЛЬНІСТЬ СИСТЕМИ. ДІАГРАМА ПРЕЦЕДЕНТІВ



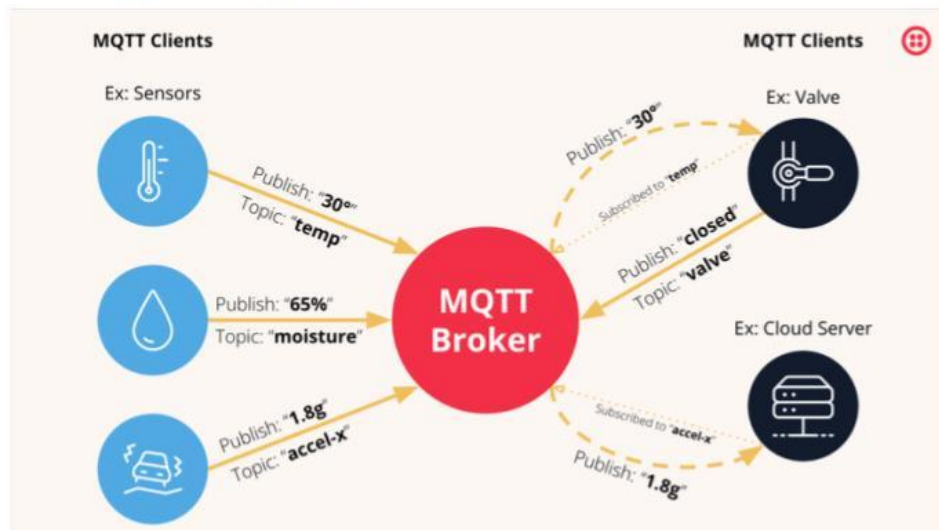
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ДІАГРАМА ПОТОКІВ ДАНИХ ПРОЕКТУ



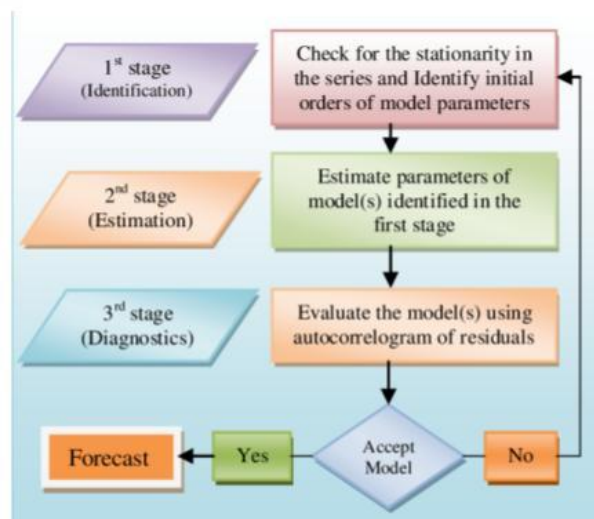
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ОСНОВНА ІДЕЯ MQTT у IoT



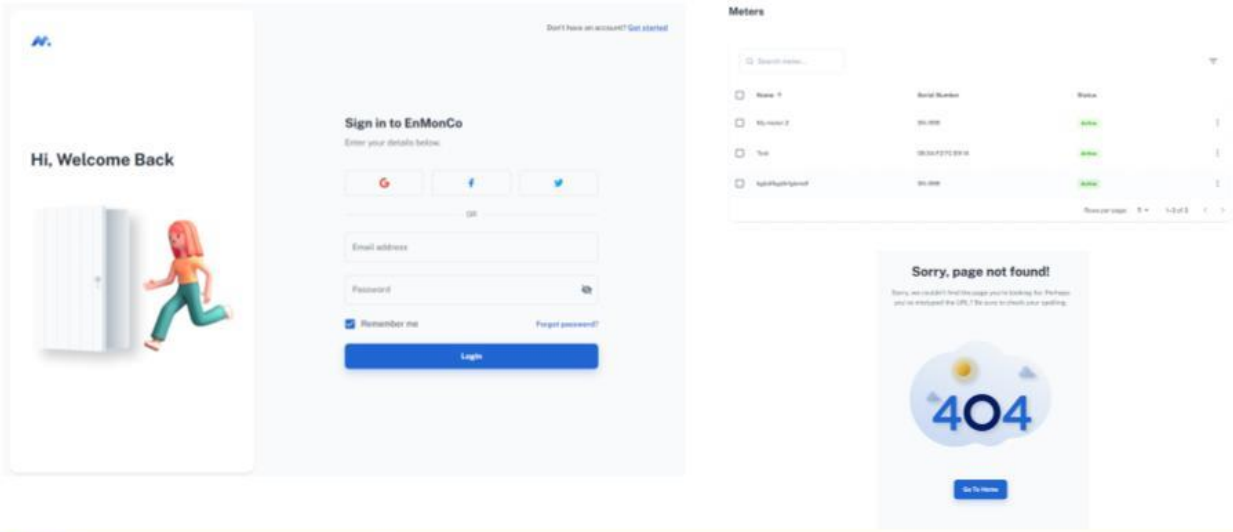
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ВИКОРИСТАННЯ ARIMA МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ

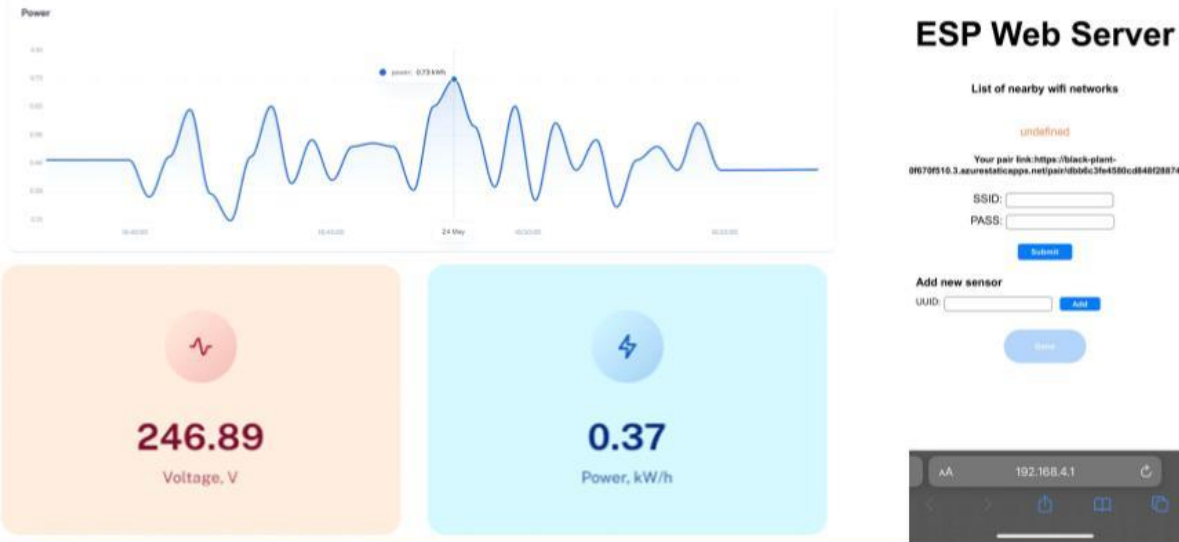


Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ІНТЕРФЕЙС ЗАСТОСУНКУ



ІНТЕРФЕЙС ЗАСТОСУНКУ



ВИСНОВКИ

- Проаналізовано ринок розумних лічильників електроенергії;
- Спроековано, розроблено, протестовано та завершено першу версію продукту EnMonCo;
- Розроблено модулі взаємодії користувача з апаратною системою через смартфон;
- Налаштовано комунікацію із сервером для відображення статистики по енергоспоживанню та значень споживання у реальному часі. Реалізовано прогнозування споживання електроенергії за допомогою ARIMA моделі.



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"