МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

Розроблення програмного забезпечення для системи електропостачання з фотоелектричними панелями

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-1-23
Кравченко М.С.

Науковий керівник: д. фіз.-мат. н., професор
Краснитський С.М.

Рецензент: к.т.н., доцент
Астістова Т.І.

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувачка кафедри КНТ

_____ Наталія ЧУПРИНКА

«_____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Кравченку Миколі Сергійовичу

1. Тема кваліфікаційної роботи: Розроблення програмного забезпечення для системи електропостачання з фотоелектричними панелями,
науковий керівник роботи Краснитський С. М., д.фіз.-мат. н. професор,
затверджені наказом КНУТД від 3" вересня 2024 року № 118-уч.
2. Строк подання студентом кваліфікаційної роботи 22.11.2024
3. Вихідні дані до кваліфікаційної роботи: Розробка кафедри комп'ютерних наук, рекомендована література, додатки.
4. Зміст кваліфікаційної роботи: Вступ, Розділ 1. Огляд існуючих засобів; Розділ 2. Вибір засобів розробки; Розділ 3. Проєктування та тестування програмного додатку; Додатки.

6. Дата видачі завдання 03 .09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів	Терміни виконання етапів	Примітка про виконання
1	Вступ	03.10.2024	
2	Розділ 1. Огляд існуючих засобів	08.10.2024	
3	Розділ 2. Вибір засобів розробки	15.10.2024	
4	Розділ 3 Проєктування та тестування програмного додатку	28.10.2024	
5	Висновки	30.10.2024	
6	Оформлення кваліфікаційної роботи	01.11.2024	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку	12.11.2024	
8	Подача кваліфікаційної роботи для рецензування	14.11.2024	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату	17.11.2024	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри	20.11.2024	

Студент _____ Микола КРАВЧЕНКО

Науковий керівник _____ Сергій КРАСНИТСЬКИЙ

АНОТАЦІЯ

Кравченко М С.: Розроблення програмного забезпечення для системи електропостачання з фотоелектричними панелями

Кваліфікаційна робота за спеціальністю 122 – «Комп'ютерні науки» , Київський національний університет технологій та дизайну, Київ, 2024 рік.

Кваліфікаційна робота присвячена розробки програмного засобу для збирання, зберігання та обробки даних для ефективного функціонування системи електропостачання з фотоелектричними панелями та оптимізації функціонування фотоелектричної системи на основі технічного критерію, що грунтується на оцінці імовірності відключення навантаження та імовірності втрати потужності та розробки многорівневої архітектури програмного додатку.

Програмний продукт дає можливість забезпечити необхідними даними та методами для виконання оптимального функціонування фотоелектричної системи на основі технічного критерію оптимізації, яке здійснюється в результаті моніторингу та аналізу її функціонування**,** виконувати прогноз очікуваної потужності на основі метеорологічних параметрів, використовуючи кластеризацію та правила асоціації. В роботі проведено аналіз найкращих програмних засобів для проектування фотоелектричних систем у 2024 року, аналіз надійності енергозабезпечення в фотоелектричній системі, розроблена блок-схема імітаційного моделювання ФЕП.

Інструментів для реалізації було обрано мову програмування Python, середовище розробки VS Code, БД PostgreSQL, фреймворки Express та Qt, бібліотека D3.js, стандарт ECMAScript, шаблонізатор EJS, менеджер пакетів npm, метамова для CSS (SASS), HTML, середовище розробки Visual Studio Code (VS Code).

Ключові слова: інтелектуальна електрична система, фотоелектричний перетворювач, JSON HTTP HTML, CSS,Python, MATLAB, JSON, HTTP, PostgreSQL, Express, бібліотека D3.js

ANNOTATION

Kravchenko M S.: Software development for a power supply system with photovoltaic panels.

Qualification work in specialty 122 - "Computer Science", Kyiv National University of Technologies and Design, Kyiv, 2024.

The qualification work is devoted to the development of a software tool for collecting, storing and processing data for the effective functioning of a power supply system with photovoltaic panels and optimizing the functioning of the photovoltaic system based on a technical criterion based on the assessment of the probability of load disconnection and the probability of power loss and the development of a multi-level architecture of the software applicationPurpose of the qualification work

The software product makes it possible to provide the necessary data and methods for optimal functioning of a photovoltaic system based on a technical optimization criterion, which is carried out as a result of monitoring and analyzing its functioning, to forecast the expected power based on meteorological parameters, using clustering and association rules. The paper analyzes the best software tools for designing photovoltaic systems in 2024, analyzes the reliability of energy supply in a photovoltaic system, and develops a block diagram of simulation modeling of the photovoltaic system.

The tools chosen for implementation were the Python programming language, VS Code development environment, PostgreSQL database, Express and Qt frameworks, D3.js library, ECMAScript standard, EJS templating tool, npm package manager, CSS metalanguage (SASS), HTML, Visual Studio Code development environment (VS Code).

Keywords: intelligent electrical system, photovoltaic converter, JSON HTTP HTML, CSS, Python, MATLAB, JSON, HTTP, PostgreSQL, Express, D3.js library

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

API – *Application Programming Interface* (прикладний програмний інтерфейс)

CAD – Computer-Aided Design (комп'ютерне автоматизоване проектування)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

D3.js – *Data-Driven Documents* (дані, орієнтовані на документи)

EJS – *Embedded JavaScript* (вбудований *JS*)

EMS – *Energy management system* (енергоменеджмент)

GUI – *Graphical User Interface* (графічний інтерфейс користувача)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

HTML – Hypertext Markup Language (мова розмітки гіпертексту)

IoT – *Internet of Things* (Інтернет речей)

JS – JavaScript.

JSON – JavaScript Object Notation (формат обміну даними)

Middleware – проміжне ПЗ.

npm – *Node Package Manager* (менеджер пакетів *Node.js*).

PV – Photovoltaic (фотоелектричний).

PVGIS – *Photovoltaic Geographical Information System* (система географічної інформації для ФЕП).

REST – *Representational State Transfer* (передача представлення стану).

SASS – *Syntactically Awesome Style Sheets* (скриптова метамова, яка інтерпретується в каскадні таблиці стилів)

SCADA – *Supervisory Control and Data Acquisition* (забезпечення роботи в реальному часі систем збору, обробки, відображення та архівування інформації про об'єкт моніторингу або управління)

SemVer – *Semantic Versioning* (семантичне версіонування)

SQL – *Structured Query Language* (мова структурованих запитів)

STC – *Standard Test Condition* (стандартні умови тестування)

VS – *CodeVisual Studio Code*

WEB – *World Wide Web* (всесвітня павутина)

XML – *Extensible Markup Language* (мова розмітки)

URL – *Uniform Resource Locator* (уніфікований локатор ресурсу)

БД – База Даних.

ВАХ – вольт-амперна характеристика.

ВВХ – вольт-ватна характеристика.

ІЕМ – інтелектуальні електричні мережі.

ІЕС – інтелектуальні електричні системи.

МКЕ – метод кінцевих елементів.

ПЗ – програмне забезпечення.

ФЕП – фотоелектрична панель.

ШІ – штучний інтелект.

ЗМІСТ

ВСТУП

Актуальність теми. Стрімке зменшення ресурсів викопного палива та зростаюча очевидність настання глобального потепління обумовлюють необхідність термінового пошуку альтернативних джерел енергії. Останні дослідження показують, що відновлювана енергетика має великий потенціал і може бути використана для задоволення світового попиту на енергію. Згідно даних, за останні десятиліття фотоелектрична індустрія зросла більш ніж на 40% на рік завдяки швидкому зниженню вартості фотоелектричних технологій. Ці технології можуть стати основним альтернативним джерелом енергії в майбутньому, оскільки вони мають ряд позитивних властивостей, низькі експлуатаційні потреби, безкоштовне та невичерпне джерело енергії, а також надійні та довговічні компоненти системи. Однак сонячна енергетика не завжди є надійною, оскільки сонячне випромінювання, що надходить до фотопанелей, постійно змінюється через непередбачуваний характер, який обумовлений залежністю від погодних умов. Саме тому вироблена фотопанелями потужність не завжди відповідає потребам у навантаженні. Географічне розташування, температура навколишнього середовища, коефіцієнт хмарності, нахил та орієнтація фотоелектричної панелі є основними факторами, які впливають на виробництво панеллю електричної енергії. Виходячи з цього, вивчення даних метеорологічних умов є надзвичайно важливим при попередньому проектуванні фотоелектричної системи.

Існує два типи фотоелектричних енергосистем, а саме мережеві та автономні фотоелектричні системи. У мережевій фотоелектричній системі інвертор перетворює електроенергію постійного струму, що виробляється фотоелектричними модулями, на електроенергію змінного струму, яка подається потім в загальну мережу. З іншого боку, автономні фотоелектричні системи – це системи в яких фотоелектричні модулі не підключені до загальної мережі, і вся вироблена потужність йде на потреби конкретного комунального господарства або підприємства. Перш ніж рекомендувати та встановлювати фотоелектричну систему, важливо переконатися, що в системі вироблена потужність не є занадто

надмірною або занадто недостатньою. При проектуванні такої системи виникає необхідність в ретельному дослідженні її функціонування. Для ефективного та економного використання сонячної енергії в фотоелектричній системі необхідно забезпечити належну розмірність системи, при якій її функціонування буде оптимальним з точки зору вартості придбаних елементів системи та надійності забезпечення відповідної потужності. Коли мова йде про розмір системи, споживання електроенергії в такій системі є дуже важливим параметром, оскільки перевиробництво електричних потужностей може негативно вплинути на доцільність самої системи. Тому бажаним є велике власне споживання, щоб забезпечити ефективне використання інвестицій на придбання та експлуатацію відповідного фотоелектричного масиву, який генерує необхідні потужності.

Оптимізація фотоелектричної системи означає підвищення ефективності, надійності та терміну її функціонування. Вона також полягає у зменшенні витрат на технічне обслуговування та залежності від загальної мережі. Оптимізація фотоелектричної системи дозволяє отримати максимальну віддачу від інвестицій, що вкладаються при встановленні на визначеній ділянці місцевості фотоелектричної системи.

Використання програмного забезпечення при проектуванні фотоелектричної системи надає можливість виконувати таке проектування швидше та ефективніше. Переваги, які при цьому надаються, полягають в: 1) забезпеченні більш точної імітації дизайну фотоелектричної системи; 2) зменшенні проміжку часу, що потрібен для проектування системи; 3) зменшенні кошторису та матеріальних витрат; 4) забезпеченні якісної візуалізації дизайну, що проектується.

Об’єкт дослідження – процес збирання, зберігання та обробки даних для ефективного функціонування системи електропостачання з фотоелектричними панелями.

Предмет дослідження – програмний додаток для збирання, зберігання та обробки даних для ефективного функціонування системи електропостачання з фотоелектричними панелями.

Мета кваліфікаційної роботи – розробка програмного засобу для збирання, зберігання та обробки даних для ефективного функціонування системи електропостачання з фотоелектричними панелями..

Призначення цього програмного засобу полягає в забезпеченні необхідних даних та методів для виконання оптимального функціонування фотоелектричної системи на основі технічного критерію оптимізації. Оптимізація фотоелектричної системи здійснюється в результаті моніторингу та аналізу її функціонування.

Моніторинг фотоелектричної системи є важливим етапом в її оптимізації, оскільки він дозволяє відстежувати та вимірювати ключові параметри, такі як потужність, енергетична продуктивність, напруга, струм, температура та освітленість. Це, в свою чергу, допомагає виявляти та усувати проблеми, які пов'язані із затіненням, забрудненням поверхні, несправностями або пошкодженням фотоелектричних панелей. Крім того, моніторинг фотоелектричної системи дозволяє порівнювати фактичну продуктивність з очікуваною та відповідно налаштовувати систему. Отримані дані передаються на сервер або хмарну платформу, де зберігаються та проходять необхідну обробку. Аналіз фотоелектричної системи є наступним кроком до оптимізації, що дозволяє оцінити та вияснити, як функціонує фотоелектрична система та які фактори на неї впливають. Можна використовувати різні методи та підходи для аналізу фотоелектричної системи, такі як відношення фактичного виходу енергії до теоретичного виходу енергії фотоелектричної системи, кількість енергії, що виробляється фотоелектричною системою на одиницю встановленої потужності або на одиницю площі модуля протягом певного періоду часу, вимірювання відсотку втрат потужності фотоелектричної системи з часом через зношуваність компонентів. Аналіз фотоелектричної системи дозволяє оптимізувати її конструкцію, конфігурацію та експлуатацію для підвищення енергоефективності та економії. Оптимізація фотоелектричної системи – це завершальний крок до досягнення найкращих результатів. Це передбачає впровадження дій та прийняття рішень, що отримані з даних моніторингу та аналізу, що сприяє покращенню якості, функціональності та прибутковості системи.

План роботи передбачає детальний розгляд як теоретичних так і практичних аспектів для досягнення конкретної реалізації програмного засобу. В першому розділі проведений детальний огляд існуючих програмних засобів, описані їх переваги та на цій основі виявлені особливості програмного засобу, який розробляється. В другому розділі був виконаний відповідний вибір інструментів та методів, що використовуються для розробки програмного продукту. В третьому розділі описуються етапи розробки програмного засобу та результати його тестування.

Наукова новизна отриманих результатів визначається тим, що програмний засіб дозволяє: 1) за рахунок інтегрованості з MATLAB, створювати базу фотоелектричних панелей, які присутні на ринку, розраховуючи їх вольт-амперні характеристики на основі технічних параметрів, що надаються виробником; 2) виконувати прогноз очікуваної потужності на основі метеорологічних параметрів, використовуючи кластеризацію та правила асоціації; 3) виконувати оптимізацію функціонування системи електропостачання з фотоелектричними панелями на основі технічного критерію.

Практичне значення отриманих результатів. Отримані результати можуть знайти практичне застосування в області відновлювальних джерел енергії. Застосування програмного додатку підвищить економічну доцільність проектів, пов'язаних зі збором, зберіганням та аналізом даних, що використовуються в задачах проектування ефективних фотоелектричних систем. Крім того, програмний засіб може бути використаний для прогнозування погодних умов та в системах машинного навчання.

Особистий внесок випускника. Всі результати, представлені у кваліфікаційній роботі, отримані випускником особисто.

Апробація отриманих результатів. Практичні та теоретичні результати, що описані в кваліфікаційній роботі, проходили апробацію на IV Всеукраїнська конференція здобувачів вищої освіти і молодих учених “Інноватика в освіті, науці та бізнесі: виклики та можливості” та VIII Міжнародній науково-практичній конференції «Мехатронні системи: інновації та інжиніринг» – «MSIE-2024».

Публікації.

M.S. Kravchenko,T.I. Astistova, O.P. Kravchenko Processing data module in monitoring environment parameters for electrical power generation / IV Всеукраїнська конференція здобувачів вищої освіти і молодих учених “Інноватика в освіті, науці та бізнесі: виклики та можливості”, КНУТД м.Київ,17 листопада 2023 р.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ ЗАСОБІВ

1.1.Програмні засоби імітаційного моделювання фотоелектричних систем

Програмне забезпечення для проектування фотоелектричних систем – це спеціалізовані програмні продукти, які використовуються інженерами, архітекторами та фахівцями в галузі сонячної енергетики для проектування, планування та оптимізації сонячних фотоелектричних систем. При правильному використанні ці засоби дозволяють моделювати різні сценарії, розраховувати виробництво енергії та прогнозувати потенційну економію, що робить їх важливими інструментами для проектування бажаних фотоелектричних систем. Хоча кожне програмне забезпечення для проектування таких систем є різним за своїм функціоналом, більшість із них включають такі функції, як 3D-моделювання, інтеграцію даних про погодні умови та величину сонячного випромінювання, а також можливість для проектування електричної системи. Як вже згадувалося у вступній частині використання програмного забезпечення для проектування фотоелектричних систем значно підвищує ефективність та функціональність таких систем, скорочує кошторис при їх впровадженні та зменшує матеріальні витрати.

На даний час існує безліч програмних засобів для аналізу, моделювання та проектування фотоелектричних систем. Більшість систем передбачають оцінювання величини сонячного випромінювання з урахуванням характеристик та розташування фотоелектричної системи, яка проектується. Програмне забезпечення для моделювання (імітаційне програмне забезпечення) значно прискорює загальний процес проектування фотоелектричних систем, порівняно з інтуїтивним підходом, де всі їх продуктивні характеристики оцінюються *in situ* з використанням рутинних методів обчислення.

Існує п'ять категорій такого програмного забезпечення, а саме (рис. 1.1):

1) програмні засоби для моделювання (*simulation tools*),
2) програмні засоби для економічної оцінки (*economic evaluation tools*),

3) програмні засоби для аналізу та планування (*analysis and planning tools*),
4) програмні засоби для аналізу ділянок місцевості (*site analysis tools*),
5) програмні засоби, що надають дані про сонячне випромінювання у вигляді відповідних карт (*solar radiation maps*).

Всі категорії, крім останньої, поділяються на дві підкатегорії, а саме програмне забезпечення, яке призначене для моделювання 1) виключно фотоелектричних систем та 2) гібридних електроенергетичних систем, що включають в себе інші відновлювальні джерела генерації, крім фотоелектричних модулів, наприклад вітрогенератори.

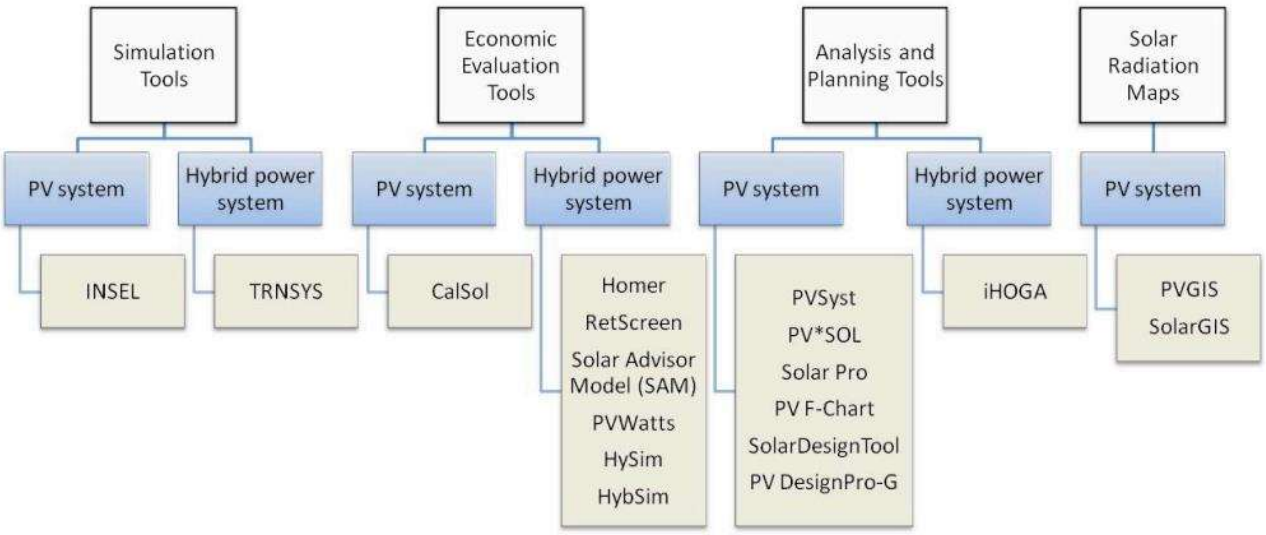


Рис 1.1. Класифікація імітаційного програмного забезпечення.

Отже, програмні засоби моделювання фотоелектричних систем представляють собою програмне забезпечення, направлене на моделювання або прогнозування роботи енергосистеми, з використанням зібраної метеорологічної бази даних. Такі програмні засоби були розроблені для моделювання, моніторингу, аналізу та візуалізації особливостей функціонування енергосистеми, однак ці засоби не спроможні виконувати оптимізацію функціонування фотоелектричної системи з точки зору генерації та споживання електричної потужності. Розглянемо ці засоби більш детальніше.

1.1.1. Програмні засоби для моделювання фотоелектричних систем

Першим програмним продуктом цієї категорії був і все ще залишається популярним, INSEL (*Integrated Simulation Environment Language*, мова інтегрованого імітаційного середовища), який є програмним засобом для моделювання і був розроблений Ольденбурзьким університетом (Німеччина). INSEL здатний створювати моделі систем та їх конфігурації для планування і контролю при роботі як з електричною, так і з тепловою системами. Це програмне забезпечення підходить для моделювання 1) величин сонячного випромінювання в визначеному часовому інтервалі, 2) фотоелектричної електростанції, 3) системи охолодження фотоелектричних модулів та 4) системи опалення. Користувач конструює запропоновану конфігурацію енергосистеми шляхом підключення блоків, наданих у бібліотеці. Перевага цього програмного забезпечення полягає в тому, що воно надає базу даних фотоелектричних компонентів, наприклад, фотоелектричних модулів, присутніх на ринку. Крім того, програмне забезпечення також здатне виявляти несправності в системі. Програмне забезпечення містить метеорологічну базу даних з 2000 ділянок по всьому світу, може генерувати погодинні значення освітленості, температури, вологості та швидкості вітру (рис. 1.2) [12].

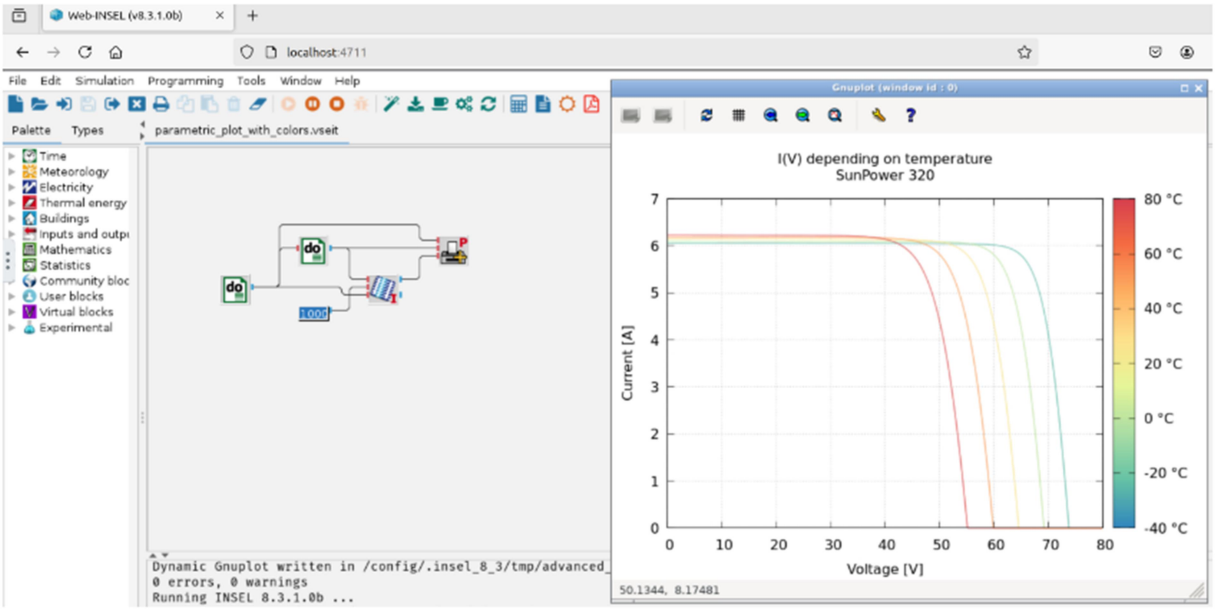


Рис. 1.2. Робоче вікно INSEL

Іншим доступним інструментом моделювання є TRNSYS (*Transient System Simulation*), який був розроблений Університетом Вісконсіна (США). У порівнянні з

INSEL, TRNSYS здатний моделювати гібридну електроенергетичну систему, включаючи, крім фотоелектричних модулів, також вітрогенератори, паливні елементи та акумулятори. Метою програмного забезпечення є моделювання електроенергетичної системи з низьким споживанням енергії, систем опалення, вентиляції та кондиціонування. Це програмне забезпечення є гнучким, де користувачі можуть модифікувати математичну модель у своїй бібліотеці. Крім того, метеорологічні дані та компоненти, які отримані або розроблені користувачами, можуть використовуватися як вхідні дані при моделюванні (рис. 1.3) [13].

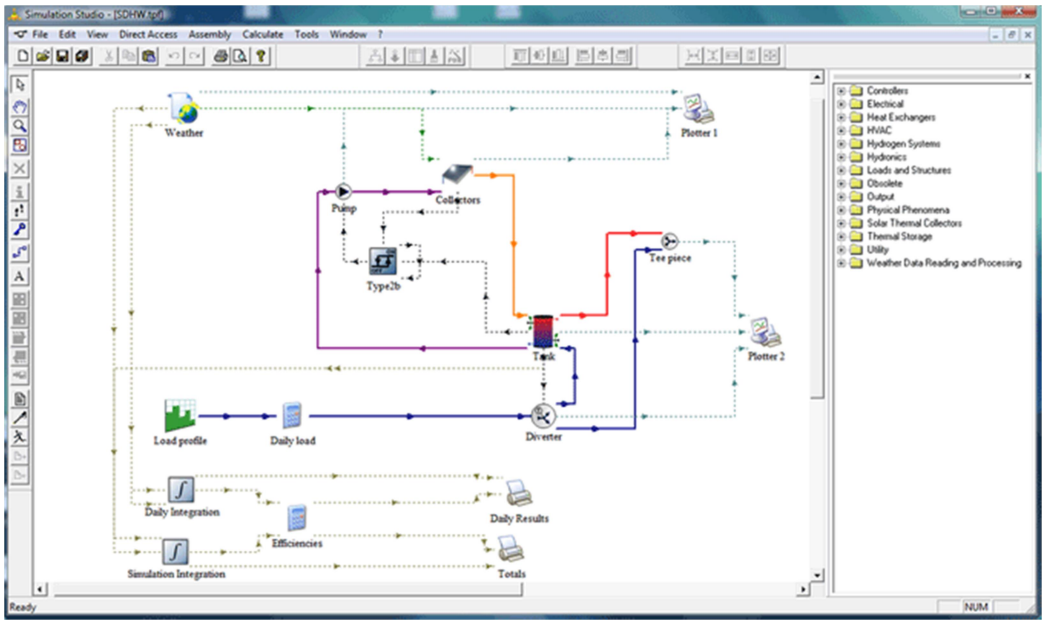


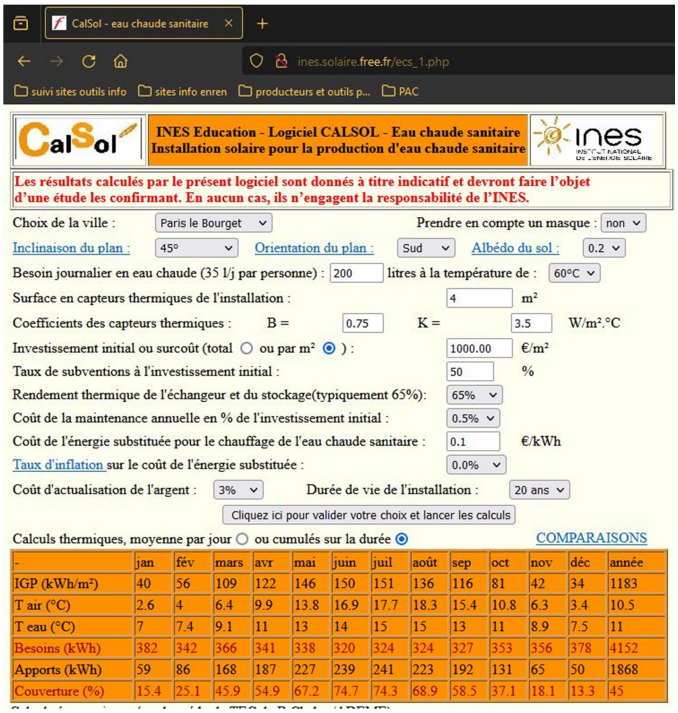
Рис. 1.3. Робоче вікно TRNSYS 18:

1.1.2. Програмні засоби для економічної оцінки

Програмне забезпечення, яке класифікується як інструменти економічної оцінки, здатне забезпечити економічний аналіз для запропонованої системи, що проектується. Щоб визначити, чи є система реальною або для того, щоб максимізувати чистий прибуток від споживання електроенергії, користувачеві потрібно ввести всі витрати та фінансові параметри як вхідні дані, а потім виконати аналіз, необхідний для мінімізації загальних витрат на проект.

Однією з популярних програм цього типу є CalSol, яка була розроблена Національним Інститутом *I'Energie Solaire* (INES, Франція). Це програмне

забезпечення здатне проводити економічний аналіз у мережевих та автономних системах. Слід зауважити, що, у цьому програмному продукті доступні лише французькі метеорологічні бази даних. Крім того, це програмне забезпечення не містить бази даних фотоелектричних компонентів та не може взаємодіяти з іншими програмним засобами. Це програмне забезпечення просте у використанні, підходить для попереднього проектування фотоелектричної системи, є безкоштовним та доступним через Інтернет (рис. 1.4) [14].



-	jan	fév	mars	avr	mai	juin	juil	août	sep	oct	nov	déc	année
IGP (kWh/m²)	40	56	109	122	146	150	151	136	116	81	42	34	1183
T air (°C)	2.6	4	6.4	9.9	13.8	16.9	17.7	18.3	15.4	10.8	6.3	3.4	10.5
T eau (°C)	7	7.4	9.1	11	13	14	15	15	13	11	8.9	7.5	11
Besoins (kWh)	382	342	366	341	338	320	324	324	327	353	356	378	4152
Apports (kWh)	59	86	168	187	227	239	241	223	192	131	65	50	1868
Couverture (%)	15.4	25.1	45.9	54.9	67.2	74.7	74.3	68.9	58.5	37.1	18.1	13.3	45

Рис. 1.4. Робоче вікно CalSol.

HOMER (*Hybrid Optimization Model for Electrical Renewable*, гібридна оптимізаційна модель для відновлювальної електроенергетики) – це програмний продукт, який був розроблений NREL (Національна лабораторія відновлюваної енергії, США). Програмне засіб підходить для проектування та аналізу гібридних електроенергетичних систем, яка містить звичайні генератори, вітрогенератори, фотоелектричні модулі, гідрогенератори, акумулятори та паливні елементи. HOMER здатен моделювати роботу енергосистеми протягом 8760 годин за рік, а результати представляти у вигляді різноманітних таблиць та графіків. Метеорологічні дані для визначеної ділянки, на якій знаходяться джерела генерації,

можуть бути імпортовані з веб-сайту HOMER energy або надаватися користувачами (рис. 1.5) [15].

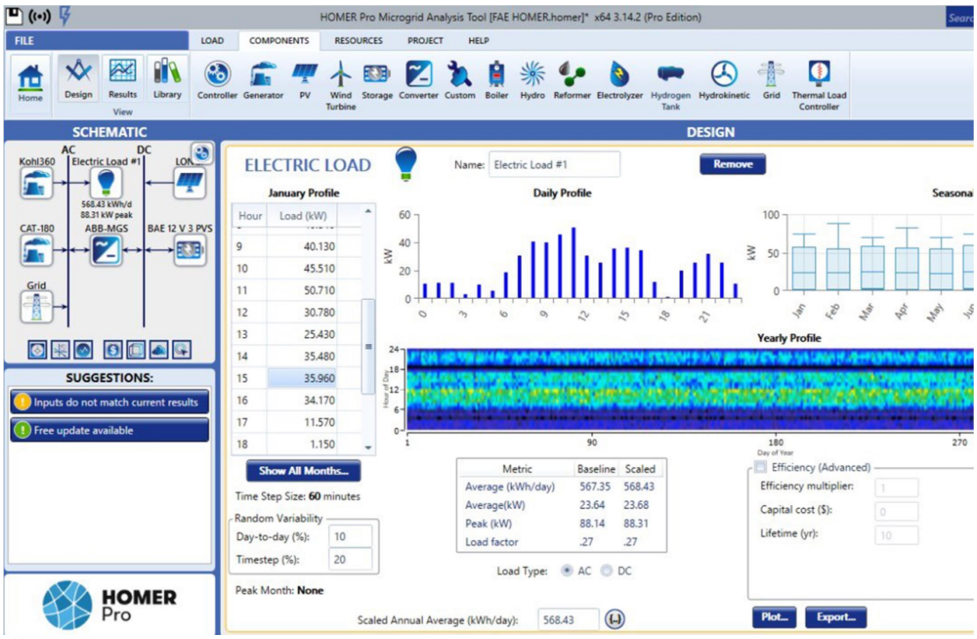


Рис. 1.5. Робоче вікно HOMER.

Програмний засіб RETScreen також підходить як інструмент техніко-економічного обґрунтування для гібридних технологій виробництва електроенергії і був розроблений Міністерством природних ресурсів Канади як інструмент для моделювання за допомогою електронних таблиць на основі Microsoft Excel. Ядром інструментарію є стандартизований та інтегрований аналіз проектів відновлювальної енергії, який використовується в усьому світі для оцінки виробництва енергії, вартості життєвого циклу та зменшення викидів парникових газів для різних типів відновлюваних енергоефективних технологій. Програмне забезпечення містить глобальну кліматичну базу даних, що генеруються 6000 наземними станціями, а також надає посилання на кліматичну базу даних NASA. RETScreen надає можливість аналізу енергетичного моделювання, витрат, викидів парникових газів, фінансового аналізу та оцінки ризиків її роботи. (рис. 1.6) [16].

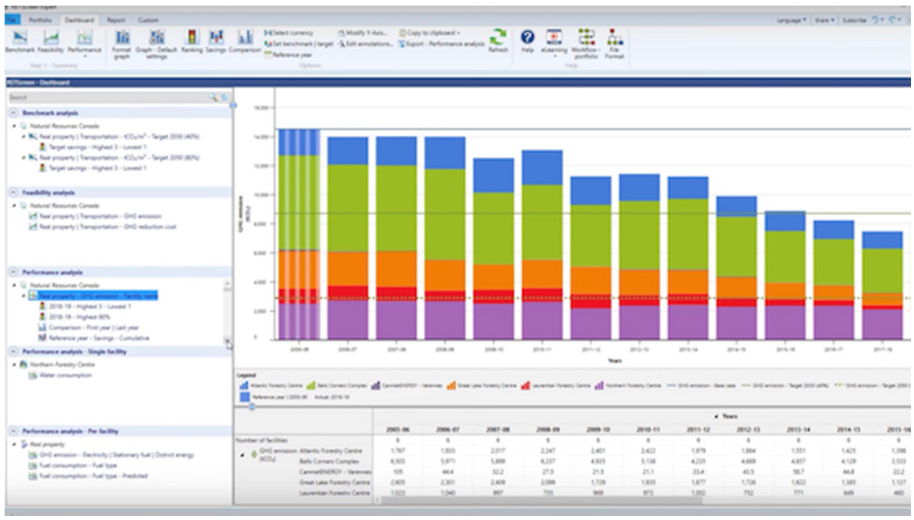


Рис. 1.6. Робоче вікно RETScreen.

SAM (*System Advisor Model*) — це вільне програмне забезпечення, розроблене Національною лабораторією відновлюваної енергетики (Вашингтон, США). Програмне забезпечення здатне аналізувати фотоелектричні технології та забезпечувати аналіз фінансування і витрат. Результати аналізу представляються у вигляді усередненої вартості енергії, вихідної потужності системи, пікової та річної ефективності системи, а також погодинного виробництва електроенергії системою у вигляді таблиць та графіків. SAM може автоматично завантажувати онлайн-базу даних, включаючи енергетичні ресурси для сонячної, вітрової, біопаливної та геотермальної енергетики (рис. 1.7) [17].

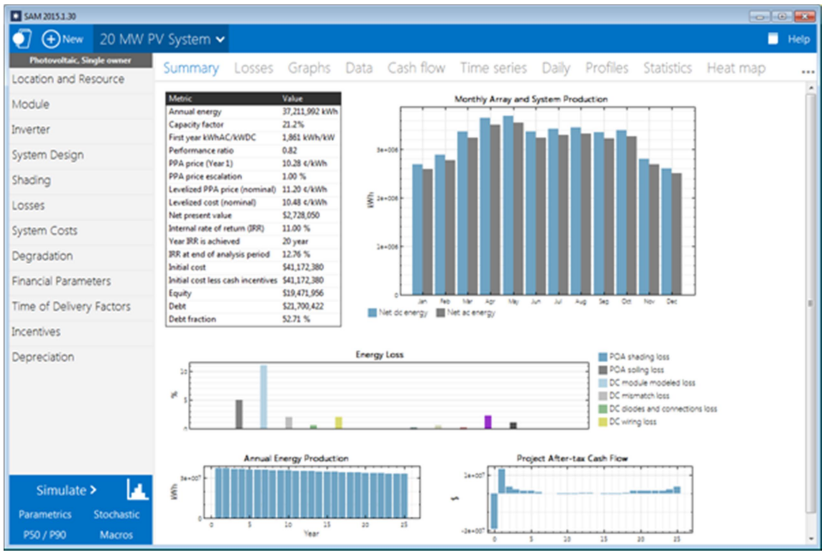


Рис. 1.7. Робоче вікно System Advisor Model.

PVWatts – це автоматизоване просте програмне забезпечення-калькулятор, яке дає швидку відповідь щодо очікуваного виробництва енергії та економії коштів у системі, що проектується, його можна використовувати через SAM. [18].

HybSim (Гібридна симуляція) — це симулятор гібридної енергії, розроблений Sandia National Laboratory (США), і підходить для моделювання відновлюваних джерел енергії, таких як фотоелектричні модулі, дизельні генератори та акумуляторні батареї в автономній системі. Він здатний виконувати фінансовий аналіз, а також порівняння витрат між різними конфігураціями. HybSim здатний взаємодіяти з іншим програмним забезпеченням, наприклад, даними про погоду та сонячне випромінювання [19].

1.1.3. Програмні засоби для планування та аналізу

Програмні засоби цієї категорії направлені на те, щоб допомогти користувачам у плануванні, проектуванні, визначенні розмірів системи та оптимізації джерел живлення, деякі з цих засобів надають базу даних фотоелектричних компонентів, доступних на ринку.

PVsyst є найбільш часто використовуваним програмним засобом для проектування фотоелектричних систем. PVsyst спроможний проектувати, моделювати та аналізувати як автономні фотоелектричні системи, так і системи, що підключені до мережі. Окрім бази даних Meteo, яка міститься в PVsyst, цей програмний засіб також може імпортувати метеорологічні дані з Інтернету. Програмне забезпечення також має базу даних фотоелектричних компонентів, які доступні на ринку. Однак PVsyst не допускає взаємозв'язку з іншими програмами (рис. 1.8) [20].

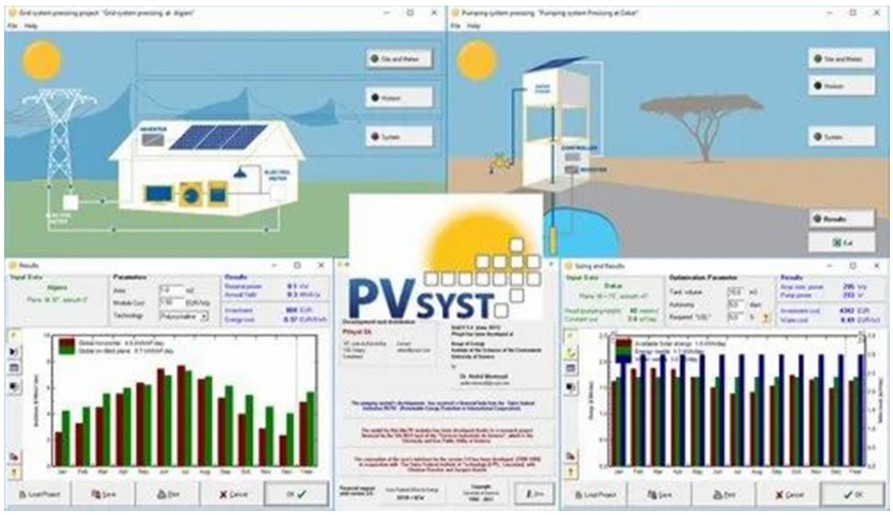


Рис. 1.8. Робоче вікно PVsyst.

PV*SOL Expert – це програмне забезпечення для 3D-проектування та аналізу функціонування фотоелектричних систем в яких використовуються покрівельні фотоелектричні модулі. Цей програмний засіб може аналізувати вплив затінення на енергетичні показники та виконувати оптимізацію конфігурації фотоелектричних модулів. PV*SOL направлений на швидке проектування та невеликий фінансовий аналіз. Отримані результати включають звіт про вихідну потужність, ефективність системи, втрати та економічний кошторис, взаємозв'язок з іншими програмами не забезпечений (рис. 1.9) [21].

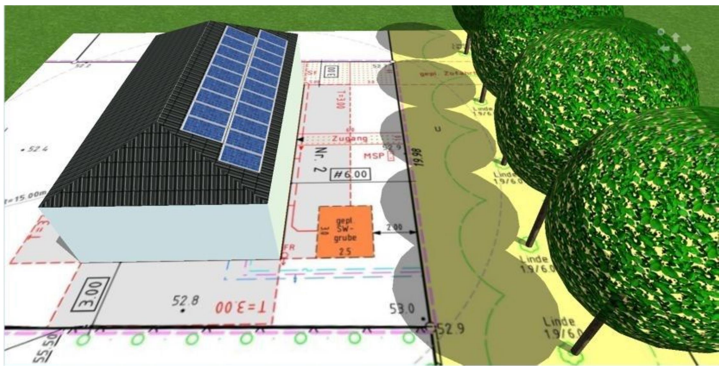


Рис. 1.9. Робоче вікно PV*SOL.

PV F-Chart здатний надавати щомісячні середні оцінки ефективності погодинно на протязі дня. База даних PV F-Chart складається з даних про погоду з 300 сайтів і може бути доповнена користувачами (рис. 1.10) [22].

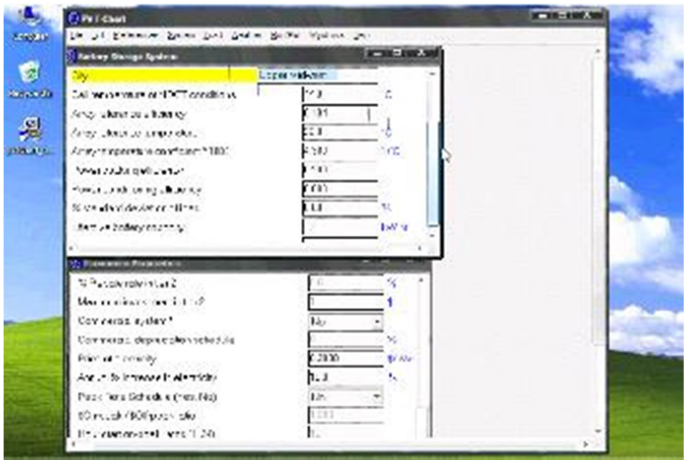


Рис. 1.10. Робоче вікно PV F-CHART.

1.1.4. Карти величин сонячного випромінювання

Карти сонячного випромінювання дозволяють користувачам охопити сонячні ресурси для кожної точки на Землі за допомогою простого візуального відображення. Для цього в Інтернеті доступні дві програми: PVGIS та SolarGIS. Обидві програми надають географічну інформацію на основі карт або уявних супутників. PVGIS– це онлайн-програмне забезпечення, яке оцінює показники сонячного випромінювання, формує карти на їх основі та розраховує річну енергію, яка виробляється фотоелектричною системою та є доступною в мережі Інтернет [23]. SolarGIS – це також безкоштовний, доступний через Інтернет програмний засіб, який оцінює показники сонячного випромінювання, на основі яких визначається величина електричної потужності, що генерується фотоелектричною системою. SolarGIS виконує моніторинг продуктивності системи в режимі реального часу та здійснює візуалізацію даних на картах. Всі дані в SolarGIS генеруються з використанням власних алгоритмів [24].

Крім описаних вище програмних засобів, що використовуються в проектуванні фотоелектричних систем слід зазначити також про наявність BlueSol [25] та Pylon [26], які поєднують в собі властивості декількох засобів.

BlueSol моделює поведінку фотоелектричної системи у всіх її компонентах. Схематичне зображення дозволяє надати проектувальнику точне уявлення про операції, які відбуваються в системі. BlueSol інтегрує систему CAD, яка автоматично генерує відповідну електричну схему (рис. 1.11).

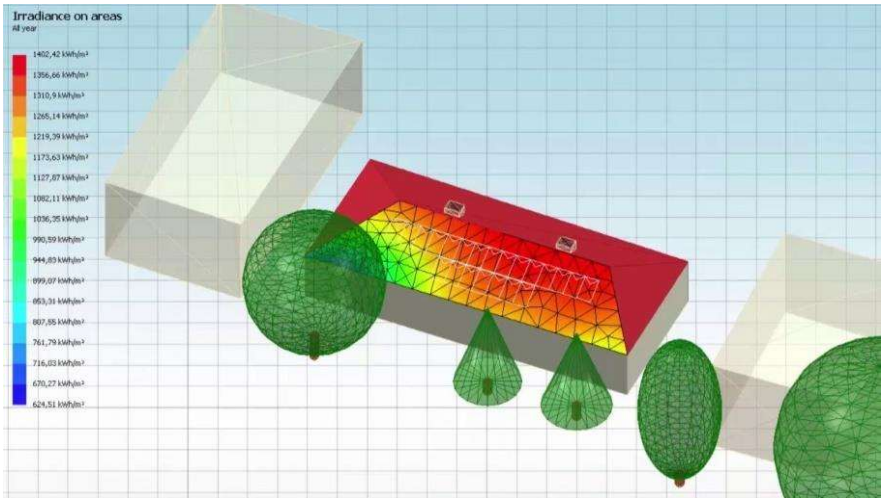


Рис. 1.11. Робоче вікно BlueSol.

Pylon є доступним онлайн-інструментом, користування яким не вимагає обов'язкового щомісячного платежу, лише. плата у розмірі 4 долари США вимагається за кожен проект, який створюється на Pylon. Важливою особливістю програмного засобу є наявність аерофотознімків високої роздільної здатності. Pylon спроможний виконувати 3D-аналіз сонячного затінення, аналіз інтервальних даних та профілів навантаження, і також здійснювати фінансові прогнози (рис. 1.12).

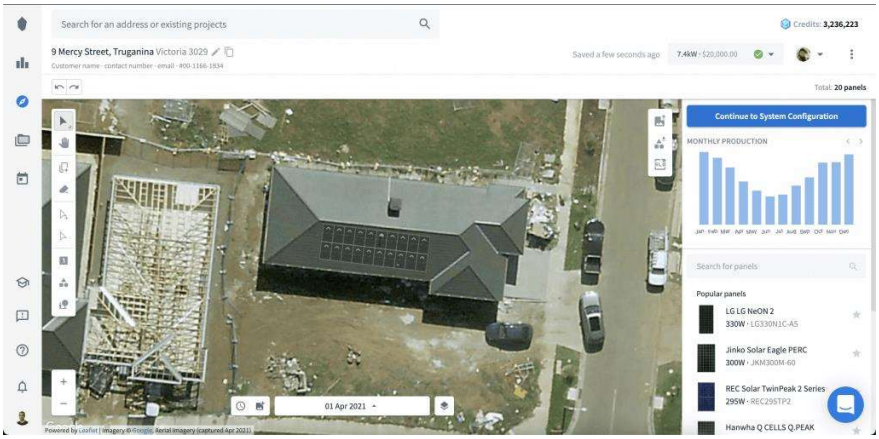


Рис. 1.12. Робоче вікно BlueSol.

1.2. Перелік найкращих програмних засобів для проектування фотоелектричних систем у 2024 році

1. Aurora Solar

Aurora Solar є одним з найпопулярніших інструментів у цій області.

Ключові особливості:

1) можливість проектування житлових та комерційних фотоелектричних систем;

2) створення високоточних конструкцій без виїзду на об'єкт;

3) візуалізація зазначених конструкцій за допомогою вбудованих програмних 3D-інструментів.

2. Helioscope

Helioscope є найкращим вибором для комерційних компаній. Платформа включає всі функції, необхідні для проектування складних фотоелектричних систем за короткий проміжок часу, покращуючи рентабельність інвестицій в бажані проекти.

Ключові особливості:

1) містить технологію LIDAR для забезпечення точного проектування;

2) створення CAD-проектів;

3) створення індивідуальних пропозицій щодо продажу та документів фінансового аналізу.

3. SolarEdge

Ключові особливості:

1) можливість покращення точності проектування за допомогою супутникових зображень HD та 3D-моделювання на основі штучного інтелекту;

2) швидка оптимізація планування даху завдяки карті освітленості та аналізу затінення;

3) автоматизовані розрахунки електричного проектування;

4) генерація пропозицій клієнтів, доповнених енергетичним моделюванням та прогнозами рентабельності інвестицій.

4. Pylon

Pylon є одним із найкращих програм для проектування фотоелектричних систем, оскільки він є потужним і простим у використанні та характеризується унікальною структурою ціноутворення.

Ключові особливості:

1) наявність аерофотознімків з високою роздільною здатністю;

2) набір інструментів для аналізу 3D-сонячного затінення;

3) наявність даних про інтервали, побудова моделі споживання клієнтами на основі аналізу профілів навантаження.

5. PVSOL

PVSOL надає користувачам доступ до розширених функцій, таких як 3D-аналіз затінення, планування використання потужностей акумуляторів, простий імпорт карт. Проте, недоліком є висока вартість придбання та обслуговування засобу.

Ключові особливості:

1) отримання реального зображення затінення від навколишніх об'єктів поблизу фотоелектричної системи, що проектується;

2) планування використання потужностей акумуляторів на основі обраних стратегій заряджання;

3) створення будівлі та об'єктів конструкцій фотоелектричної системи за допомогою планів поверхів та супутникових карт.

6. SolarPlus V4

SolarPlus V4 є універсальним засобом для компаній, що займаються сонячною енергетикою та характеризується найвищою ціною серед програмних продуктів, що використовуються в проектуванні фотоелектричних систем.

Ключові особливості:

1) варіанти проектування автономних гібридних та мережевих систем;

2) доступні можливості картографування завдяки інтеграції зображень MetroMap;

3) хмарні сховища для зберігання даних;

4) детальний фінансовий аналіз;

5) велика бібліотека компонентів, що присутні на ринку.

7. BlueSol

BlueSol дозволяє користувачам проектувати кожен аспект фотоелектричних систем. Не менш важливим є те, що цей програмний засіб може використовуватися компаніями в кожній країні світу.

Ключові особливості:

1) точна імітація поведінки фотоелектричних систем з усіма їх компонентами;

2) отримання точного уявлення про операції в системі завдяки зручній схематизації зображення компонентів системи;

3) інтеграція з системою CAD для створення електричних схем;

4) наявність функцій 3D-дизайну, які спрощують процес проектування.

1.3. Критерії оптимізації фотоелектричних систем

Існує два основних критерії оптимізації функціонування фотоелектричної системи, які ґрунтуються на технічній оцінці надійності енергозабезпечення в системі та економічній оцінці її функціонування. Ідеальною комбінацією для будь-якої фотоелектричної системи є оптимальний компроміс між двома розглянутими критеріями. Для цілі, що представлена у дипломі, оптимізація фотоелектричної системи розглядається лише з точки зору надійності енергозабезпечення в системі, тобто технічного критерію.

1.3.1. Аналіз надійності енергозабезпечення в фотоелектричній системі

При проектуванні фотоелектричних систем, особливо в автономних системах, одним з найважливіших аспектів забезпечення оптимального функціонування енергосистеми є аналіз надійності енергозабезпечення. Як відомо, генерація електричної енергії, що виробляється фотоелектричними панелями, розміщених у визначеній просторовій локації характеризується переривчастим, мінливим профілем генерації, і тому, зазвичай, вироблена енергія не відповідає потребам у навантаженні. Надійною фотоелектричною системою є енергосистема, в якій генерація електроенергії надає достатню потужність для забезпечення потреби в навантаженні протягом певного періоду. Існують два основних методи для визначення надійності енергосистеми, а саме оцінка імовірності відключення навантаження ІВН (*loss of load probability*, LOLP) та імовірності втрати потужності ІВП (*loss of power supply probability*, LPSP). В обох методах, якщо ймовірність дорівнює нулю, то навантаження є повністю забезпечене потрібною потужністю, в той час, коли ймовірність дорівнює одиниці, то навантаження не забезпечується ніякою потужністю.

ІВН – це ймовірність для випадку, коли потреба в навантаженні перевищує потужність, що виробляється фотоелектричною системою. В даному випадку надійна фотоелектрична система визначається як система, яка здатна генерувати достатню потужність E_{PV} для забезпечення необхідного навантаження E_L протягом відповідного проміжку часу.

$$\text{ІВН} = \frac{\sum_{i=1}^{8760} \text{Дефіцит потужності}_i}{\sum_{i=1}^{8760} \text{Потреба потужності}_i}$$

де

$$\text{Дефіцит потужності}_i = \sum_{i=1}^{8760} \left(E_L(i) - E_{PV}(i)\right)$$

З іншого боку, (LPSP) імовірність втрати потужності ІВП визначається як ймовірність виникнення випадку, коли система генерує недостатню потужність для задоволення потреби в навантаженні.

$$\text{ІВП} = \frac{\sum_{i=1}^{8760} \text{ВПЖ}(t)}{\sum_{i=1}^{8760} E(L)}$$

де

$$\text{ВПЖ}(t) = E_L(t) - [(E_{PV}(t)\eta_{bat}) + E_B(t) - E_{Bmin}] * [\eta_{inv} * \eta_{wire}]$$

Існує два підходи до застосування ІВП при проектуванні фотоелектричної системи, що ґрунтуються на хронологічному моделюванні та ймовірнісній оцінці. Хронологічне моделювання показує динамічні зміни в продуктивності системи, що, в свою чергу, вимагає даних часових рядів (*time series data*) за певний період, і тому більших обчислювальних зусиль, порівняно з методом, де використовується імовірнісна оцінка. З іншого боку, метод імовірнісної оцінки усуває потребу в даних часових рядів і може оцінювати довгострокові показники системи. Проте ця техніка більше не використовується останнім часом, оскільки характеризується відсутністю можливості для спостереження динамічних змін в продуктивності фотоелектричної системи.

1.4. Висновки до розділу

Були розглянуті різноманітні програмні засоби моделювання фотоелектричних систем, які дають змогу проектувати енергосистеми з бажаними параметрами, оцінювати їх функціонування та виконувати прогнозування роботи енергосистем з використанням метеорологічних даних. Найбільш актуальними на даний час є програмні засоби, які надають можливість для спостереження в режимі реального часу динамічних змін, що відбуваються в системі під час її роботи. Крім цього, найбільший інтерес викликають програмні засоби, які містять базу технічних характеристик фотоелектричних пристроїв, які присутні на теперішній час на ринку. Ці фактори обумовлюють задачу для створення такого програмного засобу, який забезпечив би потребу у базі фотоелектричних модулів та можливість моделювання функціонування системи в режимі реального часу. Слід зауважити, що всі описані засоби не спроможні виконувати оптимізацію функціонування фотоелектричної системи на основі технічного критерію, при якому забезпечується оптимальне функціонування енергосистеми. Для забезпечення такої вимоги необхідним є створення програмного засобу, який здатний виконувати прогнозування електрогенерації та енергоспоживання в фотоелектричній системі, використовуючи методи кластеризації даних та створення правил асоціацій на основі даних. Для цього програмний продукт повинен мати здатність збирати, зберігати та обробляти відповідним чином необхідні вхідні дані.

РОЗДІЛ 2
ВИБІР ЗАСОБІВ РОЗРОБКИ

2.1. Загальні положення

Фотоелектричний перетворювач (ФЕП) – це пристрій, що перетворює енергію електромагнітного випромінювання в електричну за рахунок виникнення явища фотоелектричного ефекту. ФЕП складається з двох шарів легованого кремнію *p*- та *n*-типу, в просторі між якими утворюються *p*–*n* перехід. (рис. 2.1).

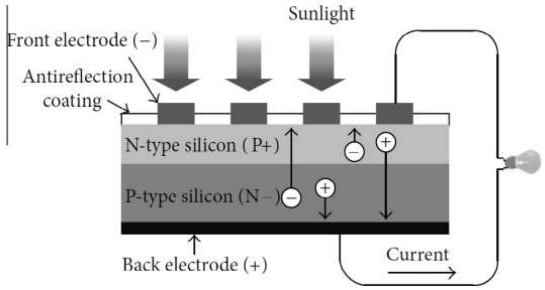


Рис. 2.1. Структура ФЕП

Кількість сонячного електромагнітного випромінювання,, що падає на поверхню ФЕП, буде залежати від його географічного положення, часу доби, пори року, локальних особливостей місцевості та погоди. Промені, що падають на поверхню містять дві складові: 1) безпосередні промені, інтенсивність яких залежить від відстані до джерелп випромінювання, 2) дифузні промені, виникнення яких спричинюється розсіюванням світла (рис. 2.2).

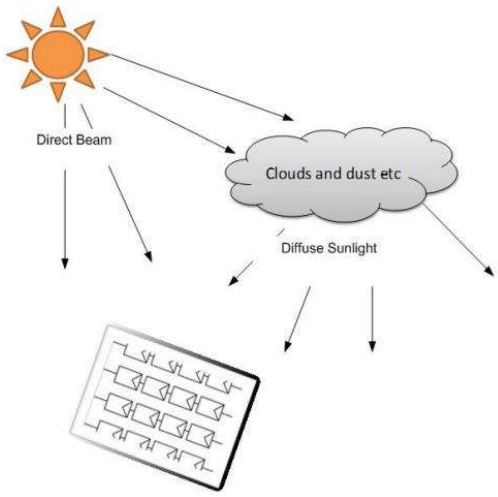


Рис. 1.2. Види променів, що падають на поверхню ФЕП

Стандартні умови (*Standard Test Conditions*, STC*)* призначаються для того, щоб можна було порівнянювати продуктивність різних ФЕП. Вимірювання, отримані в результаті такого тестування, вказуються в технічних характеристиках ФЕП, що надаються виробником. STC включають в себе: 1) стандартну потужність електромагнітного випромінювання, яке падає на поверхню (1000 $Вт/м^2$); 2) стандартну температуру поверхні ФЕП (25 $\circ C$) та 3) стандартний спектральний розподіл випромінювання з показником повітряної маси AM = 1,5, який виражає відстань, що проходять сонячні промені, коли тухаються через атмосферу.

Для того, щоб відповідним чином охарактеризувати ФЕП використовують такі технічні характеристики пристрою як струм короткого замикання (*Isc*), напруга холостого ходу (*Voc*) та параметри максимальної точки потужності (*Imp*, *Vmp*), в якій ФЕП досягає максимального значення продуктивності. Ці параметри представляються в технічній документації виробника на ФЕП і є тією базовою інформацією, яка потрібна для побудови математичної моделі ФЕП (рис. 2.3).

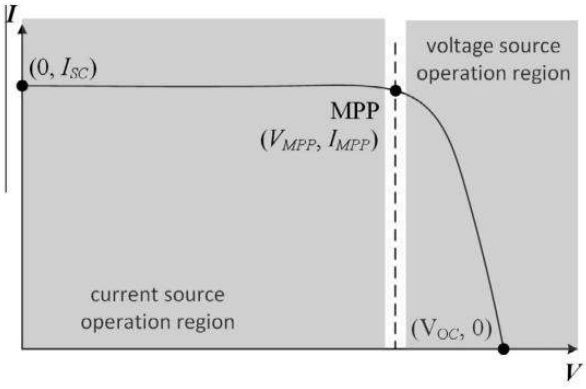


Рис. 2.3. Основні технічні характеристики ФЕП на кривій ВАХ

2.2. Математичне моделювання ФЕП

Для побудови математичної моделі, що опичує функціонування ФЕП спочатку визначають параметри рівняння моделі для заданих величин освітленості та температури з наступним розв'язком отриманого рівняння, використовуючи еквівалентні електричні схеми (рис. 2.4).

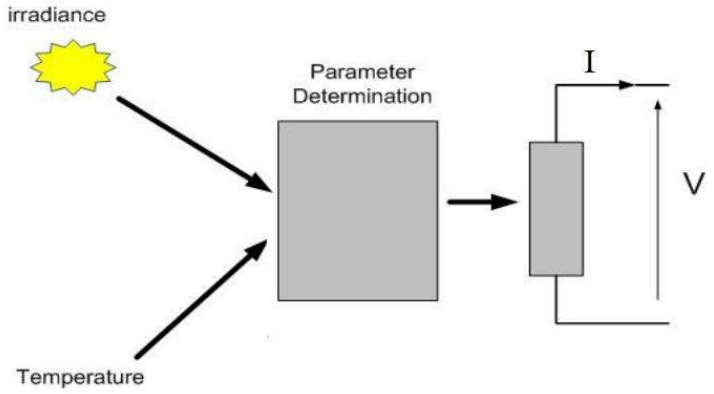
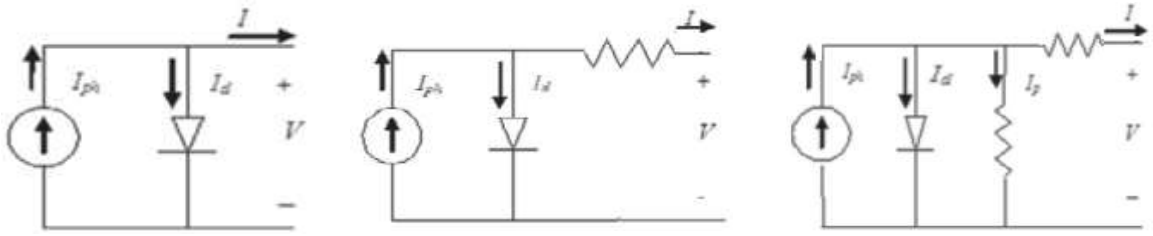


Рис. 2.4. Побудова математичної модної ФЕП.

Схему заміщення ФЕП, яка складається з таких елементів як джерело струму, діод, послідовний та(або) шунтувальний опори можна представити у вигляді ідеальної моделі з одним діодом (рис.2.5а), моделі з послідовним опором $\boldsymbol{R_s}$ (рис.2.5б) та моделі, що включає в себе як послідовний так і шунтувальний опори $\boldsymbol{R_s}$ та $\boldsymbol{R_p}$ (рис.2.5в).



а) Ідеальна модель з одним діодом

б) Модель з послідовним опором $\boldsymbol{R_s}$

в) Модель з послідовним опором $\boldsymbol{R_s}$ та шунтувальним опором $\boldsymbol{R_p}$

Рис. 2.5. Електричні схеми заміщення ФЕП

Ідеальна модель не враховує внутрішні втрати струму, діод підключається антипаралельно з джерелом струму, що генерується світлом, що падає на поверхню ФЕП. Вихідний струм $\boldsymbol{I}$, згідно закону Кірхгофа, розраховується за формулою:

$$I = I_{ph} - I_d \qquad (1)$$

де $\boldsymbol{I_{ph}}$ – фотострум та I_d – струм, що протікає через діод, який пропорційний струму насичення і визначається рівнянням:

$$I_d = I_0 \left[exp\left(\frac{V}{AN_S V_T}\right) - 1 \right] \qquad (2)$$

V – падіння напруги на діоді

$$V_T = k * {}^{T_c}/_{q} \quad (3)$$

I_0 – зворотний струм насичення або витоку діода (А), V_{Tc} = 26 мВ (при 300 К для кремнієвої панелі),

T_c – фактична температура комірки (К),

k – постійна Больцмана 1,381×10^{-23} Дж/К,

q – заряд електрона (1,602×10^{-19} С).

Величина V_T також називається тепловою напругою через її виняткову залежність від температури.

N_s – кількість ФВ панелей, з'єднаних послідовно,

А – коефіцієнт ідеальності, величина, що залежить від технології виготовлення ФЕП (Табл. 2.1).

Табл. 2.1. Коефіцієнт ідеальності (А)

Технологія виготовлення	**Коефіцієнт ідеальності**
Si-mono	*1,2*
Si-poly	*1,3*
a-Si-H	*1,8*
a-Si-H tandem	*3,3*
a-Si-H triple	*5*
CdTe	*1,5*
CTs	*1,5*
AsGa	*1,3*

Можна записати, що

$$a = \frac{N_s * a * k * T_c}{q} = N_s * A * V_T \quad (4)$$

де a є «модифікованим коефіцієнтом ідеальності».

З практичної точки зору неможливо нехтувати послідовним опором R_s та паралельним опором R_p через їх вплив на ефективність ФВ панелі. Якщо прийнятті до уваги R_s, тоді рівняння (2) набуде вигляду:

$$I_d = I_0 \left[exp\left(\frac{V + I*R_S}{a}\right) - 1 \right] \quad (5)$$

Знову ж, згідно закону Кірхгофа, величина струму, яку генерує ФВ панель, розраховується за рівнянням:

$$I = I_{ph} - I_d - I_p \quad (6)$$

де $\boldsymbol{I_p}$ – це витік струму з паралельного резистора.

Тоді,

$$I = I_{ph} - I_0 \left[exp\left(\frac{V + I*R_S}{a}\right) - \frac{V + I*R_S}{R_p} \right] \quad (7)$$

Необхідно зауважити, що визначення параметрів на основі цього трансцендентного рівняння є непростою справою. Проте, саме ця модель пропонує найкращий збіг розрахованих значень з експериментальними.

Визначення $\boldsymbol{I_{ph}}$

Вихідний струм при стандартних умовах випробування (СУВ, Standard Test Conditions, STC), згідно з рис. 1а, дорівнює:

$$I = I_{ph,ref} - I_{0,ref} \left[exp\left(\frac{V}{a_{ref}}\right) - 1 \right] \quad (8)$$

Це рівняння дозволяє кількісно визначити $\boldsymbol{I_{ph,ref}}$, у випадку, коли відбувається коротке замикання ФЕП:

$$I_{sc,ref} = I_{ph,ref} - I_{0,ref} \left[exp\left(\frac{0}{a_{ref}}\right) - 1 \right] = I_{ph,ref} \quad (9)$$

Але це рівняння є дійсним тільки в ідеальному випадку, тому рівняння (10) потрібно записати у вигляді:

$$I_{ph,ref} \approx I_{sc,ref} \quad (10)$$

Фотострум залежить як від освітленості, так і від температури:

$$I_{ph} = \frac{G}{G_{ref}} \left(I_{ph,ref} + \mu_{sc} * \Delta T \right) \quad (11)$$

де

$\boldsymbol{G}$ – Освітленість (Вт/м^2),

$\boldsymbol{G_{ref}}$ – Освітленість при STC = 1000 Вт/м^2,

$\boldsymbol{\Delta T = T_c - T_{c,ref}}$ (К),

$\boldsymbol{T_{c,ref}}$ – Температура ФВ панелі при STC = 25 + 273 = 298 К,

$\boldsymbol{\mu_{sc}}$ – Температурний коефіцієнт струму короткого замикання (А/К), згідно

паспортних даних,

$\boldsymbol{I_{ph,ref}}$ – Фотострум при STC (A).

Визначення $\boldsymbol{I_0}$

Опір шунтованого опору $\boldsymbol{R_p}$ прийнято вважати великим, тому останній член відношення (8) слід виключити з рівняння для того, щоб виконати відповідне наближення (апроксимацію). Застосовуючи рівняння (8) в трьох найбільш характерних точках при STC:

1) напруги при розімкнутому ланцюзі ($\boldsymbol{I = 0, V = V_{oc,ref}}$),

2) струму при короткому замиканні ($\boldsymbol{V = 0, I = I_{sc,ref}}$),

3) напруги ($\boldsymbol{V_{mp,ref}}$) та струму ($\boldsymbol{I_{mp,ref}}$) при максимальній потужності, можна записати наступні рівняння:

$$I_{sc,ref} = I_{ph,ref} - I_{0,ref}\left[exp\left(\frac{I_{sc,ref}*R_S}{a_{ref}}\right) - 1\right] \quad (12)$$

$$0 = I_{ph,ref} - I_{0,ref}\left[exp\left(\frac{V_{oc}}{a_{ref}}\right) - 1\right] \quad (13)$$

$$I_{pm,ref} = I_{ph,ref} - I_{0,ref}\left[exp\left(\frac{V_{pm,ref}+I_{pm,ref}*R_S}{a_{ref}}\right) - 1\right] \quad (14)$$

Величиною (-1) можна знехтувати, оскільки вона є значно меншою за експоненціальний вираз. Згідно рівнянням (11), та підставляючи ($\boldsymbol{I_{ph,ref}}$) в рівняння (14), отримується:

$$0 \approx I_{sc,ref} - I_{0,ref}exp\left(\frac{V_{oc,ref}}{a_{ref}}\right) \quad (15)$$

звідки

$$I_{0,ref} = I_{sc,ref}exp\left(\frac{-V_{oc,ref}}{a_{ref}}\right) \quad (16)$$

Зворотний струм насичення визначається як:

$$I_0 = DT_C^3 exp\left(\frac{-q\varepsilon_G}{A*k}\right) \quad (17)$$

ε_G – Енергія забороненої зони матеріалу (еВ), (1,12 еВ для Si),

D = коефіцієнт дифузії діода.

Для того щоб виключити коефіцієнт дифузії діоду, рівняння (18) обчислюється двічі: 1) при T_c та 2) при $T_{c,ref}$, після чого відношення двох рівнянь записується наступним чином:

$$I_0 = I_{0,ref}\left(\frac{T_c}{T_{c,ref}}\right)^3 exp\left[\left(\frac{q\varepsilon_G}{A*K}\right)\left(\frac{1}{T_{c,ref}} - \frac{1}{T_c}\right)\right] \quad (18)$$

$$I_0 = I_{SC,ref} exp\left(\frac{-V_{oc,ref}}{a}\right)\left(\frac{T_c}{T_{c,ref}}\right)^3 exp\left[\left(\frac{q\varepsilon_G}{A*K}\right)\left(\frac{1}{T_{c,ref}} - \frac{1}{T_c}\right)\right] \quad (19)$$

В рівнянні (19) параметри $V_{oc,ref}$, $T_{c,ref}$ та A, ε_G надаються виробниками ФВ панелей, в той час як параметри a та T_c є залежними від фактичної температури, саме з цієї причини I_0 визначається в режимі реального часу.

Визначення R_p та R_s

Для того, щоб зробити запропоновану модель більш достовірною, R_p та R_s вибирають так, щоб обчислена максимальна потужність P_{mp} була рівною експериментальнsq $P_{mp,ex}$ в умовах STC, таким чином записуючи наступні рівняння:

$$I_{mp,ref} = {P_{mp,ref}}\Big/{V_{mp,ref}} = {P_{mp,ex}}\Big/{V_{mp,ex}} = I_{ph,ref} -$$

$$-I_{0,ref}\left[exp\left(\frac{V_{mp,ref}+I_{mp,ref}*R_S}{a}\right) - 1\right] - \frac{V_{mp,ref}+R_S I_{mp,ref}}{R_p} \quad (20)$$

$$R_p = \frac{V_{mp,ref} + I_{mp,ref}R_S}{I_{sc,ref} - I_{sc,ref}\left\{exp\left[\frac{V_{mp,ref} + R_S I_{mp,ref} - V_{oc,ref}}{a}\right]\right\} +}$$

$$+I_{sc,ref}\left\{exp\left(-{V_{oc,ref}}/{a}\right)\right\} - \left({P_{max,ex}}\Big/{V_{mp,ref}}\right) \quad (21)$$

Ітерація починається з величини послідовного опору $R_s=0$, який повинен збільшуватися, при переміщенні змодельованої точки максимальної потужності до того значення, де вона збігається із значенням експериментальної точки максимальної потужності, після чого обчислюється відповідне значення шунтованого опору R_p. Слід зауважити, що існує тільки одна пара значень R_p та R_s, яка задовольняє цій умові.

Таким чином, відповідне експериментальне значення величини максимальної потужності ФЕП (з паспортних даних. що надаються виробником панелей) вводиться в рівняння (22), для того, щоб за допомогою ітераційного методу обчислити значення послідовного R_S та шунтованого R_P опорів (Рис. 2.6).

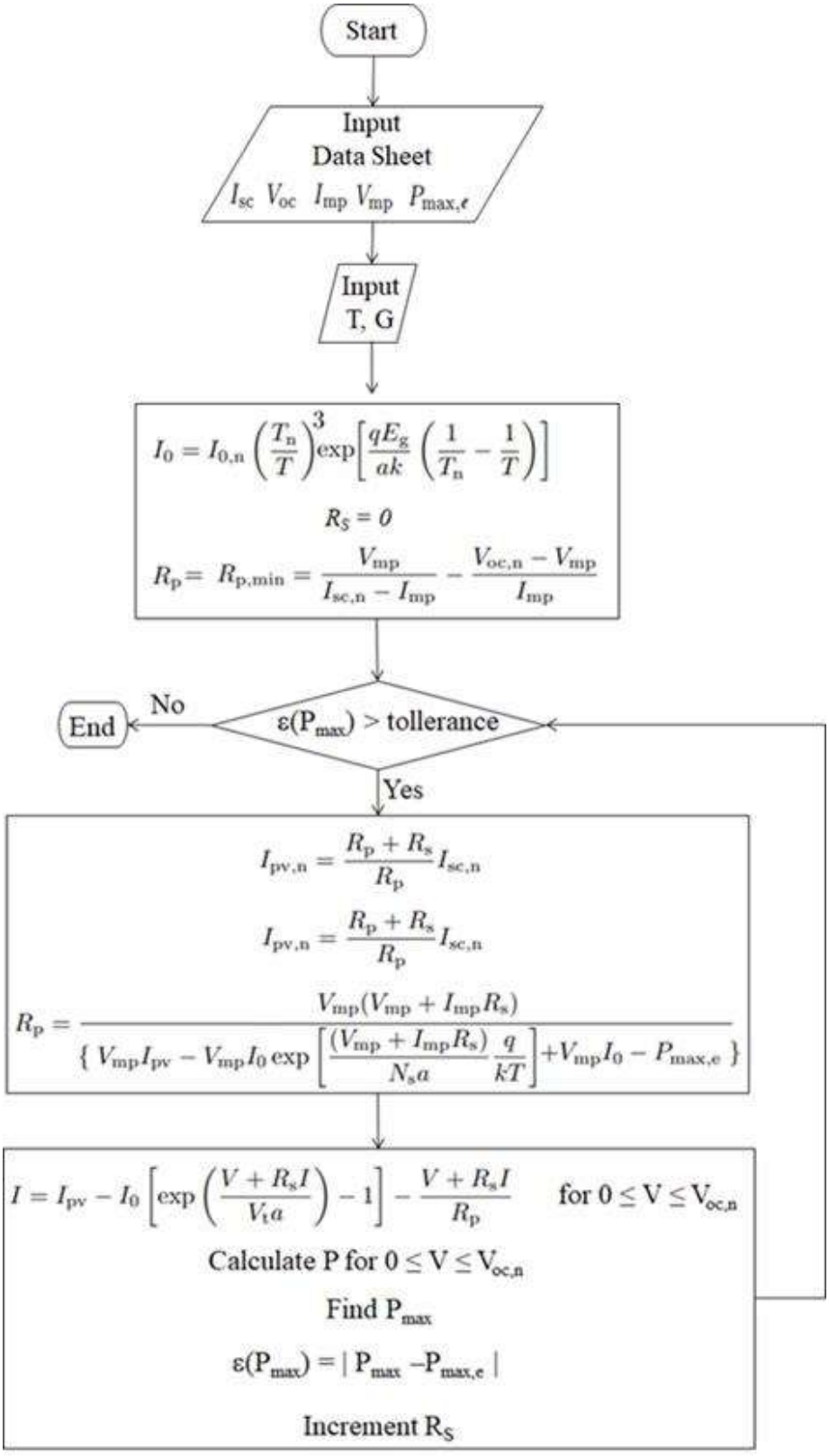


Рис. 2.6. Блок-схема імітаційного моделювання ФЕП

2.3. Вибір технологій та інструментів для реалізації

2.3.1. Мови програмування

Python

Мова програмування *Python* характеризується багатим набором можливостей, гнучкістю та доступністю інструментів при реалізації програмних проектів. Головною перевагою *Python* є високий рівень абстракції, що дозволяє концентруватися на алгоритмах та логіці та витрачати мінімум часу, який потрібен для розробки технічних елементів. *Python* надає зручну інтеграцію з бібліотеками широкого спектру функціональності при обчисленях, обробки даних та математичного моделювання. Зокрема, бібліотеки *NumPy* та *SciPy* дозволяють оптимально використовувати чисельні методи, включно з розв'язанням диференціальних рівнянь, для виконання задач, що пов'язані з інтерполяцією, оптимізацією та регресійним аналізом. Бібліотека *pandas* дозволяє ефективно працювати з великими масивами даних, включаючи виконання відповідних операцій над ними, зокрема проводити статистичний аналіз та кластеризацію з використанням інструментів бібліотеки *scikit-learn*. Також *Python* характеризується значними можливостями візуалізації як вхідних даних, так і отриманих результатів, що надають бібліотеки *Matplotlib* і *Seaborn*, які забезпечують створення високоякісних графіків та діаграм. Інтеграція з інструментами баз даних, наприкла *PostgreSQL*, з використанням бібліотеки *psycopg*2, дозволяє ефективно маніпулювати великими масивами даних.

JavaScript* та платформа *Node.js

JavaScript (JS) є динамічною, високорівневою мовою програмування для потреб *WEB*-розробки. Хоча, слід зауважити, що від самого початку JS була розроблена для сценаріїв на стороні клієнта для реалізації клієнтських *WEB*-додатків і покращення інтерактивності користувацьких інтерфейсів у браузері. Проте, згодом, завдяки платформі *Node.js*, вона розширила свої можливості для свого використання на стороні сервера. Це дозволило зробити JS важливим інструментом при створенні серверних застосунків, забезпечуючи високу гнучкість та продуктивність у їх проектуванні як на клієнтському, так і на серверному боці. Основною особливістю JS є інтерпретація коду безпосередньо у браузері в режимі реального часу без потреби попередньої компіляції, що забезпечує ефективну роботу з *HTML*-елементами та *DOM*-моделлю та дозволяє динамічно змінювати вміст і структуру *WEB*-сторінок під час їх відображення у

браузері. Таким чином, описані особливості JS надають йому здатність бути універсальним інструментом для ефективного програмування в клієнт-серверній архітектурі, що забезпечує оптимальне обслуговування та інтеграцію всіх компонентів системи. Враховуючи перспективу розширення проєкту, що розробляється та включення нових джерел даних, обраний стек технологій має бути масштабованим. *Node.js* забезпечує розроблену сукупність бібліотек, що надає можливість інтегрувати необхідні компоненти для роботи з БД, виконання статистичних операцій та аналізу даних.

Таким чином, *JS* у поєднанні з *Node.js* надає потрібні можливості для розробки та проектування ІЕС завдяки своїй асинхронній, подієво-орієнтованій архітектурі, яка дозволяє ефективно обробляти запити в режимі реального часу. Як зазначалося вище, *Node.js* є потужною платформою, яка дозволяє використовувати *JS* з відкритим кодом, яка дозволяє розробникам створювати масштабовані та високопродуктивні мережеві застосунки. Його модель *I/O*, керована подіями, що не блокується, дозволяє ефективно обробляти декілька з'єднань, що робить його особливо придатним для застосунків, що працюють з даними у режимі реальному часі.

2.3.2. Програмні засоби

HTML та CSS

HTML (*Hypertext Markup Language*) та *CSS* (*Cascading Style Sheets*) є фундаментальними технологіями для створення структури та візуального оформлення *WEB*-застосунків. *HTML* є мовою розмітки, яка визначає структуру *WEB*-сторінки створенням її базових елементів, до яких відносять текстові фрагменти, рисунки, таблиці та шаблони. *CSS*, в свою чергу, забезпечує стилістику таких елементів, задаючи їхні відповідні текстові характеристики, які включають кольори, шрифти, розміри та розміщення на сторінці. *HTML* і *CSS*, як слідує з вищенаведеного, надають широкі можливості для створеняя структурованих та естетично привабливих інтерфейсів користувачів.

Використання *HTML* та *CSS* в границях клієнт-серверної архітектури забезпечує потрібну основу для створення інтерактивних *WEB*-сторінок, які мають можливість взаємодіяти з сервером на *JS*. На цій базі, клієнтська частина

ІЕС надаватиме зручний та візуально естетичний інтерфейс для користувача, який надсилає відповідний запит, відображаючи необхідні дані з серверної частини, надаючи при цьому змогу користувачеві здійснювати пошук, перегляд та обробку інформації про процеси, що протікають в ІЕС.

SASS (*Syntactically Awesome Style Sheets*) – це метамова для *CSS*, що розширює можливості традиційної каскадної стилізації за допомогою додаткових функцій, які спрощують створення та редакцію коду. На відміну від *CSS*, *SASS* забезпечує використання змінних, вкладених правил, циклів, множин та умов та підтримку вбудованих математичних виразів. Використовуючи змінні у *SASS* є можливість централізовано керувати характеристиками стилю (наприклад, кольоровими схемами та розмірами шрифтів), що спрощує процес оновлення та підтримки коду. Крім того, використання вкладених правил надає можливість створювати ієрархічну структуру стилів, що покращує загальне сприйняття та організайію коду.

Комплексне використання *HTML*, *CSS* та *SASS у поєднання* з *JS* надає ефективний набір інструментів для розробки необхідного *WEB*-додатку. Знову ж таки, можна повторити, що *HTML* визначає структуру сторінки, *CSS* та *SASS* забезпечують її стилізацію, а *JS* інтерактивність та логіку. Таким чином, резумуючи, можна стверджувати, що об’єднання *HTML*, *CSS*, *SASS* та *JS* забезпечує зручний, структурований та гнучкий базовий інструментарій для розробки клієнтської частини ІЕС. Використання такого інструментарію дозволяє створити інтерфейс, який відповідає всім необхідним вимогам, що стосуються комфорту, ефективності та інтерактивності, тоді як використання *SASS*, як засобу для управління стилями, забезпечує включення додаткових переваг у масштабуванні та підтримці коду.

Фреймворк Express

Express є часто використовуваним програмним фреймворком для розробки серверних додатків в *Node.js*, який характеризується достатньою простотою та гнучкістю. Він забезпечує набором функцій, які полегшують створення *WEB*-додатків та *API*, що спричинює його високу популярність серед розробників. Він надає можливість швидко та ефективно створювати *WEB*-сервери, обробляти

HTTP-запити та взаємодіяти з клієнтами через *RESTful API*. *Express* відрізняється своїм мінімалістичним дизайном та завершеною бібліотекою функцій, які належним чином спрощують створення серверних за стосунків, що дозволяє швидко налаштувати основну структуру сервера, що є надзвичайно важливим для скорочення часу розробки. *Express* дозволяє створювати модульні та масштабовані *WEB*-додатки, оскільки підтримує широкий спектр *middleware*, який надає можливість розширювати функціональність без необхідності переписування основного коду, що. особливо є корисним для проєктів, які мають потенціал для подальшого розвитку.

Для розроблюваної системи, яка передбачає високу інтерактивність між сервером та клієнтами, *Express* ідеально підходить для побудови та обробки *RESTful API*, що дозволяє ефективно організувати відповідний обмін даними між сервером та клієнтською частиною. Такий підхід спрощує роботу з великими масивами даних та забезпечує відповідний до них доступ. *Express* належним чином інтегрується з іншими інструментами програмування, що використовується для розробки серверів, наприклад, БД, інструменти для обробки запитів та маршрутизації, інші фреймворки та бібліотеки для зручності своєїрозробки. Саме завдяки таким характеристикам *Express* є ідеальним інструмент ним засобом для розробки серверної частини ІЕС, де важливими критеріями вибору вважається масштабованість системи та можливість ефективного використання *API*.

Шаблоні затори EJS

EJS (*Embedded JS*) є достатньо простим та ефективним шаблонізатором для створення *HTML*-кодів у *Node.js*-застосунках. Він дозволяє вставляти *JS*-логіку безпосередньо в *HTML*-шаблони, що дозволяє динамічно формувати відповідний вміст на сервері, перед тим як надіслати його клієнту. *EJS* використовує синтаксис, подібний до стандартного *HTML*, проте дає можливість вбудовувати *JS* вирази та логіку за допомогою спеціальних значків, для виведення значень змінних або для виконання *JS*-коду без показу результату. *EJS* надає можливість створювати динамічні *HTML*-сторінки, де контент формується на серверній частині, залежно від вхідних даних, що дає змогу ефективно відображати інформацію, яка змінюється, без зайвої потреби у перезавантаженні сторінки, або

виконанні комплексних запитів, що формуються в клієнтській частині. *EJS* дозволяє безпосередньо передавати змінні величини з серверної частини у *HTML*-шаблони, що забезпечує створення такої *WEB*-сторінки, в якій дані з бази даних або з *API*, можуть відображатися в режимі реального часу.

EJS підтримує основні конструкції *JS*, такі як умови та цикли, що дозволяє гнучко керувати виведенням даних у шаблони. Це дуже корисно при створенні інтерфейсів, де необхідно відображати змінний контент, залежно від отриманих результатів аналізу даних або стану системи. Оскільки *EJS* виконується на сервері перед тим, як буде відправлено *HTML*-код клієнту, зменшується навантаження на клієнтську сторону, що є важливим для підтримання високої швидкості завантаження та ефективності функціонування інтерфейсу. *EJS* легко інтегрується з різними фреймворками, наприклад, фреймворком *Express*, що дозволяє швидко налаштувати сервер для відповідної обробки шаблонів. При цьому забезпечується необхідна зручність при розробці серверної частини для динамічного створення *HTML*-сторінок, які використовують дані, що отриуються з сервера, Можна стверджувати, що завдяки простоті, гнучкості та високій інтеграції з іншими інструментами на платформі *Node.js*, *EJS* є належним вибором для розробки динамічних *WEB*-сторінок при проектуванні, забезпечуючи необхідний та ефективний спосіб створення відповідного *HTML*-контенту.

npm менеджер пакетів

npm (*Node Package Manager*) – це менеджер пакетів для функціонування в *JS*, який автоматично встановлюється разом із середовищем *Node.js* та надає можливість керувати бібліотеками та модулями, які використовуються різного типу проектах; *npm* надає змогу без зайвих труднощів встановлювати, оновлювати та видаляти пакети, а також керувати їх взаємодією. *npm* є необхідним засобом для управління бібліотеками, що, в свою чергу, дозволяє суттєво спрощуваи розробку та обслуговування проєктів на *JS*. Однією з найбільших переваг *npm* є централізована базисна система пакетів, в якій розміщені тисячі бібліотек та модулів для вирішення самих різноманітних задач, починаючи від побудови серверних за стосунків, та закінчуючи роботою з БД та інтерфейсами користувача. Це, в свою чергу, значною мірою скорочує час

розробки. *npm* також підтримує систему семантичного шаблонування версій, що відповідає стандарту *Semantic Versioning* 2.0.0 (*SemVer*). Ця система регулює призначення версій пакетів, де номер версії задається форматом *MAJOR.MINOR.PATCH*, в якому вказується тип змін у новому шаблоні. *SemVer* забезпечує сумісність пакетів, забезпечуючи впевненість у тому, що відповідні оновлення будуть сумісними з поточною версією проєкту та не порушать стабільність його функціонування. Завдяки описаним особливостям, *npm* надає можливість належним чином відстежувати зміни та впроваджувати оновлення, забезпечуючи підтримку актуальності проєкту.

Фреймворк *Qt*

Фреймворк *Qt* характеризується багатофункціональністю, кросплатформеністю та широкими можливостями для розробки графічних інтерфейсів та складних прикладних програм. *Qt* забезпечує інтегроване середовище для створення застосунків, які поєднують в собі високу продуктивність, інтуїтивний користувацький інтерфейс та достатню модульну архітектуру. Однією з основних переваг фреймворку *Qt* є його підтримка кросплатформенності, що дозволяє виконувати програмні рішення, які можуть фнкціонувати на різних операційних системах, таких як *Windows*, *Linux*, *macOS* чи навіть мобільних платформах, без необхідності значних змін у коді. Така особливість є важливою у контексті проєкту, орієнтованого на універсальність та масштабованість IEC.

Бібліотека *D3.js*

D3.js (*Data-Driven Documents*) є бібліотекою *JS* для створення динамічних, інтерактивних візуалізацій даних у *WEB*-браузері. Ця бібліотека дозволяє зв'язувати дані з елементами *WEB*-сторінки, керуючи ними за допомогою *HTML* та *CSS* для створення графіків, діаграм, карт та інших візуальних презентацій, які динамічно оновлюються у відповідності до змін у даних. *D3.js* підтримує достатній набір засобів для роботи з даними, зокрема, це фільтрація, агрегація, сортування та трансформація даних перед тим, як виконати їх візуалізацією. *D3.js* знаходить широке застосування в різних галузях, де важливою є візуалізація великих масивів даних. Перевагами використання *D3.js* в проектуваннях є

інтерактивність, можливість роботи з великими масивами даних, управління візуалізацією, підтримка різноманітних типів графіків, інтеграція з іншими *WEB*-інструментами та динамічне оновлення даних. Однак, слід зауважити, що робота з графіками в *JS* є технічно складною і потребує значного проміжку часу на налаштування, редагування та форматування візуальних елементів. Проте, потрібно підкреслити, що саме використання бібліотеки *D3.js* забезпечує високий рівень управління візуалізацією, що надає можливість належним чином виконувати адаптацію відповідних графіків до специфіки проєкту відносно вимог користувача. Завдяки гнучкій структурі даних, *D3.js* надає можливість не тільки відображати графіки в браузері, але й також створювати функціональну здатність для збереження візуалізацій у форматі зображень.

MATLAB

MATLAB (Matrix Laboratory) є високорівневим програмним середовищем, яке надає достатні можливості для виконання технічних обчислень та програмування, з відповідними функціональн можливостями, які охоплюють досить широкий простір задач, що включають в себе роботу з даними, моделювання та розробки. Це програмне середовище характеризується наступними функціональними можливостями, а саме: 1) здатністю здійснювати різноманітні математичні обчислення за допомогою базових або складних арифметичних операцій, матриць та операцій над ними, розвязувати диференційні рівняння та системи рівнянь, використовувати статистичні та чисельні методи для виконання інтегрування або оптимізації; 2) проводити аналіз та обробку даних, використовуючи засоби для роботи з великими масивами даних, включаючи аналіз часових рядів, статистичний аналіз та методи машинного навчання; 3) засобами ефективної візуалізації для створення 2D (3D) графіків, їх налаштування та анімації, побудови графічних інтерфейсів користувача (GUI); 4) можливістю розробки алгоритмів та програмування на основі мови MATLAB, підтримки об'єктно-орієнтованого програмування, автоматизації робочих процесів через сценарії та функції, інтеграції з іншими мовами програмування (C, C++, Python, Java). Крім того, MATLAB у формі сукупного інструментарію Simulink надає майже необмежені можливості для створення моделей динамічних систем на базі

додаткових пакетів (toolboxes), що використовуються для моделювання складних систем, зокрема механічних, електричних або хімічних, виконуючи їх симуляцію в режимі реального часу.

MATLAB забезпечує ефективну інтеграцію з пристроями, що належать до апаратного забезпечення через Data Acquisition Toolbox, зокрема він надає підтримку для роботи з Arduino, Raspberry Pi, та іншими мікроконтролерами. Також MATLAB має здатність забезпечувати автоматизацію інженерних процесів через Model-Based Design Toolbox, генеруючи відповідний код. І нарешті MATLAB спроможний виконувати хмарні та паралельні обчислення, використовуючи обчислювальні ресурси та робота з графічними процесорами (GPU) у хмарному середовищі.

2.3.3 Протоколи HTTP

HTTP (*HyperText Transfer Protocol*) – це протокол, який визначає стандарти передачі даних між *WEB*-клієнтами та серверами в мережі Інтернет. *HTTP* здійснює обмін запитами та відповідями між браузером) та сервером, надаючи клієнту можливість надсилати запити щодо отримання або передачі необхідної інформації та отримувати відповіді з відповідним вмістом. Завжди, коли користувач заходить на *WEB*-ресурс, його браузер надсилає *HTTP*-запит, в той час як сервер обробляє цей запит і надсилає відповідь у вигляді *HTML*, зображень, стилів чи інших ресурсів, які відображаються на екрані браузера. Для створення динамічного вмісту в клієнт-серверній архітектурі використовується шаблонізатор *EJS*, який дозволяє формувати *HTML* безпосередньо на сервері, використовуючи шаблони з вбудованим *JS*-кодом. Таким чином, використання *EJS на*дає можливість серверу створювати динамічний *HTML*, який відповідає на запити користувачів, відображає відповідні дані та підвищує ступінь комунікативності, забезпечуюючи основні функції для передачі даних та генерації контенту на сервері. Тому *HTTP* є основою функціонування *WEB*-програм, забезпечуючи передачу тексту, даних, зображень, файлів. *HTTP* використовує запити, серед яких найпоширенішими є *GET*, *POST*, *PUT* та *DELETE*. *GET* формує запит ресурсу з сервера, *POST* надсилає дані для обробки, *PUT* оновлює та замінює дані, а *DELETE* видаляє контент. *HTTP* характеризується здатністю легко

налаштовуватися для роботи в мережі, що надає можливість інтегрувати сервер із клієнтськими додатками, незалежно від платформи та операційної системи (ОС). Крім того, *HTTP* спрощує обробку запитів на отримання даних та дозволяє клієнтам формувати запит на інформацію з сервера в режимі реального часу.

Таким чином, *HTTP* виступає універсальним та надійним рішенням для побудови WEB-додатку з клієнт-серверною архітектурою, забезпечуючи ефективний обмін даними, сприяючи ефективній інтерактивній взаємодію між користувачем та серверною частиною застосунку.

2.3.4. Формат обміну даними *JSON*

Використання *HTTP* є ключовим для реалізації взаємодії з *API*, що забезпечує обмін даними між серверною частиною системи та зовнішніми джерелами інформації, такими як погодні сервіси або бази даних енергетичних показників. *HTTP* виступає основним транспортним протоколом для передачі запитів та отримання відповідей, що дозволяє організувати надійну і ефективну комунікацію в рамках системи, що проектується. Для цього використовується *JSON* (*JavaScript Object Notation*) формат обміну текстовими даними який часто використовується як ефективний засіб для представлення даних у різних програмних проєктах. Відсутність складних шаблонів та типів даних значно спрощує роботу з *JSON*, що є важливим у процесі розробки, коли потрібно швидко та ефективно обробляти дані з різних джерел. Чітка та ієрархічна структура цього формату полегшує візуальний аналіз та всестороннє охоплення інформації, що надходить з різних серверів. Також важливим є. *JSON* характеризується незалежністю від використовуваної мови програмування, підтримується більшістю наявних мов програмування, що забезпечує універсальність цього формату при його використанні у проєктах, в яких залучаються різноманітні технології. Слід зауважити, що більшість мов програмування мають вбудовані або зовнішні бібліотеки для роботи з *JSON*, що значно спрощує процес обробки даних, дозволяючи швидко отримувати доступ до необхідної інформації, виконувати потрібні операції та забезпечувати ефективну обробку великих масивів даних за мінімальний проміжок часу та без зайвих

витрат програмних ресурсів. Завдяки наявності компактного формату, *JSON* дозволяє зберігати великі масиви даних в невеликих за об'ємом файлах, надаючи можливість значно знижувати витрати на зберігання та передавання інформації, що є важливим для проектування системи, яку планують використовувати для роботи з великими масивами даних різних типів та змісту.

Формат *JSON* є основним, який використовується для роботи з *API* при передачі запитів клієнтів та відповідей з серверної частини. Кожен запит до зовнішнього *API* формулюється як *HTTP*-запит з параметрами, що відповідають потребам клієнта, а відповідь від *API*, в якому містяться відповідні дані, надаються у форматі *JSON*, який потім обробляється серверною частиною відповідним способом, в залежності від потреб клієнта.

2.4. Висновки до розділу

У другому розділі кваліфікаційної роботи були розглянуті необхідні інструменти (програмні засоби, бібліотеки та протоколи) для проектування клієнт-серверної архітектури ІЕС, виконано аналіз та обґрунтування їх вибору. Зокрема, описано спосіб реалізації імітаційного моделювання ФЕП, озглянуто структуру вхідних даних для реалізації такого моделювання, способи їх отримання та використання. Крім того, були описані методи та інструменти для створення відповідної архітектури програмного засобу, який включає в себе концепцію інтерфейсу користувача в клієнтській частині та відповідну структуру серверної частини за стосунку, включаючи її взаємодію з БД. Запропонована архітектура включає необхідні етапи обробки запитів, та механізми для ефективної обробки транзакцій, ґрунтуючись на відповідних вимогах високої продуктивності, масштабованості, безпеки та надійності функціонування програмного засобу. Був вибраний та визначений основним для реалізації взаємодії між компонентами системи протокол *HTTP*, який також підтримує передачу даних через *API*,. Формат *JSON* вибрано як стандартизований спосіб представлення даних для забезпечення сумісності та зручності обміну інформацією. Крім того, були вибрані середовище розробки *VS Code*, БД *PostgreSQL*, фреймворки *Express* та *Qt*, бібліотека *D*3.*js*, стандарт *ECMAScript*,

шаблонізатор *EJS*, менеджер пакетів *npm*, метамова для *CSS* (*SASS*), *HTML*, середовище розробки *Visual Studio Code* (*VS Code*).

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

3.1. Загальні положення. Інтелектуальна електрична система ІЕС

ІЕС – це взаємозв'язана енергетична система з відповідними технологіями, які використовуються для апаратної автоматизації та програмного інтелектуального керування з метою підвищення ефективності, надійності, стійкості та гнучкості функціонування електричної системи при виробництві, передачі, розподілі та споживанні електроенергії (рис. 3.1).

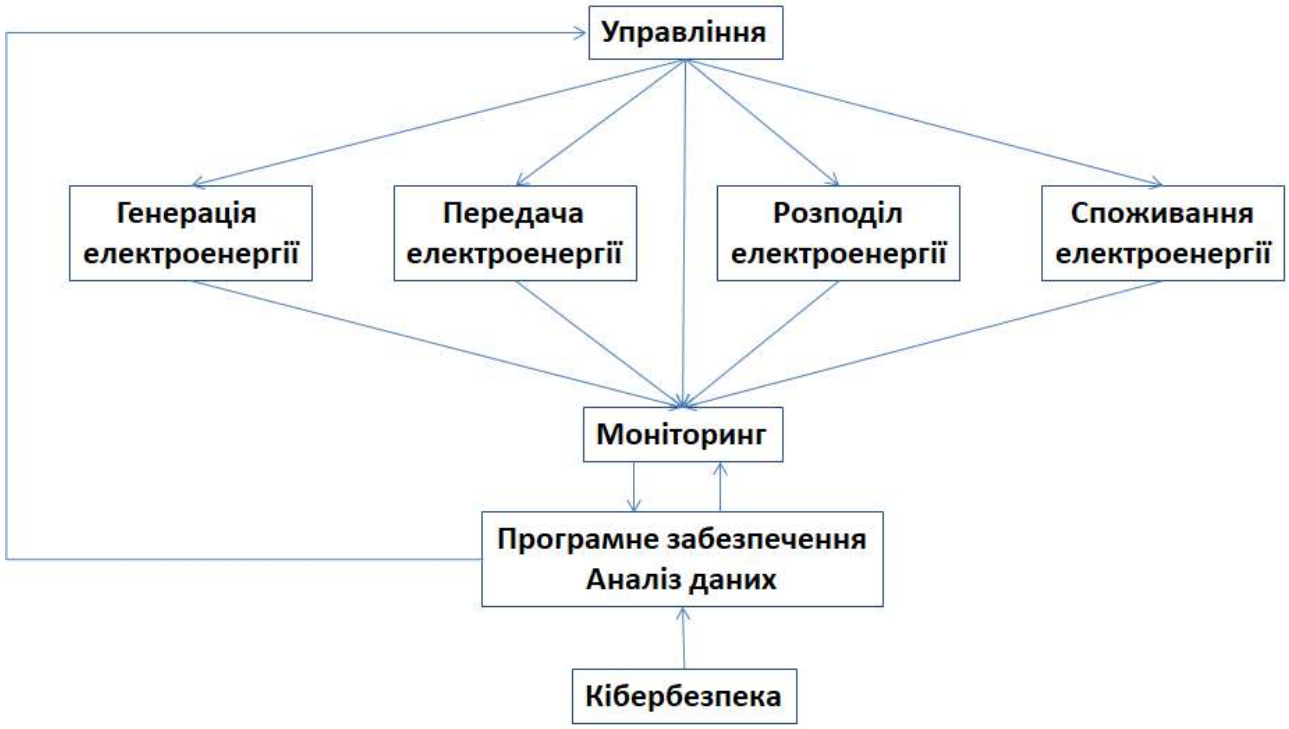


Рис. 3.1. Структура ІЕС

Така концепція часто асоціюється з концепцією розумних мереж (*smart grids)*, які також концептуально охоплюплюють мікромережі (*microgrids)*, системи відновлюваної енергії (*renewable energy systems)* та інтелектуальне управління енергією будівель (*smart building energy management)*.

Компоненти ІЕС можна класифікувати на основі їх функцій, які вони виконують в системі: 1) моніторинг, 2) управління, 3) комунікація та 4) енергоменеджмент. До компонентів ІЕС належать:

Засоби для генерації та накопичення електроенергії (сонячні панелі, вітрогенератори, паливні генератори, акумулятори та емнісні накопичувачі).

Засоби для передачі електроенергії (високовольтні лінії електропередачі, пристрої стабілізації електричної мережі та релейний захист, пристрої компенсації реактивної потужності, трансформатори).

Засоби для розподілу електроенергії (розподільчі підстанції та лінії, автоматичні вимикачі та стабілізатори напруги).

Пристрої, що споживають електроенергію.

Моніторинг та управління електроенергією забезпечується диспетчерським управлінням шляхом збору відповідних даних, на основі яких виконується контроль над ефективним функціонуванням електричної системи.

Програмний додаток, що розробляється для ефективного функціонування системи електропостачання з ФЕП складається з двох основних частин: 1) програмного засобу для імітаційного моделювання ФЕП та 2) WEB-додатку (рис. 3.2).

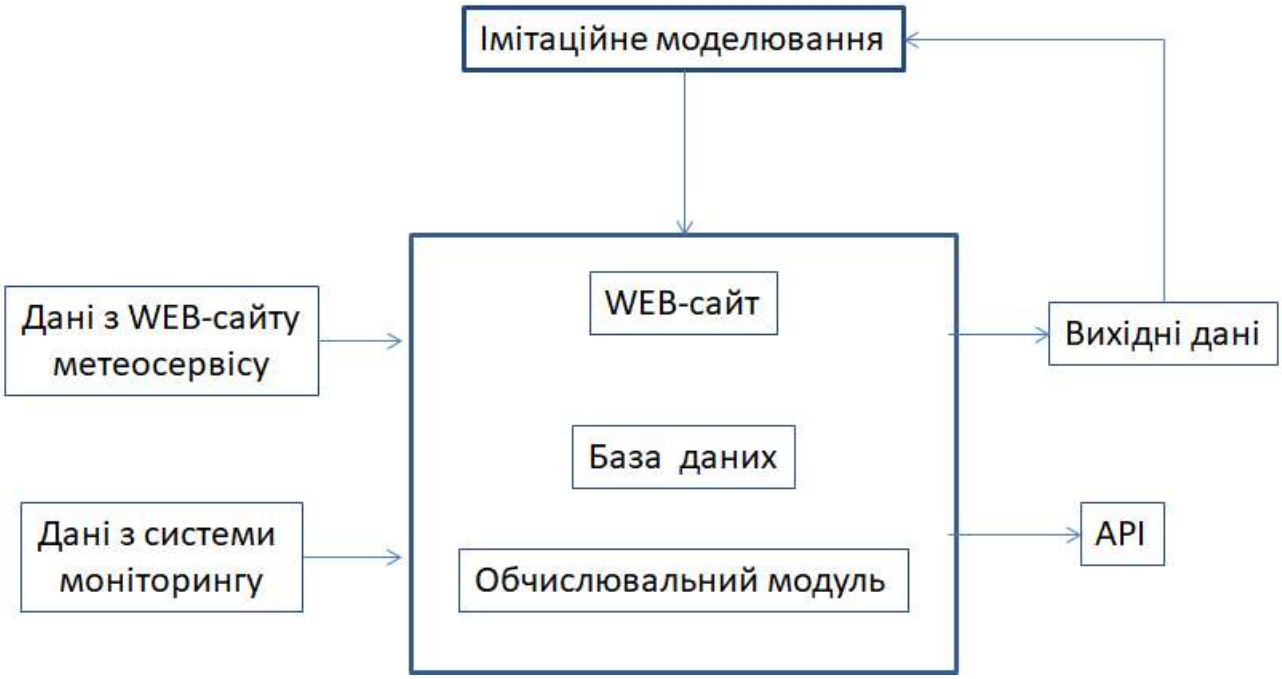
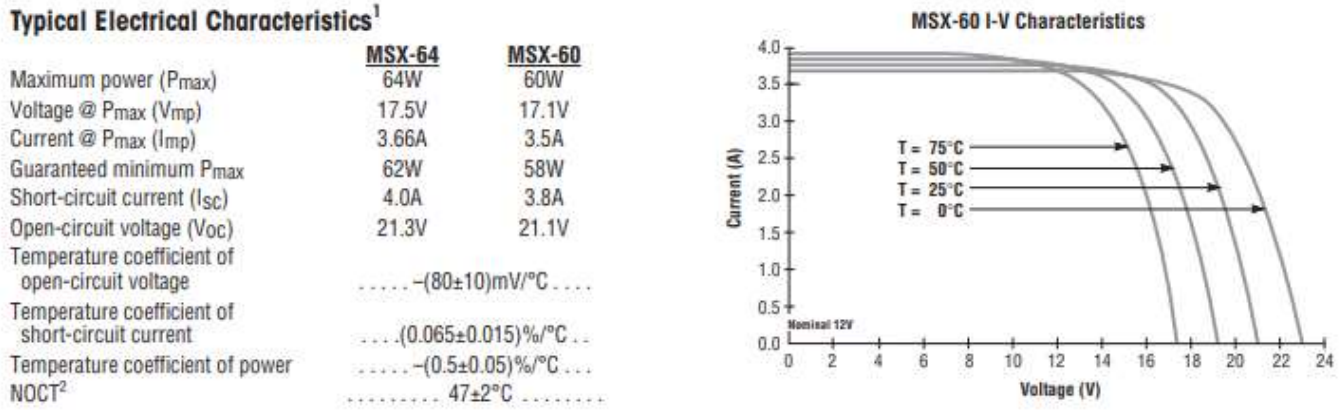


Рис. 3.2. Архітектура програмного додатку,для ефективного функціонування системи електропостачання з ФЕП.

3.2. Програмний засіб для імітаційного моделювання ФЕП

Основним елементом інтерфейсу програмного засобу є головне вікно, що забезпечує вхід для користувача, який виконує симуляцію. Це вікно надає доступ до всіх основних функцій імітаційного моделювання, включаючи вибір типу компонентів та налаштування необхідних параметрів для подальших розрахунків. Користувач вносить необхідні дані для моделювання конкретної ФЕП або використовує функцією збереження конфігурацій.

Параметри, що характеризують конкретну панель, вносяться користувачем. Ці параметри включають в себе введення назви панелі, її марки та шифр, технічні характеристики на основі даних, що надаються виробником (рис. 3.3).



Typical Electrical Characteristics[1]

	MSX-64	MSX-60
Maximum power (P_{max})	64W	60W
Voltage @ P_{max} (V_{mp})	17.5V	17.1V
Current @ P_{max} (I_{mp})	3.66A	3.5A
Guaranteed minimum P_{max}	62W	58W
Short-circuit current (I_{sc})	4.0A	3.8A
Open-circuit voltage (V_{oc})	21.3V	21.1V
Temperature coefficient of open-circuit voltage	.....–(80±10)mV/°C....	
Temperature coefficient of short-circuit current	....(0.065±0.015)%/°C..	
Temperature coefficient of power	.....–(0.5±0.05)%/°C...	
NOCT[2]	..... 47±2°C	

Рис. 3.3. Приклад технічних характеристик сонячної панелі MSX-60.

Після того, як користувач зберігає інформацію про панель, вона надалі зберігається в *JSON* файлі наступного формату:

```
{
   "Model Name": "MSX60",
   "Vocn": "21.1",
   "Vmp": "17.1",
   "Imp": "3.5",
   "Iscn": "3.8",
   "Pmax_e": "59.85",
   "Ш": "0.003",
   "Kv": "-80e-3",
   "Ns": "36"
```

}

Однією з важливих особливостей програмного додатку є його інтеграція з серверною частиною, з якої автоматично отримуються дані про освітленість та температуру, які є вхідними даними для виконання імітаційного моделювання. Після введення всіх необхідних параметрів користувач може приступити до виконання моделювання. В результаті симуляції будуються вольт-амперна характеристика (ВАХ), що показує залежність між напругою та струмом, а також вольт-ватна характеристика потужності (ВВХ), яка демонструє залежність між напругою та потужністю, що виробляється панеллю. Метою обчислення є визначення точки максимуму потужності, величина якої використовується при оптимізації функціонування ФЕП. Для збереження результатів тестування та забезпечення можливості подальшого аналізу застосунок використовує файлову систему *JSON*. Вона складається з комірок пам'яті, де зберігаються ключові дані, такі як назва, шифр, паспортні дані ФЕП, а також вольт-амперні та вольт-ватні характеристики панелі, отримані за *STC* (1000 *Вт/м²*, 25°*C*).

3.2.1. Архітектура програмного засобу

Архітектура розроблюваного застосунку є багаторівневою та модульною, що забезпечує його масштабованість, зручність у розробці та супроводі, а також адаптивність до змін вимог або інтеграції нових компонентів. Основні рівні архітектури охоплюють графічний інтерфейс користувача, бізнес-логіку, обробку даних та зовнішні інтеграції. У цьому розділі детально розглянемо ключові аспекти архітектури та їх взаємодію.

На верхньому рівні архітектури розташований графічний інтерфейс користувача (*Graphical User Interface*, *GUI*), який реалізовано з використанням бібліотеки *PySide*6 для *Python*. Інтерфейс забезпечує введення параметрів для моделювання, таких як дати, вибір таблиць даних із серверу, а також відображення результатів моделювання. Основним принципом побудови *GUI* є мінімалістичність і орієнтація на користувача, що дозволяє інтерактивно керувати параметрами системи без необхідності глибокого розуміння внутрішньої структури застосунку.

Другий рівень архітектури складається з модуля бізнес-логіки, який відповідає за обробку введених даних і управління основними процесами. Наприклад, у цьому модулі реалізовано функціональність формування *HTTP*-запитів до серверу, обробку отриманих даних у форматі *JSON*, підготовку даних для *MATLAB* та ініціацію виконання математичних обчислень. Бізнес-логіка організована так, щоб максимально розділити функціональні частини, що спрощує їх тестування та модифікацію.

Третій рівень архітектури – це обробка даних, яка виконується у два етапи. Перший етап включає отримання даних із серверу через *RESTful API*. Дані отримуються у форматі *JSON* і перетворюються на структури, придатні для подальшої обробки. Другий етап полягає в підготовці цих даних для *MATLAB*. Зокрема, числові масиви освітленості та температури конвертуються у формати, що підтримуються *MATLAB*, *matlab.double*. Цей рівень архітектури інтегрує функціонал *MATLAB* через *Python API*, що дозволяє виконувати складні математичні розрахунки, не виходячи за межі застосунку.

Четвертий рівень – це зовнішні інтеграції, які включають взаємодію з сервером для отримання даних та з *MATLAB* для обчислень. Сервер надає часо-просторову інформацію про освітленість і температуру, необхідну для моделювання роботи фотоелектричних систем. *MATLAB* виконує функцію обчислювального ядра, реалізуючи алгоритм *MPPT* і моделюючи роботу фотоелектричних панелей. Інтеграція забезпечується через *Python API MATLAB*, що дозволяє безпосередньо викликати *MATLAB*-функції з *Python*-застосунку.

Центральною концепцією архітектури є її модульність. Кожен функціональний блок, будь то *GUI*, серверна взаємодія чи обчислення, реалізується окремо, що спрощує тестування, налагодження та розширення. Така архітектура забезпечує не тільки високу продуктивність, але і точність функціонування програмного засобу, що створює фундамент для подальшого вдосконалення додатку, масштабування та інтеграції з новими модулями чи компонентами.

3.2.2. Забезпечення взаємодії з серверним *API*

Процес взаємодії розпочинається з того, що користувач через інтерфейс застосунку ініціює запит до серверу. Відповідно, в застосунку передбачена можливість відправлення *HTTP*-запитів до серверу для отримання потрібної інформації.

Коли користувач ініціює цей процес, інтерфейс застосунку відправляє запит до серверного модуля через стандартні протоколи передачі даних. Для цього використовуються бібліотеки *Python*, такі як *requests*.

Команда запиту формується таким чином:

– Користувач вибирає таблицю через графічний інтерфейс, що відображається в комбінаційному полі *table_combobox*. Це значення потім передається до *URL* як параметр *table*.

– Користувач вказує період для отримання даних, вибираючи дати у полях *start_date_edit* та *end_date_edit*. Ці значення конвертуються у формат "*yyyy-MM-dd*" і передаються як параметри *startDate* та *endDate*.

Формується *URL*, на який надсилається запит, виглядаючи таким чином:

http://{*serverip*}:3000/*view*?*columns=IrradianceTemperature*&*table=*{*table*}&*startDate=*{*start_date*}&*endDate=*{*end_date*}.

Загальний механізм взаємодії передбачає відправку запиту на сервер, де він обробляється відповідним серверним модулем. Сервер, у свою чергу, звертається до бази даних для отримання потрібних даних. Після цього сервер формує відповідь у форматі *JSON* та надсилає її назад до застосунку.

Отримані дані через сервер інтегруються в систему, що дає змогу коригувати параметри моделювання в реальному часі. Зокрема, якщо температура чи сонячна радіація змінюються, ці значення передаються до модуля симуляції, що дозволяє знову виконати розрахунки з урахуванням нових умов. Ці зміни можуть значно вплинути на вихідні результати моделювання, а також на оптимізацію роботи *MPPT*-контролера для забезпечення найбільш ефективної роботи системи.

3.2.3. Інтеграція з *MATLAB*

Одним з ключових моментів інтеграції з *MATLAB* є забезпечення коректної взаємодії між *Python* та *MATLAB* шляхом налаштування середовища для запуску *MATLAB* із *Python* за допомогою бібліотеки *matlab.engine*. Різні версії *MATLAB* та *Python* можуть викликати конфлікти, що в значній мірі ускладнює процес інтеграції. Для вирішення цього питання необхідно використовувати сумісні версії програмного забезпечення, чітко налаштувати шляхи доступу до *MATLAB*-бібліотек і проводити тестування для перевірки коректності запуску *MATLAB*-функцій із *Python*.

Ще одним викликом є обробка великого обсягу даних, які надходять із серверу у форматі *JSON*. Висока кількість запитів або обробка складних даних може вплинути на продуктивність системи. Для оптимізації цього процесу застосовується ефективний алгоритм парсингу даних із використанням бібліотек *Python*, таких як *json*. Крім того, попередня обробка даних на стороні серверу може значно зменшити обсяг переданої інформації, що сприятиме швидкості роботи застосунку.

Важливою складністю є забезпечення стабільної роботи графічного інтерфейсу користувача. Використання *PySide*6 для створення *GUI* вимагає продуманого проектування, щоб уникнути зависань або втрати продуктивності під час виконання обчислень *MATLAB* або під час очікування відповіді від сервера. Для вирішення цієї проблеми доцільно використовувати асинхронні виклики та окремі потоки для виконання завдань, які займають тривалий час. Такий підхід дає змогу уникнути блокування основного потоку, що відповідає за взаємодію з користувачем.

Крім того, інтеграція серверного запиту з урахуванням різних параметрів, таких як датовані діапазони чи вибір таблиць даних, вимагає ретельного налагодження *API*-запитів. Потенційними проблемами є відсутність стабільного з’єднання, повільна відповідь серверу або некоректний формат даних у відповіді. Для уникнення таких ситуацій необхідно забезпечити перевірку валідності отриманих даних, використання механізмів повторних запитів у разі невдалого з’єднання, а також підготовку тестового серверу для відпрацювання усіх сценаріїв.

3.2.4. Тестування програмного додатку

Першим кроком є реалізація графічного інтерфейсу користувача (*GUI*. Весь інтерфейс створюється з кількох ключових елементів, таких як поля для введення дат, комбіновані поля для вибору джерела даних, кнопки для активації процесів і відображення результатів. Всі ці елементи створюються у вигляді віджетів і компонуються в організовану структуру за допомогою вертикальних і горизонтальних розміщень.

Після отримання вхідних даних будь-яким з описаних способів, вони передаються у форматі списків до *MATLAB*, де відбувається їх функціональна обробка у функції. Для цього використовуються функції *MATLAB*, які працюють з переданими списками даних і повертають обчислені результати.

Після виконання розрахунків результат виводиться на екран користувача. Якщо ж відбувається помилка на будь-якому етапі (відправка запиту, отримання даних або виконання обчислень), користувач отримує повідомлення про помилку через інтерфейс.

Для тестування застосунку було проведено запуск його функціональних можливостей. Перш за все потрібно отримати погодинні параметри моделювання з серверу. Між двома таблицями даних *station_data* (дані з київської локальної дослідної станції) та *pvgis_api* (дані з європейського супутника спостереження, орієнтуючись на координати Києва) було обрано таблицю *pvgis_api* та отримання даних у проміжку від 01.01.2020 до 05.01.2020, тобто інформація повинна відобразитись за 5 днів (рис. 3.4).

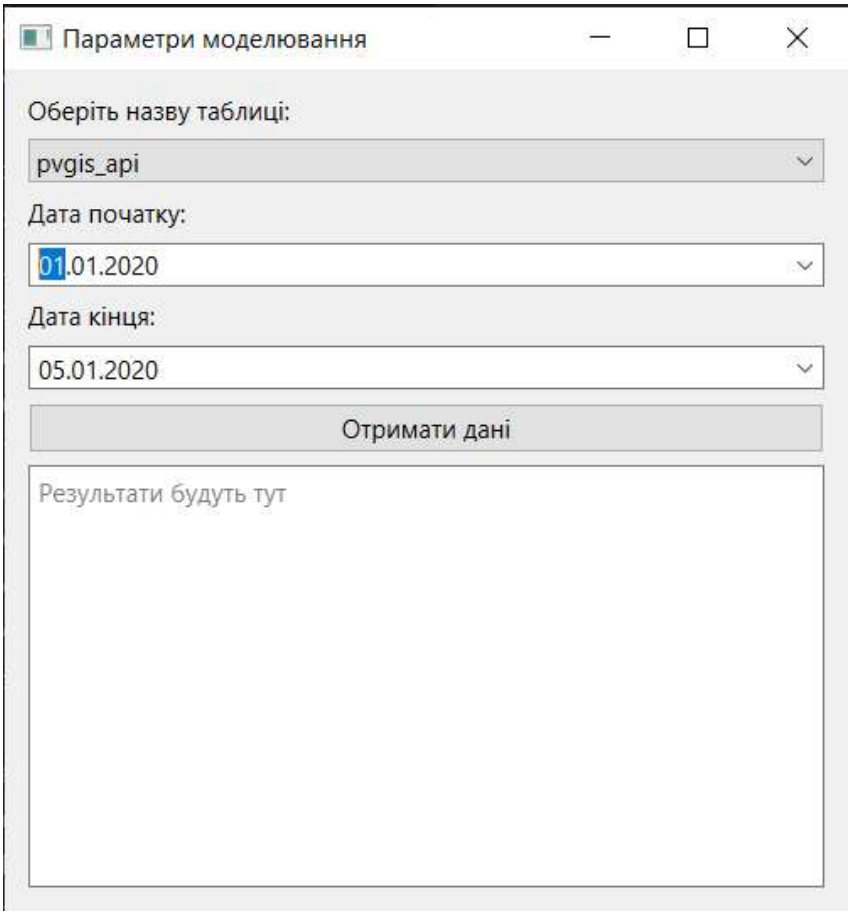


Рис. 3.4. Отримання вхідних параметрів освітленості та температури

Визначивши вхідні дані для застосунку, натискаємо кнопку “Отримати дані”. Отримані дані відображаються в об’єкті *textBox*, вміст якого відображає отриманий масив погодинних даних об’єктів, що містять інформацію про освітленість та температуру (рис. 3.5). Опісля отримання цих даних їх можна використовувати для подальшого моделювання.

Передачу інформації можна виконувати як вручну, так і за допомогою додаткового скрипта передачі інформації до наступного вікна.

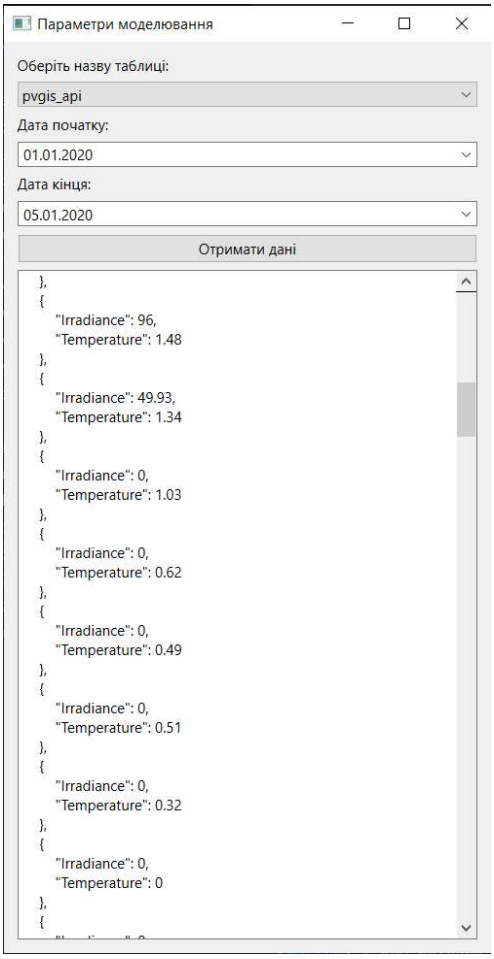


Рис. 3.5. Відображення отриманих параметрів

Маючи параметри освітленості та температури потрібно визначити технічну інформацію для сонячної панелі. Якщо користувач вже раніше користувався застосунком та зберігав раніше уведені дані, то він має можливість завантажити інформацію ФЕП в застосунок з файлу *JSON*. У тестовому випадку виконується запуск програми для характеристик панелі *MSX*60 (рис. 3.6).

Попри те, що інформація може завантажуватись, користувач може динамічно змінювати її в полях форми, що допомагає в швидкій заміні інформації на оновлену, з якої можна отримати нові дані. В особливості це стосується освітленості та температури.

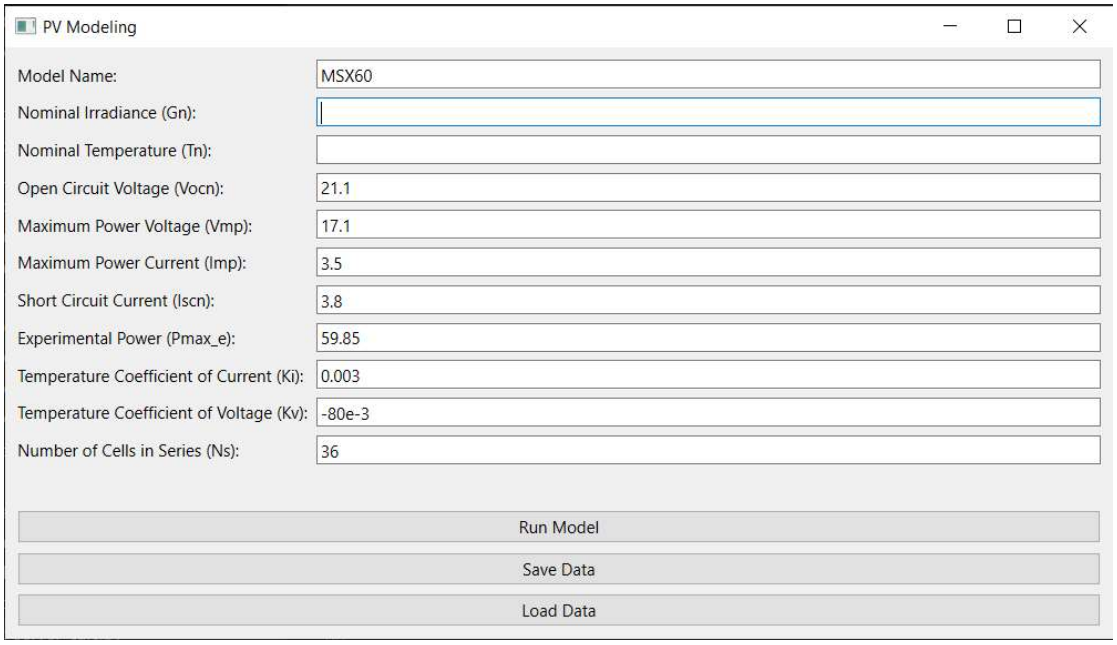


Рис. 3.6. Головне вікно програми

Після запуску програми ми отримуємо графіки проміжних значень, де застосунок намагається знайти точку максимальної потужності проходячи цикл ітерацій (рис. 3.7).

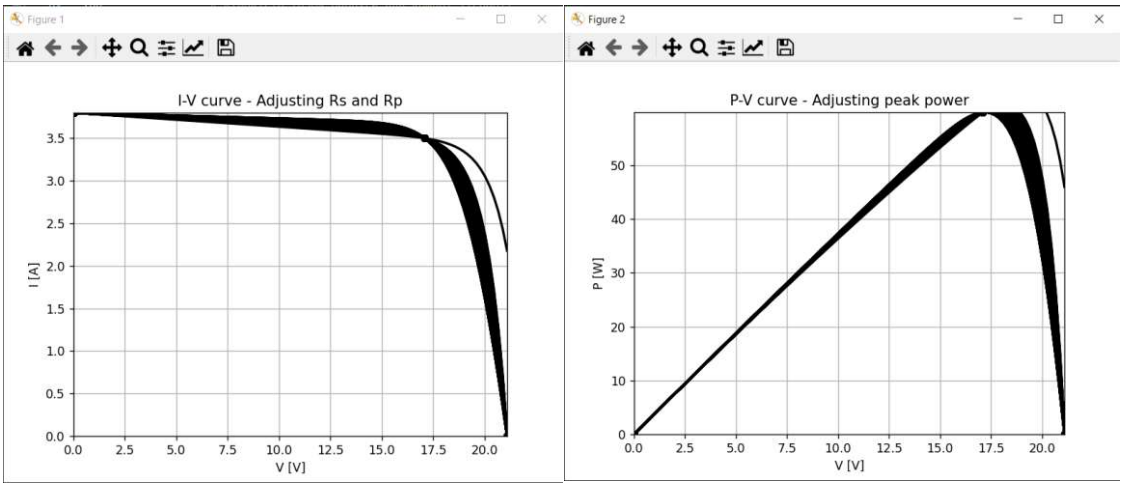


Рис. 3.7. Проміжні результати обрахунку точки потужності

У кінці виконання застосунку отримується результат з двох характеристик, вольт-амперної та вольт-ватної, що характеризують точку максимальної потужності панелі при введених даних (рис. 3.8).

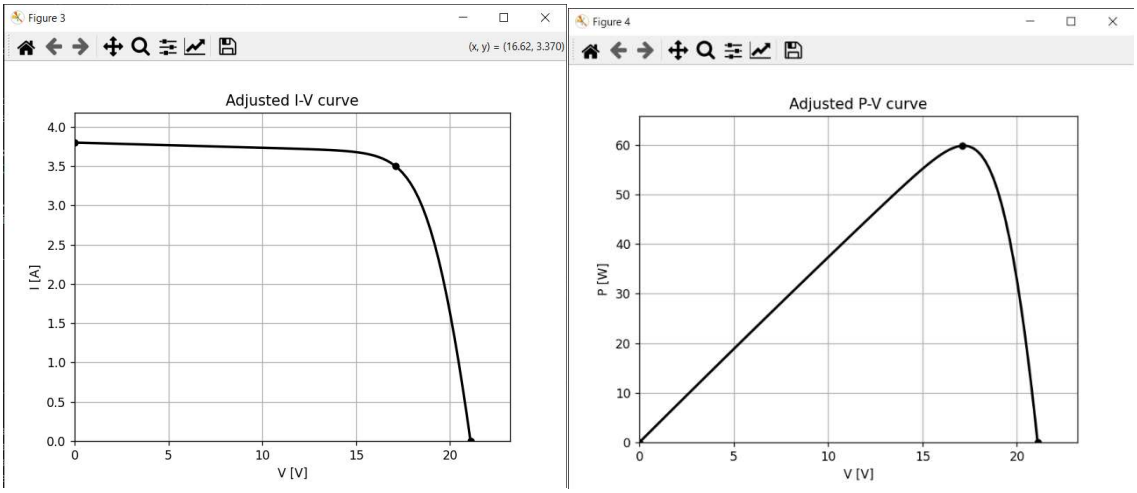


Рис. 3.8. Отримані оптимізовані точки максимальної потужності для сонячної панелі *MSX*60

3.2.5. Інтеграція застосунку разом з компонентами *MATLAB Simulink*

Інтеграція *Python* з *Simulink* через *S*-функції використовується для розширення можливостей імітаційного математичного моделювання, при застосуванні складних обчислення або спеціалізованих алгоритмів. *S*-функція в *Simulink* є спеціалізованим блоком, який дає можливість користувачам використовувати свої власні преференції для створення моделі. Це дозволяє використовувати як стандартні функції *Simulink*, так і включати власний користувацький код, написаний на інших мовах програмування, включаючи *Python*. Для інтеграції *Python* з *Simulink* за допомогою *S*-функції, створюють спеціалізований блок, який застосовує *Python*-скрипт або функцію через *MATLAB*.

Однією з ключових переваг такого підходу є його гнучкість, адже він забезпечує доступ до широкого спектра інструментів і технологій Python, які можна використовувати для вдосконалення та розширення можливостей моделі Simulink. Завдяки інтеграції Python можна поєднувати потужність MATLAB і Simulink для моделювання складних інженерних систем, одночасно використовуючи сучасні методи обчислень, аналізу та машинного навчання, які пропонує Python. Використання Python у Simulink через S-функції є особливо корисним у ситуаціях, коли потрібні зовнішні бібліотеки чи розширені функції, недоступні у стандартному середовищі MATLAB/Simulink. Наприклад, такі бібліотеки Python, як NumPy, SciPy чи Pandas, дозволяють виконувати

високопродуктивну обробку даних і складні математичні розрахунки, тоді як TensorFlow або PyTorch дають змогу інтегрувати методи глибокого навчання безпосередньо у моделі. Це особливо актуально для розробки інтелектуальних систем управління, які потребують прогнозування, оптимізації чи класифікації даних у реальному часі.

Крім того, інтеграція Python спрощує взаємодію з іншими зовнішніми системами, такими як бази даних, веб-сервіси чи апаратні пристрої. Наприклад, можна налаштувати блок Simulink для виконання Python-скриптів, які отримують дані з сенсорів або передають результати моделювання в хмарний сервіс для подальшого аналізу. Це розширює сферу застосування Simulink у міждисциплінарних проєктах, що вимагають інтеграції з різними платформами й технологіями. Таким чином, використання Python через S-функції сприяє підвищенню ефективності розробки та створенню інноваційних рішень у галузі інженерного моделювання та аналізу.

3.3. Розробка *WEB* додатку

3.3.1. Компоненти серверної частини

WEB-додаток складається з клієнтської та серверної частин, БД, а також засобів їхньої взаємодії. Основним елементом проєкту є файл *app.js*, який слугує вхідною точкою для серверної програми. Цей файл відіграє роль координатора, що ініціює запуск основних компонентів системи, Папка *bin* концентрується на виконавчих файлах, включаючи файл *www*, який працює як технічна вхідна точка, забезпечуючи запуск сервера та налаштування середовища, що розгортається. В папкці *config* містяться конфігураційні файли, зокрема *default.json*, що зберігає глобальні константи та налаштування. *Controllers* реалізує імплементарну логіку додатку, де кожен файл відповідає за конкретну функціональність, що дозволяє ізолювати основні алгоритми від клієнт-серверної взаємодії, що сприяє повторному використанню коду. Дані, що отримуються з локальної метеостанції, зберігаються у папці *data* у вигляді *log*-файлів. Відкриті ресурси зберігаються у папці *public*. Вона включає файли стилів, зображення, шрифти та скрипти для клієнтської частини. У папці *routes* визначаються маршрути для обробки запитів

до сервера, де кожен файл відповідає за окремий функціонал, наприклад, отримання результатів нормалізації чи кореляції через *API*, що впорядковує взаємодію між клієнтом і сервером. Файли стилів *SCSS* організовані у папці *scss*, з яких вони компілюються у вигляді готових *CSS*-стилі, що зберігаються у *public*/*css*. В папці *Utils* містить допоміжні скрипти, що надають функції для роботи з базою даних, обробки даних і візуалізації. Вони діють як сервісний шар, підтримуючи основний функціонал програми. Папка *views*, що відповідає за генерацію *HTML*-сторінок, поділена на компоненти (*partials*) та повні сторінки (*pages*), які забезпечують динамічне формування інтерфейсу користувача та повторне відтворення вже створених елементів.

Таблиці *pvgis_api* та *station_data* являються основними джерелами даних для аналізу видобутку енергії сонячними панелями.

Таблиця *pvgis_api* містить дані, отримані через *API PVGIS* для географічної точки з координатами 50.45° *N*, 30.525° *E*:

1. *Day* (*Date*) це дата вимірювання.

2. *Time* (*time without time zone*) це час вимірювання, вказаний без прив'язки до часової зони, що дозволяє аналізувати та ідентифікувати погодинні показники сонячної енергії.

3. *Power* (*real*) використовується для оцінки ефективності вироблення енергії, тобто це прогнозована потужність, яку може виробити сонячна панель у визначеній місцевості за певних умов.

4. *Irradiance* (*real*): є інтенсивністю сонячного випромінювання на площу (Вт/м²), яка відображає кількість сонячної енергії, доступної для перетворення в електроенергію.

5. *Sun_height* (*real*)це висота сонця над горизонтом (градуси).

6. *Temperature* (*real*) є температурою навколишнього середовища (°C).

7. *Wind* (*real*) виражає швидкість вітру (м/с).

Таблиця *station_data* містить фактичні дані, зібрані локальною метеостанцією, і забезпечує реальний контекст вироблення енергії.), де

1. *Day* (*Date*): Дата вимірювання, що дозволяє співвіднести дані з таблиці *pvgis_api*.

2. *Time* це час вимірювання значення якого відповідають значенням з *PVGIS*.

3. *Power* (*real*) - потужність, вироблена ФЕП (Вт).

4. *Voltage* (*real*) - напруга, що вироблена ФЕП (В).

5. *Current* (*real*) - струм, який генерується ФЕП (А).

6. *Irradiance* (*real*) - інтенсивність сонячного випромінювання (Вт/м²).

7. *Humidity* (*real*):- відносна вологість повітря (%).

8. *Temperature* (*real*) - температура навколишнього середовища (°*C*).

9. *Pressure* (*real*) - атмосферний тиск (гПа).

Порівняння фактичних та історичних значень, що містяться в таблицях, дає змогу оцінити точність аналізу, а за зміни даних і точність прогнозу, виявити відхилення, спричинені локальними умовами, та ідентифікувати фактори, що найбільше впливають на вироблення енергії.

Серверна частина застосунку реалізована на платформі *Node.js* з використанням фреймворку *Express.js*. При отриманні запиту сервер аналізує його, визначає відповідний маршрут та ініціює виконання логіки, описаної у відповідному модулі. Для роботи з даними сервер взаємодіє із системою управління БД, наприклад *PostgreSQL*. Для обміну повідомленнями сервера та бази даних використовується модуль *pg*. Команди модулю *pg* використовують *SQL* запити для керування БД,. отримані дані з БД управляються сервером *express.js*.

Клієнтська частина застосунку, представлена *HTML*-сторінками з інтерактивними компонентами, які надаються користувачу у відповідь сервера на *HTTP* запити. Для взаємодії з *API* клієнт надсилає звичні *URL* запити для отримання даних або відправки інформації, а сервер повертає відповідь у форматі *JSON*, яка обробляється клієнтом. *D3.js* виконує рендеринг графічних зображень на сервері, а потім відправляє ці зображення клієнту на сторінку у вигляді готових *svg* тегів.

Створюваний серверний застосунок компілюється в запущений сервер без необхідності його перезавантаження за допомогою модулю *nodemon*, який

керується *npm* скриптом. Динамічна компіляція стилів виконується за полем *devstart*.

```
"scripts": {
   "start": "node ./bin/www",
   "devstart": "nodemon ./bin/www & npm run scss",
   "serverstart": "DEBUG=add_data_DB:* npm run devstart",
   "scss": "sass --watch  scss:public/css"
 }
```

Перед тим, як почати опис розроблених основних модулів додатку, варто описати розроблені засоби, що використовуються при його реалізації.

database.js є модулем для роботи з базою даних *PostgreSQL*, що реалізує підключення, виконання запитів та обробку даних для використання в системі. Основна функціональність включає отримання даних із таблиць бази, їх нормалізацію, фільтрацію за часом та діапазонами значень. Також він виконує запити до таблиць з інформацією про інтенсивність сонячного випромінювання, температуру, тиск, вологість та інші параметри. Крім цього, файл здійснює нормалізацію значень для їх використання в алгоритмах машинного навчання чи візуалізації.

mathTools.js реалізує допоміжні математичні функції для аналізу даних, необхідних у системі обробки відповідних параметрів. Функція *elementNormalization* відповідає за масштабування значень до заданого діапазону, що полегшує роботу з неоднорідними даними. Функція *pearsonCorelation* обчислює коефіцієнт кореляції між двома масивами значень, визначаючи силу статистичного зв’язку між ними. Метод *getDeltas* розраховує зміни (дельти) між послідовними значеннями, що застосовується для аналізу динаміки змін. Функція *getCorner* визначає кут нахилу між двома точками на площині, що корисно для аналізу траєкторій або тенденцій. Нарешті, функції *formClusteringChange* та *getClusteringChanges* формують набір змін параметрів для кожного запису, що використовується для підготовки даних до кластерного аналізу.

get-pvgis.js є модулем, який реалізує взаємодію із зовнішнім *API PVGIS* для отримання погодинних даних про сонячну енергію та запис цих даних у базу

PostgreSQL. Він формує запит до *API* із заданими параметрами, отримує результат у форматі *JSON*, обробляє дані та записує їх у таблицю бази даних..

load-csv-data.js – це модуль, який завантажує дані з *CSV*-файлу у базу даних *PostgreSQL*. Він використовує *fs* для читання файлу, *csv-parser* для обробки даних і бібліотеку *config* для завантаження налаштувань. Дані фільтруються за критерієм часу (хвилина має дорівнювати 10), а потім форматуються та додаються до масиву *results*. Після завершення читання файлу кожен рядок з масиву записується до таблиці *station_data* через *SQL*-запит, включаючи перерахунок значення освітленості з люменів у ват на квадратний метр..

plot.js містить кілька класів для створення та обробки графіків, основним завданням яких є візуалізація даних за допомогою бібліотеки *D3.js*. Клас *plot2Dmask* відповідає за ініціалізацію параметрів графіка, таких як розміри, поля, та створення основної структури для *SVG*-графіків. Важливою частиною є використання *jsdom* для маніпуляцій з *DOM*-елементами на сервері, без потреби в браузері. Клас *timePlot2D* реалізує побудову графіків часу, включаючи автоматичну настройку шкал для осей та обробку форматів часу для різних інтервалів. Клас *linearPlot* створює лінійні графіки для одномірних даних, додаючи відповідні осі та лінії. Клас *corelationPlot* розширює функціональність *timePlot2D* для одночасного відображення двох набір даних, дозволяючи порівнювати кореляції між ними.

Основними модулями в WEB-додатку є

data-corelation.js, який описує функцію для обчислення кореляції між двома наборами даних. Функція *calculateCorelation* виконує кілька етапів: отримує дані з бази, обчислює коефіцієнт кореляції для самих даних та для їхніх швидкостей змін (дельт), генерує графіки для цих даних і повертає результати разом з графічним представленням. Реалізація використовує асинхронну взаємодію з базою даних через функцію *db.getUsableData*, яка отримує дані за певний період, а також перетворює дані за допомогою функцій *db.getTimeData* і *tools.getDeltas*. Перше перетворює отримані дані у формат часу, а друге – обчислює швидкість змін. Використовуються функції для обчислення кореляції Пірсона, зокрема через *tools.pearsonCorelation*, що порівнює два набори даних, спочатку для самих

значень, а потім для швидкостей їх змін. Крім того, генерація графіків для звичайної кореляції та кореляції на основі швидкостей виконана через об'єкт *corelationPlot* із модуля *plots*. Це дозволяє створювати візуальні репрезентації цих кореляцій у вигляді *SVG*-графіків.

data-normalization-visual.js направлений на виконання нормалізації даних, що знаходяться в БД, і створення двох графіків: один для візуалізації початкових даних, а інший для перевірки нормалізації. У результаті виконання цієї функції повертаються *SVG*-зображення двох графіків, що містять візуалізації для подальшого аналізу або звітності.

Модуль ***app.js*** відповідає за створення *WEB*-сервера, визначення маршрутів, підключення *middleware*, а також обробку помилок. *WEB*-сервер налаштовується для використання шаблонів *EJS*, що дозволяє генерувати динамічні *HTML*-сторінки на основі даних, які надаються сервером. Що стосується *middleware*, то використовуються кілька важливих функцій для обробки запитів. Для роботи з даними запитів сервер також налаштовує парсинг *JSON* та *URL*-кодованих даних через *express.json*() і *express.urlencoded*(), а також роботу з *cookie* завдяки *cookie-parser*. Передбачена можливість обробки статичних файлів, таких як *CSS*, відкритий клієнтський *JS* і зображення, шляхом налаштування статичного доступу через директорію *public*. Файл також визначає кілька основних маршрутів для основних функцій додатку, а саме маршрути для головної сторінки, сторінки результатів, *API* для роботи з кореляцією та маршрути для перегляду даних, де кожен із цих маршрутів обробляє відповідні запити від користувача.

3.3.2. Взаємодія між модулями серверної частини

Взаємодія між модулями, здійснюється через *HTTP*-запити, основним протоколом для передачі даних у *WEB*-середовищі, де файли взаємодіють через запити до серверної частини, які обробляються за допомогою маршрутизації *Express.js*, що дозволяє виконувати операції з базою даних *PostgreSQL* та повертати дані клієнтським запитам.

Основні елементи взаємодії між модулями *WEB*-проєкту включають:

1. ***HTTP*-запити**: Сервер обробляє *HTTP*-запити від клієнтів через різні маршрути, що визначають, які дані необхідно отримати з бази даних, а також які відповіді повернути. За допомогою *Express.js* визначаються маршрути для обробки запитів на сторінки, *API*-запити та інші ресурси.

2. **Запити до бази даних** *PostgreSQL*: Сервер за допомогою спеціальних модулів взаємодіє з *PostgreSQL*, виконуючи запити для отримання або оновлення даних. Цей процес автоматизовано через функції та методи, що підключаються до відповідних таблиць бази даних.

3. **Виклик функцій**: Файли *JS* можуть взаємодіяти між собою через виклики функцій, що забезпечує логіку обробки даних. Наприклад, файли контролерів в *Express.js* можуть викликати функції, які обробляють дані з бази або генерують графіки, а потім передають результат на клієнтську частину.

4. **Включення та рендеринг шаблонів**: Використовуючи шаблони *EJS*, *WEB*-сторінки динамічно генеруються сервером і відправляються клієнту. Це дозволяє інтегрувати дані з бази в *HTML*-сторінки на основі запитів користувача.

Таким чином, серверна частина, реалізована з використанням *PostgreSQL* та *Express.js*, здійснює взаємодію між різними частинами проєкту через запити до бази даних, обробку даних на сервері та відправку відповідей клієнту, включаючи динамічні сторінки та ресурси.

3.3.3. Дизайн інтерфейсу користувача

На головній сторінкці знаходяться *header* та *footer* та головний контент, що складається із трьох блоків. *Header* включає у собі меню з кнопками-посиланнями на головну сторінку та тести для статистичної обробки даних, які включають тест нормальності розподілу, нормування величин та їх кореляцію.

Перший блок головного контенту виконує функцію залучення користувачів та передає основну ідею ресурсу, який виражається певним девізом, де чітко та лаконічно формулюється генеральна ідея WEB-додатку. Нижче знаходиться основна концепція, яку додаток пропонує застосувати для аналізу даних (рис. 3.9).

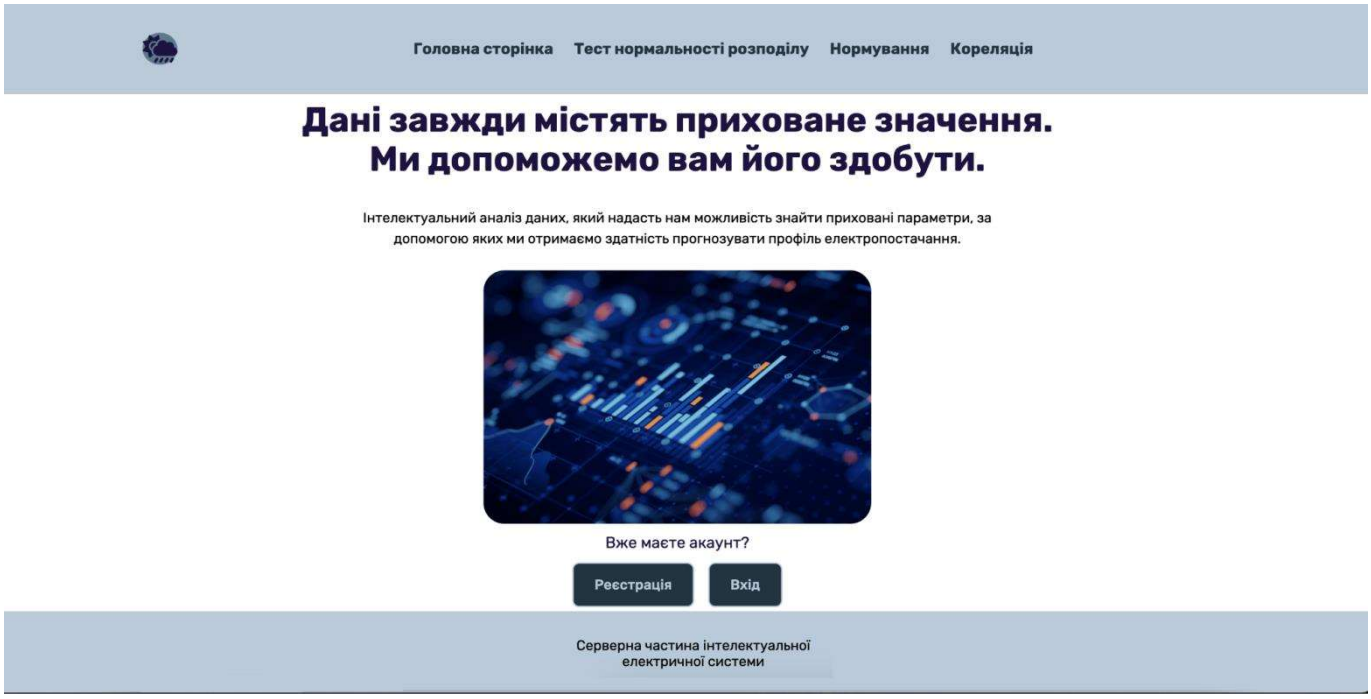
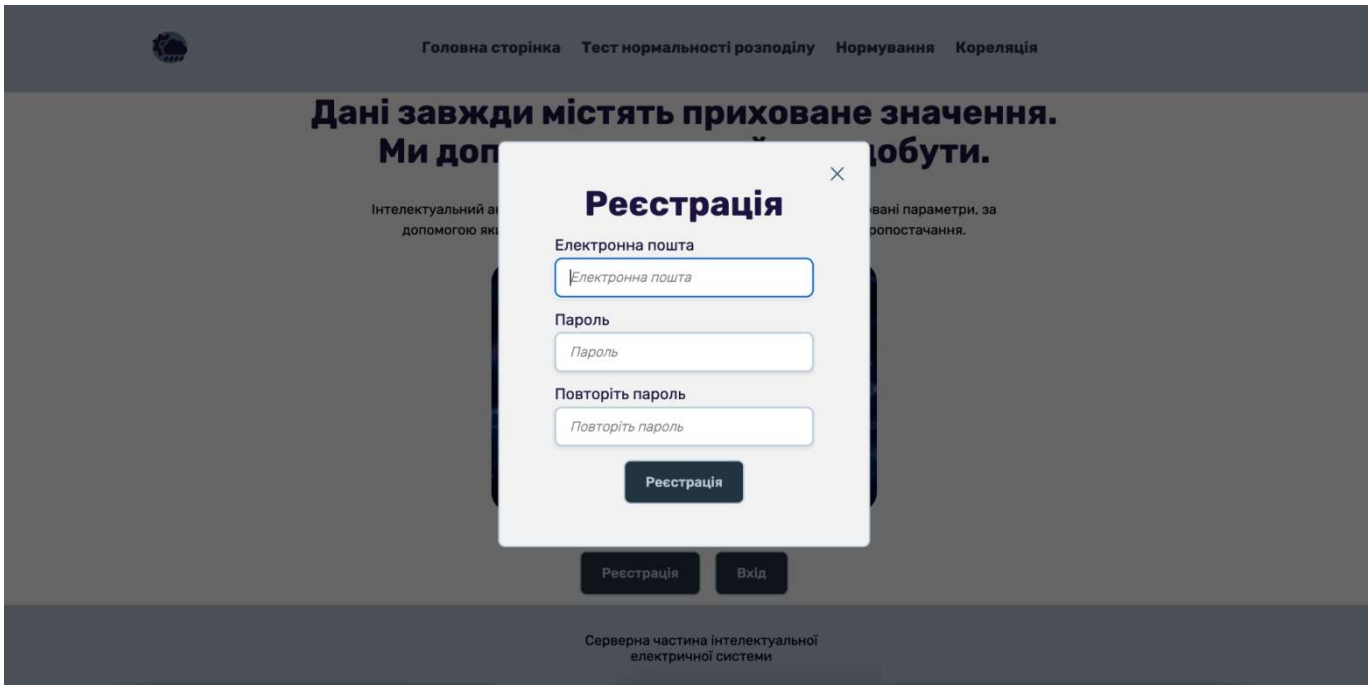


Рис. 3.9. Головна сторінка WEB-додатку

Крім цього, на сторінці знаходяться кнопки «Реєстрація» та «Вхід», за допомогою яких користувач отримує можливість відвідати сайт. Коли виконані реєстрація та авторизація, надається доступ до модульних вікон з полями, які заповнюються для того, щоб отримати досту до *WEB*-платформи (рис. 3.10).



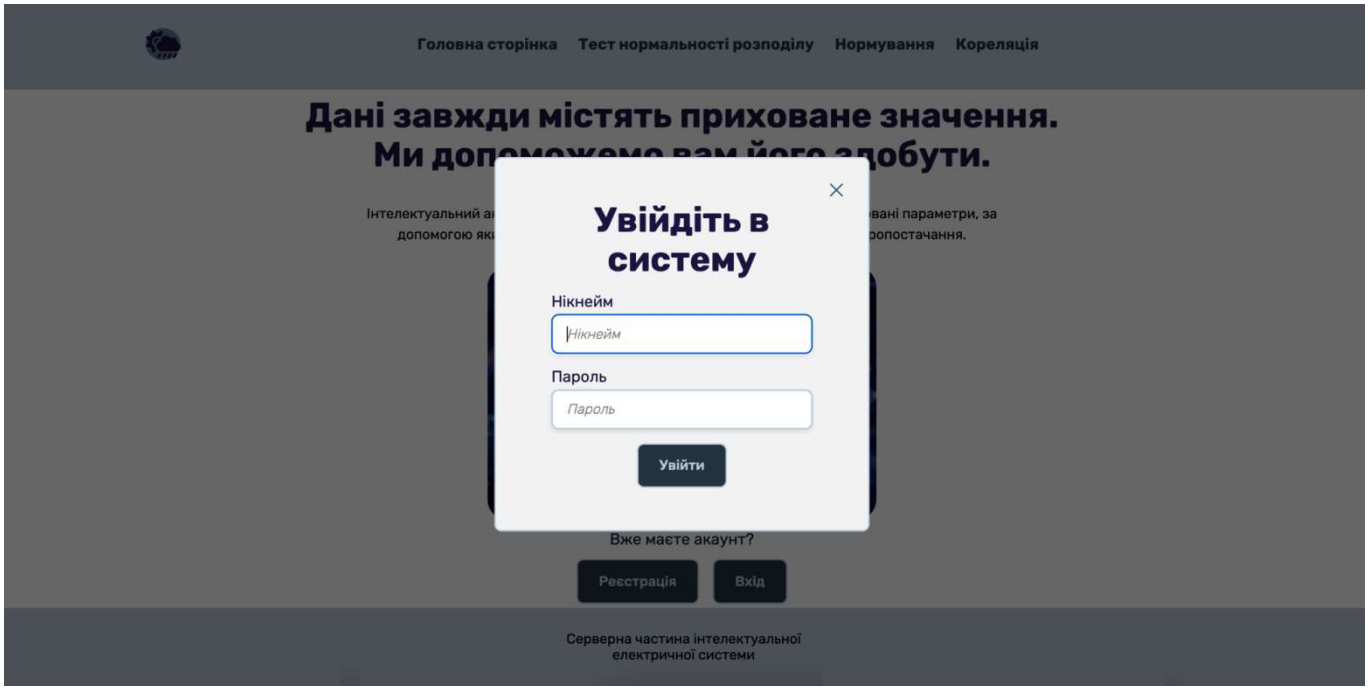


Рис. 3.10. Вікна авторизації та входу на платформу

Функціональність та можливості розробленого ПЗ були розділені на три основних блоки: Збір даних, Нормування даних та Кореляційний аналіз (рис. 3.11).

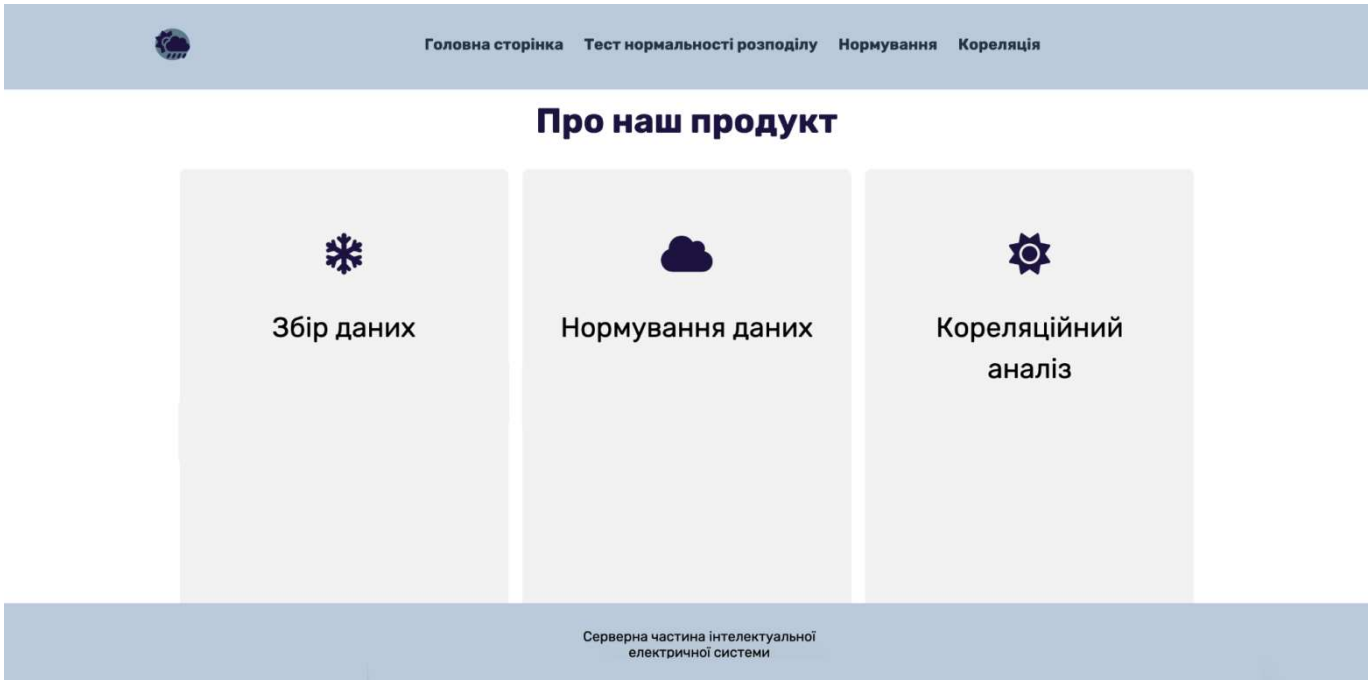


Рис. 3.11. Функціональні особливості WEB-додатку

Блок Збір даних забезпечує роботу з масивами даних, які збираються з локальної метеостанції та інтернет-сервісів (*PVGIS*) за відповідний проміжок часу. Блок Нормування даних виконує попередню обробку даних перед їх основною обробкою, яке є процесом перетворення даних у безрозмірні одиниці на проміжку [0..1] або [-1..1]. Метою нормування є приведення даних з різними одиницями виміру до однакової шкали між різними змінними (метод *Min-Max Scaling*), що дозволяє ці дані порівнювати між собою та обчислюється за формулою:

$$x' = a + \frac{(x - min(x))(b - a)}{max(x) - min(x)}$$

де a і b – нові мінімальне та максимальне значення цільового діапазону.

Блок Кореляційний аналіз є інструментом, який дозволяє виявляти зв'язки між різними параметрами, та визначати які параметри мають більший вплив на інші.

3.3.4. Тестування WEB-додатку

Для тестування використовувалися значення параметрів, отриманих за проміжок часу від 23.11.2020 до 30.11.2020. Слід зауважити, що за потреби додаток дозволяє встановлювати довільний проміжок часу, в залежності від запиту користувача.

Кореляційний аналіз значень параметру освітленості, отриманих з *PVGIS* та локальної метеостанції (*station_data*) виявив високий ступінь кореляції між двома масивами даних, який складає 0.8145 (рис. 3.12).

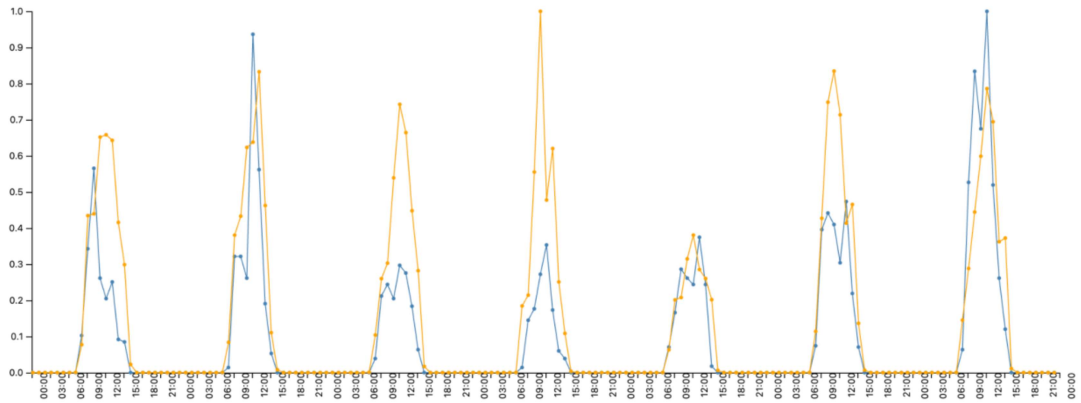


Рис. 3.12. Кореляція між значеннями величин освітленості, отриманих з *PVGIS* (синій) та *station_data* (помаранчевий)

В той же час, виявилося, що кореляція між зміною величин освітленості є низькою і складає 0.3575 (рис. 3.13).

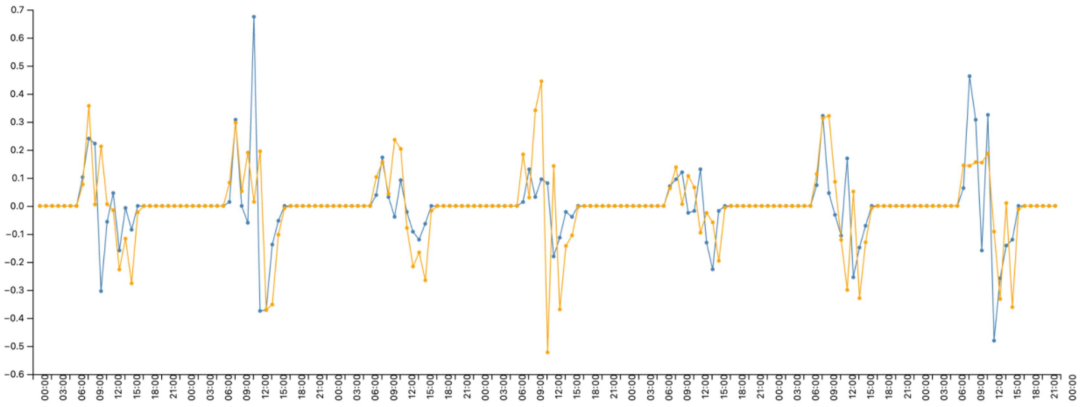


Рис. 3.13. Кореляція між зміною значень величин освітленості, отриманих з *PVGIS* (синій) та *station_data* (помаранчевий)

Не було виявлено ніякої кореляції між величинами температури та освітленості, які були отримані з *PVGIS*, обчислене значення коефіцієнту кореляції Пірсона було близьким до нуля (0.0676) (рис. 3.14).

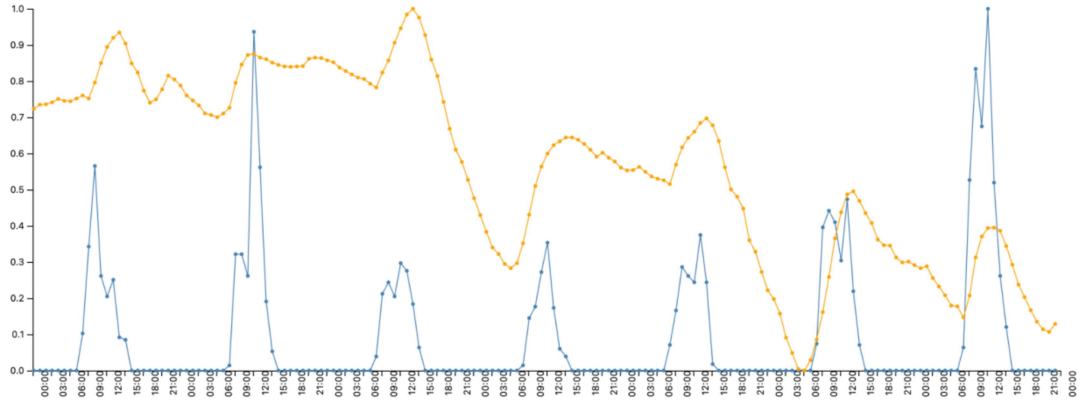


Рис. 3.14. Кореляція між величинами освітленості(синій) та температурою (помаранчевий) отриманих з *PVGIS*

Однак дослідження зміни значень цих величин виявило вищий ступінь кореляції порівняно з попереднім (0.2382) (рис. 3.15).

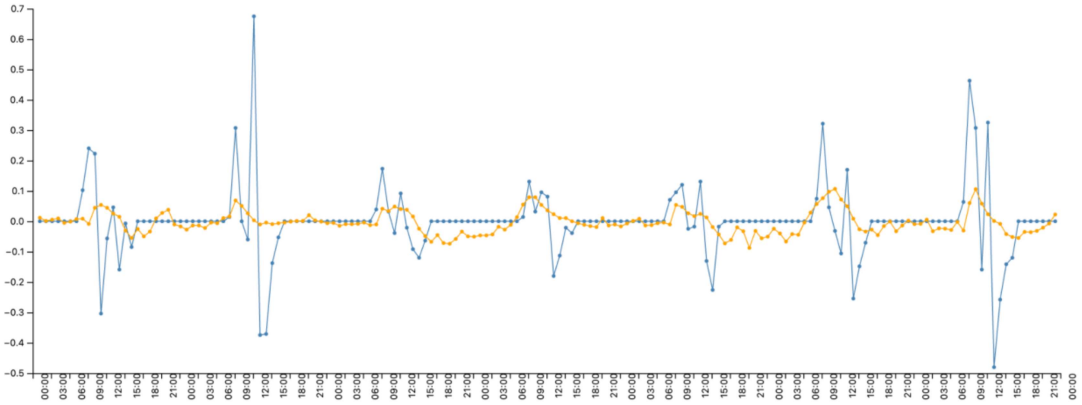


Рис. 3.15. Кореляція між зміною величин освітленості(синій) та температурою (помаранчевий) отриманих з *PVGIS*

Була виявлена кореляція між величинами потужності, яка генерується ФЕП, на основі даних отриманих з *PVGIS* та локальної метеостанції (0.7150) (рис. 3.16), проте значення зміни цих величин гірше корелюють між собою (0.1937) (рис. 3.17), що, можливо, зумовлено тим, що локальна метеостанція є більш чутливішою до чинників навколишнього середовища.

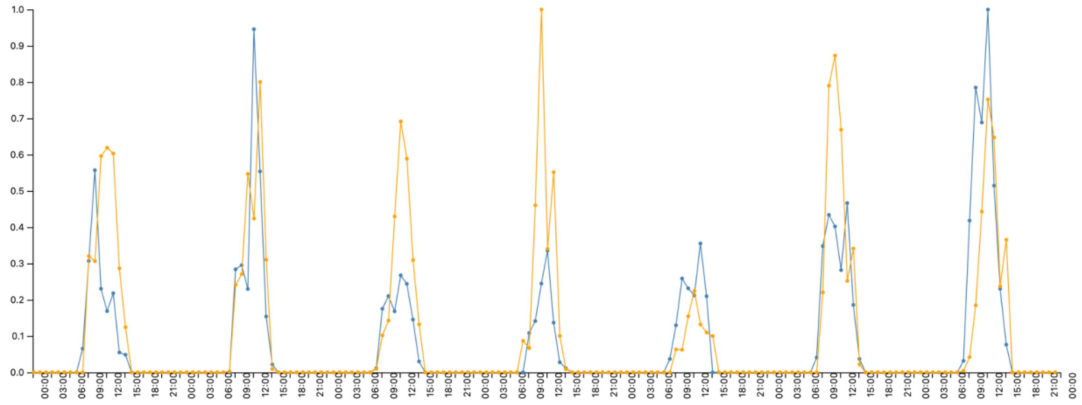


Рис. 3.16. Кореляція між значеннями величин потужностей, що генеруються ФЕП, на основі даних, отриманих з *PVGIS* (синій) та *station_data* (помаранчевий)

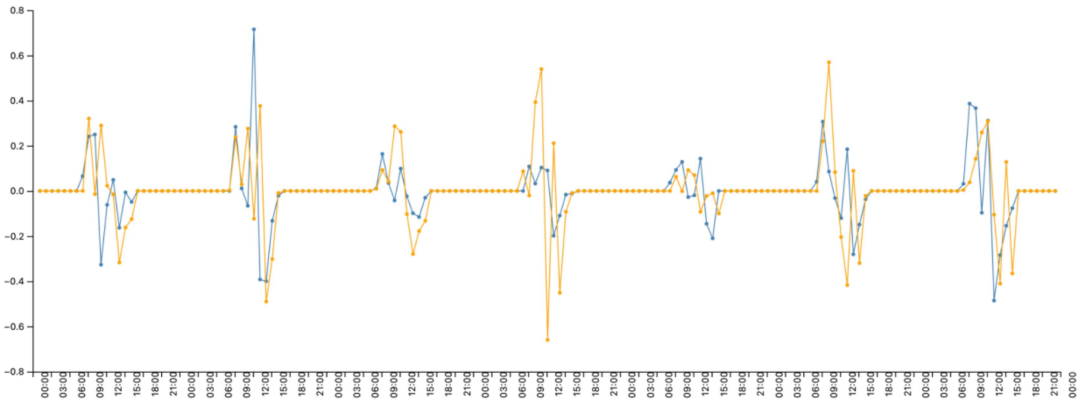


Рис. 3.17. Кореляція між змінами значень величин потужностей, що генеруються ФЕП, на основі даних, отриманих з *PVGIS* (синій) та *station_data* (помаранчевий)

Дослідження кореляції між величинами вологості та температури, отриманих з локальної метеостанції, виявило, що ці величини мають від'ємну кореляцію (-0.2933) (рис. 3.18), при цьому зміна значень величин вологості та температури показало вищий ступінь кореляції між ними (-0.6513) (рис. 3.19).

Рис. 3.18. Кореляція між значеннями величин вологості (синій) та температури (помаранчевий), отриманих з локальної метеостанції *station_data*

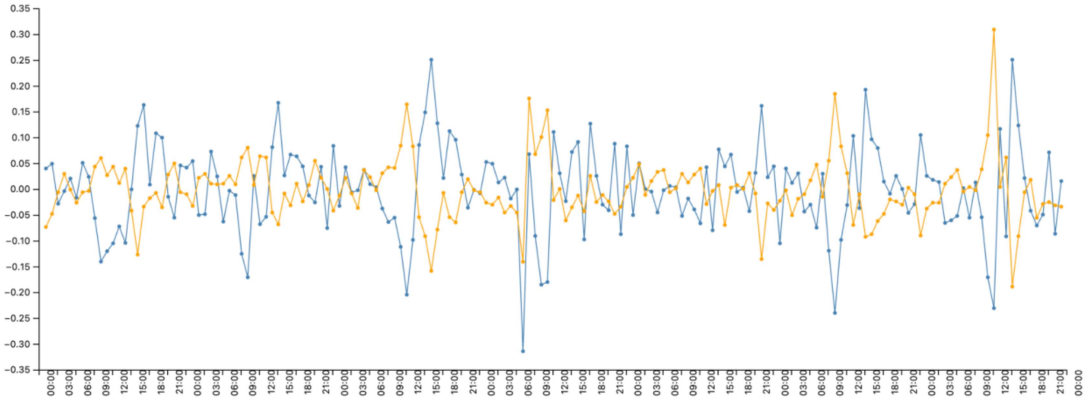


Рис. 3.19. Кореляція між зміною значень величин вологості (синій) та температури (помаранчевий), отриманих з локальної метеостанції *station_data*

Кореляція значень величин освітленості та вологості, отриманих з локальної метеостанції (-0.6667) (рис. 3.20) та їх зміна (-0.5169) (рис. 3.21), також виражаються у від'ємних значеннях, що вказує на те, що ці два параметри тісно пов'язані між собою.

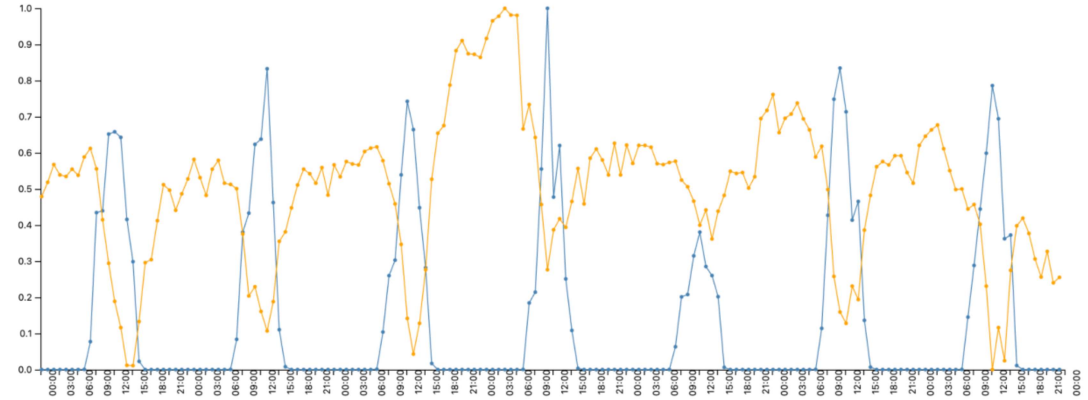


Рис. 3.20. Кореляція між значеннями величин освітленості (синій) та вологості (помаранчевий), отриманих з локальної метеостанції *station_data*

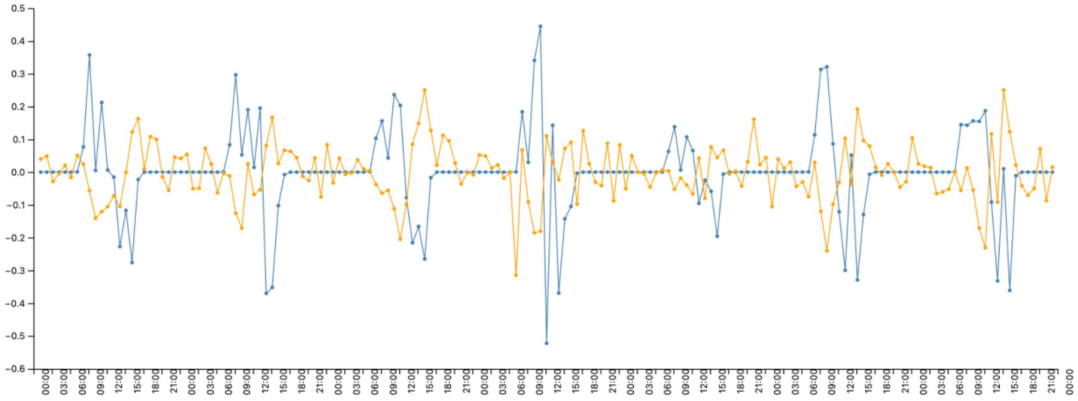


Рис. 3.21. Кореляція між зміною значень величин освітленості (синій) та вологості (помаранчевий), отриманих з локальної метеостанції *station_data*

Був виконаний кореляційний аналіз величин тиску та вологості, отриманих з локальної метеостанції за проміжок часу з 30 листопала до 23 грудня, що було обумовлено відомим фактом, що зміни тиску спричинюють інші метеорологічні зміни поступово. Отримане значення коефіцієнту кореляції Пірсона вказує на те, що ці два параметри характеризуються середнім значенням кореляції між ними (0.4032) (рис. 3.22), що не можна стверджувати для випадку, коли виконувався аналіз зміни цих величин (0.0770) (рис. 3.23).

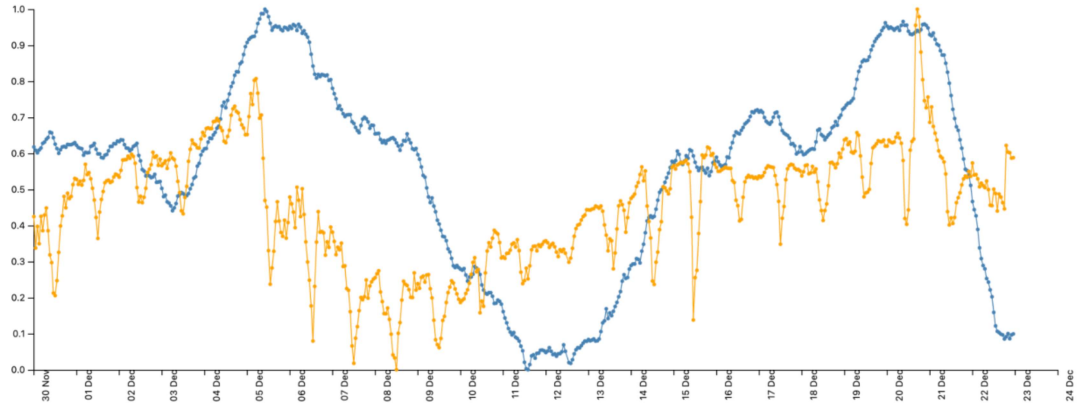


Рис. 3.22. Кореляція між значеннями величин тиску (синій) та вологості (помаранчевий), отриманих з локальної метеостанції *station_data*

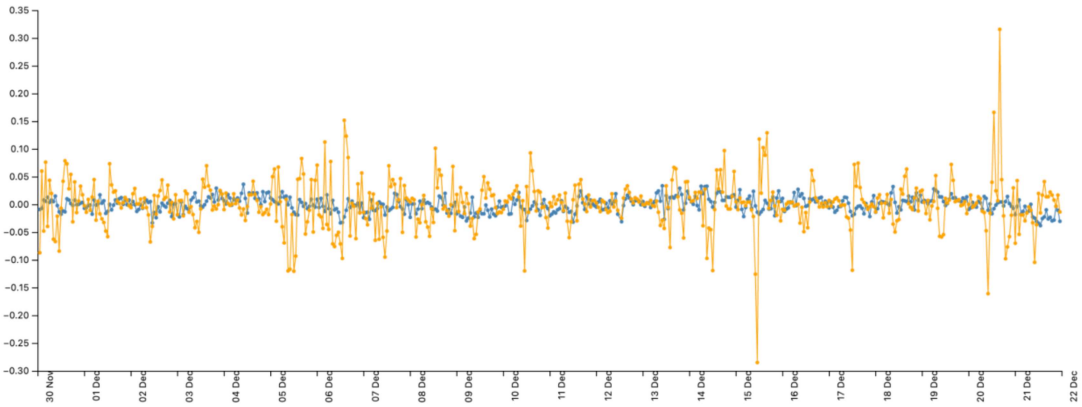


Рис. 3.23. Кореляція між змінами значень величин тиску (синій) та вологості (помаранчевий), отриманих з локальної метеостанції *station_data*

3.4. Кластерний аналіз величин

Метод статистичного аналізу який виконує групування об'єктів або даних у кластери (групи) на основі їхніх схожих атрибутивних властивостей, називається кластерний аналіз. Тобто, основною особливістю цього методу є забезпечення відповідного розбиття, в якому об'єкти згруповуються в одній сукупності, в кластері, який характеризується наявністю об’єктів з максимально схожими властивостями. Кластерний аналіз призначенний для: а) виявлення структури в масивах даних; б) поділу даних на сегменти (підгрупи) за спільними ознаками їх характеристик або поведінки; в) зменшення розмірність даних, щоб зосередитись на об’єктах окремих кластерів, а не всього набору об’єктів; г) передбачення властивостей або поведінки нових об’єктів; д) візуалізації багатовимірних даних шляхом їх наочного представлення.

Виконання кластерного аналізу на першому етапі вимагає визначення вхідних даних, які потрібно проаналізувати за відповідний визначений проміжок часу *startDate* та *endDate*, а саме дані з *PIVIGIS*, які містять інформацію про рівень освітленості, температуру, висоту сонця, швидкість вітру, та дані з локальної метеостанції, що містять інформацію про локальні параметри навколишнього середовища, в якому розміщена метеостанція, а саме значення освітленості, температури, вологості та тиску.

Другий етап кластерного аналізу включає нормування параметричних величин V в діапазоні від 0 до 1, на основі яких створюються додаткових два масиви: 1) масив зміни значень величини параметру ΔV, та 2) масив даних, що описують швидкість зміни змін значень величин. На третьому етапі в кожен момент часу знаходять відстані між точками параметричного простору (V, ΔV, $\Delta\Delta V$),, формуючи новий масив значень відстаней між параметрами. Отримавши усі можливі відстані для досліджуваного метеорологічного параметру в кожний момент часу, знаходять середньоквадратичне значення відстаней між ними. Найменше значення середньоквадратичних відстаней виявить параметри, які є найбільш тісно пов’язаними між собою, що спричинюється їх безпосереднім впливом один на одний. Після того, коли розглядати результати кластеризації в тривимірному графіку-анімації V, ΔV, $\Delta\Delta V$ для кожної окрема точки, можна виявити у наглядному представленні зміну взаємозв'язаних значень (рис. 3.24).

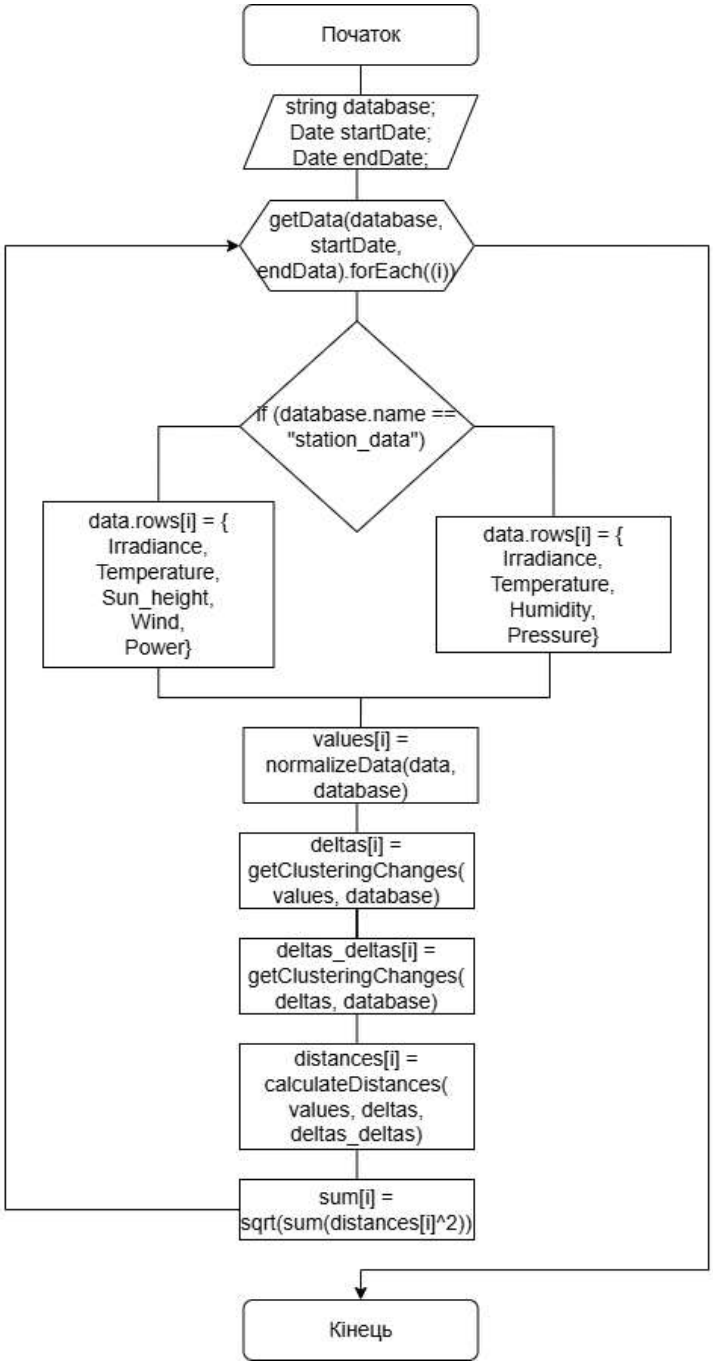


Рис. 3.24. Блок-схема виконання кластеризації

3.5. Формування асоціативних правил

Задача полягає у пошуку наборів об'єктів із усієї множини об'єктів $\boldsymbol{I} = \{\boldsymbol{i_j}\}_{\boldsymbol{j=1}}^{\boldsymbol{n}}$, що часто зустрічаються разом. Набори (підмножини) об'єктів із множини $\boldsymbol{I}$ називаються транзакціями $\boldsymbol{T} = \{\boldsymbol{i_j} | \boldsymbol{i_j} \in \boldsymbol{I}\}$. Набір транзакцій формує множину транзакцій $\boldsymbol{D} = \{\boldsymbol{T_1}, \dots, \boldsymbol{T_m}\}$, $\boldsymbol{m}$ – число транзакцій. Множини $\boldsymbol{T_k}$ можуть мати спільні елементи. Метод формулювання правил асоціацій — це метод аналізу великих масивів даних, який базується на формуванні правил, що виявляють

зв’язки між змінними або атрибутами у цих масивах даних. Число можливих асоціацій зростає експоненційно при збільшенні кількості об’єктів, тобто, якщо в масиві даних присутні k об’єктів, а всі асоціації є бінарними для випадку (**A→B** (***ЯКЩО*** A ***ТО*** B)), тоді потрібно буде проаналізувати $k \cdot 2^{k-1}$ асоціацій.

Масив транзакцій для відповідних погодних умов можна представити у вигляді множини $T=\{T_1,T_2,\ldots,T_i,\ldots,T_n\}$, яка представлена відповідною сукупністю значень $X_1,X_2,\ldots,X_i,\ldots,X_n$. Такий масив може бути стиснений шляхом пошуку транзакцій з однаковою сукупністю, використовуючи відношення еквівалентності ($\backsim$), наприклад, $(X_1 \backsim X_4 \backsim X_5) = X_{2^*}$. Тоді вихідний масив набуде вигляду $X_1,X_{2^*},\ldots,X_i,\ldots,X_n$, і в подальшому, для виявлення транзакцій з погодними параметрами, які часто повторюються, виконується попарний перетин таких сукупностей $X_1 \cap X_{2^*}, X_1 \cap X_i,\ldots,X_1 \cap X_n$; $X_{2^*} \cap X_i, X_{2^*} \cap X_n$; $X_i \cap X_n$. За результатами отрима-них перетинів формується вибірка параметрів, які є найбільш характерними для спрощеного масиву транзакцій, на основі якої формуються асоціативні бінарні відношення та розраховуються їх характеристики.

На основі даних, що отримані з PVGIS та локальної метеостанції, були сформовані визначені сукупності параметрів для формування відповідного масиву транзакцій, який був оброблений за допомогою алгоритма Apriori для знаходження асоціативних правил:

Application for PhotovoltaicPower.

Application for PlaneOfArray.

Application for SunHeightItem.

Application for AirTemperature.

Application for WindSpeedItem.

associations.RuleBuilder for [AirTemperature, PhotovoltaicPower].

associations.RuleBuilder with 0.4 support for [AirTemperature]

associations.RuleBuilder for [PlaneOfArray, SunHeightItem].

associations.RuleBuilder for [PlaneOfArray, PhotovoltaicPower].

associations.RuleBuilder for [PlaneOfArray, SunHeightItem, PhotovoltaicPower].

associations.RuleBuilder for [PlaneOfArray, AirTemperature, SunHeightItem].

associations.RuleBuilder with 0.4 support for [AirTemperature]

associations.RuleBuilder for [AirTemperature, PhotovoltaicPower].

associations.RuleBuilder with 0.4 support for [AirTemperature]

associations.RuleBuilder for [WindSpeedItem, PlaneOfArray, PhotovoltaicPower].

associations.RuleBuilder with 0.4 support for [WindSpeedItem]

associations.RuleBuilder for [WindSpeedItem, PlaneOfArray, AirTemperature].

associations.RuleBuilder with 0.4 support for [WindSpeedItem, AirTemperature]

Rule Hsun(0.0) -> Gi(0.0) support=0.51, confidence=1.0, lift=1.95, leverage=0.25}

Rule Gi(0.0) -> Ppv(0.0) support=0.51, confidence=0.97, lift=1.83, leverage=0.24}

Rule Gi(0.0) ∩ Hsun(0.0) -> Ppv(0.0) support=0.51, confidence=0.97, lift=1.83, leverage=0.24}

3.6. Висновки до розділу

У третьому розділі кваліфікаційної роботи було розглянуті ключові етапи розробки програмного додаток, для ефективного функціонування системи електропостачання з ФЕП, яка складається з розробки двох основних частин: програмного засобу для імітаційного моделювання ФЕП та WEB-додатку.

Основним елементом розробки програмного засобу для імітаційного моделювання ФЕП є розробка графічного інтерфейсу користувача, який забезпечує доступ до всіх основних функцій імітаційного моделювання, включаючи вибір типу компонентів та налаштування необхідних параметрів для подальших розрахунків.

Розробка WEB-додатку складається з розробки клієнтської та серверної частин, БД, а також засобів їхньої взаємодії. Основним елементом проєкту є файл app.js, який слугує вхідною точкою для серверної програми. Цей файл відіграє роль координатора, що ініціює запуск основних компонентів системи, Серверна частина застосунку реалізована на платформі Node.js з використанням

фреймворку Express.js. При отриманні запиту сервер аналізує його, визначає відповідний маршрут та ініціює виконання логіки, описаної у відповідному модулі. Клієнтська частина застосунку, представлена HTML-сторінками з інтерактивними компонентами, які надаються користувачу у відповідь сервера на HTTP запити.

ВИСНОВКИ

1. Оптимізація фотоелектричної системи полягає в підвищенні ефективності, надійності та терміну її функціонування за рахунок зменшення витрат на технічне обслуговування та залежності від загальної мережі. Розробка програмного додатку, який є основним інструментом, що забезпечує ефективне функціонуванняпри електричної системи з ФЕП є головною задачею кваліфікаційної роботи.

2. Були розглянуті різноманітні програмні засоби моделювання фотоелектричних систем, які дають змогу проектувати енергосистеми з бажаними параметрами, оцінювати їх функціонування та виконувати прогнозування роботи енергосистем з використанням метеорологічних даних. Найбільш актуальними виявилися програмні засоби, які надають можливість для моніторингу та аналізу динамічних змін, що відбуваються в системі під час її роботи в режимі реального часу.

3. Оптимізацію функціонування фотоелектричної системи здійснюється на основі технічного критерію, що грунтується на оцінці імовірності відключення навантаження та імовірності втрати потужності.

4. Розробка програмного додаток, для ефективного функціонування системи електропостачання з ФЕП складається з розробки програмного засобу для імітаційного моделювання ФЕП та WEB-додатку.

5. Архітектура програмного додатку є багаторівневою та модульною, що забезпечує його масштабованість, зручність у розробці та супроводі, а також адаптивність до змін вимог або інтеграції нових компонентів. На першому рівні архітектури розташований графічний інтерфейс користувача, в той час, як другий рівень архітектури складається з модуля бізнес-логіки, який відповідає за обробку введених даних і управління основними процесами. На третьому рівні архітектури виконується обробка даних, тоді як четвертий рівень архітектури забезпечує зовнішні інтеграції.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Положення про атестацію студентів та екзаменаційну комісію у Київському національному університету технологій та дизайну, затв. рішенням Вченої ради КНУТД від 04.09.2023 № 1.
2. ДСТУ 3008–95. Документація. Звіти у сфері науки й техніки. Структура і правила оформлення. Введ. 1995-23-02. ІПрІн, УкрІНТЕЇ, Головний відділ стандартизації Технічного центру НАН України, 1995. №58, 88 с.ДСТУ ГОСТ 7.1:2006 «Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання»;
3. ДСТУ 3582: 2013 «Бібліографічний опис скорочення слів і словосполучень в українській мові»;
4. ДСТУ 8302-2015 «Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання».
5. Nasle A. (2008). Electrical power system modeling, design, analysis, and reporting via a client-server application framework. Proceedings of the 2008 International Conference on Electrical Engineering and Computer Science, 08 Oct. 2008.
6. Nasle A. (2007). Automatic real-time optimization and intelligent control of electrical power distribution and transmission systems, 28 Jun. 2007.
7. Zhongxiong L., Qinglei W., Binghai H. (2018). Http (Hypertext Transfer Protocol) access method, HTTP server and HTTP system, 06 Jul. 2018.
8. ECMA-262. 15th edition, June 2024. ECMA International. https://ecma-international.org/publications-and-standards/standards/ecma-262/.
9. Qt (software). [Електронний ресурс] – (Дата звернення: 14.10.2023). Режим доступу: http://surl.li/nvhqs
10. Embedded Software Development Tools & Cross Platform IDE | Qt Creator. [Електронний ресурс] – (Дата звернення: 14.10.2023). Режим доступу: https://www.qt.io/product/development-tools.
11. Visual Studio Code - Code Editing. [Електронний ресурс] – (Дата звернення: 16.10.2023). Режим доступу: https://code.visualstudio.com/

12. INSEL—The Graphical Programming Language for the Simulation of Renewable Energy Systems. Доступний online: http://www.insel.eu/ (на 10 листопада 2024 року).
13. TRNSYS 18: A Transient System Simulation Program. Доступний online: http://sel.me.wisc.edu/trnsys (на 10 листопада 2024 року).
14. Utiliser le simulateur solaire thermique de l'INES. Доступний online: https://www.solaire-diffusion.eu/utiliser-le-simulateur-solaire-thermique-de-lines/ (на 10 листопада 2024 року).
15. HOMER software – the trusted global standard in hybrid power system modeling. Доступний online: https://homerenergy.com/ (на 10 листопада 2024 року).
16. The RETScreen Clean Energy Management Software platform. Доступний online: https://natural-resources.canada.ca/maps-tools-and-publications/tools/modelling-tools/retscreen/7465 (на 10 листопада 2024 року).
17. System Advisor Model (SAM). Naional Renewable Energy Laboratory. Доступний online: https://sam.nrel.gov/ (на 10 листопада 2024 року).
18. Marion, B.; Anderberg, M. PVWATTS-An Online Performance Calculator for Grid-Connected PV Systems. In Proceedings of the Solar Conference, Madison, WI, USA, 16–21 June 2000; pp. 119–124.
19. Klise, G.T.; Stein, J.S. Models Used to Assess the Performance of Photovoltaic Systems; Report SAND2009-8258; Sandia National Laboratories: Albuquerque, NM, USA, 2009.
20. PVsyst: Software for Photovoltaic System. Доступний online: https://www.pvsyst.com/ (на 10 листопада 2024 року).
21. PV*SOL Take your solar installations to the next level of efficiency. Доступний online: https://pvsol.software/en/ (на 10 листопада 2024 року).
22. PV F-CHART comprehensive photovoltaic system analysis and design program. Доступний online: https://fchartsoftware.com/pvfchart/ (на 10 листопада 2024 року).
23. Photovoltaic Geographical Information System (PVGIS). Доступний online: https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis_en (на 10 листопада 2024 року).

24. SolarGIS Most accurate data and software for PV plant investments. Доступний online: https://solargis.com/ (на 10 листопада 2024 року).
25. BlueSol Photovoltaic design software. Доступний online: http://www.bluesolpv.com/ (на 10 листопада 2024 року).
26. PYLON #1 Solar Design & CRM Software for fast-growing solar businesses. Доступний online: https://getpylon.com/ (на 10 листопада 2024 року).
27. Abbassi, R., Abbassi, A., Jemli, M., Chebbi, S., 2018. Identification of unknown parameters of solar cell models: A comprehensive overview of available approaches. Renew. Sustain. Energy Rev. 90, 453–474.
28. Brano, V.L., Ciulla, G., 2013. An efficient analytical approach for obtaining a five parameters model of photovoltaic modules using only reference data. Appl. Energy 111, 894–903.
29. Chegaar, M., Azzouzi, G., Mialhe, P., 2006. Simple parameter extraction method for illuminated solar cells. Solid-State Electron. 50 (7), 1234–1237.
30. Humada, A.M., Hojabri, M., Mekhilef, S., Hamada, H.M., 2016b. Solar cell parameters extraction based on single and double-diode models: A review. Renew. Sustain. Energy Rev. 56, 494–509.

Бекенд логіка

checkdata-shapiroTest-command.js

```
const config = require('config');
const db = require(`${config.get('Path.utils')}database.js`);

const normalInverse = (x) => {
        const a1 = -3.969683028665376e+1;
        const a2 = 2.209460984245205e+2;
        const a3 = -2.759285104469687e+2;
        const a4 = 1.383577518672690e+2;
        const a5 = -3.066479806614716e+1;
        const a6 = 2.506628277459239e+0;

        const b1 = -5.447609879822406e+1;
        const b2 = 1.615858368580409e+2;
        const b3 = -1.556989798598866e+2;
        const b4 = 6.680131188771972e+1;
        const b5 = -1.328068155288572e+1;

        const c1 = -7.784894002430293e-3;
        const c2 = -3.223964580411365e-1;
        const c3 = -2.400758277161838e+0;
        const c4 = -2.549732539343734e+0;
        const c5 = 4.374664141464968e+0;
        const c6 = 2.938163982698783e+0;

        const d1 = 7.784695709041462e-3;
        const d2 = 3.224671290700398e-1;
        const d3 = 2.445134137142996e+0;
        const d4 = 3.754408661907416e+0;

        let q, r, y;
```

```
        if (x < 0.02425) {
                q = Math.sqrt(-2 * Math.log(x));
                y = (((((c1 * q + c2) * q + c3) * q + c4) * q + c5) * q + c6) / ((((d1 * q + d2) * q + d3) *
q + d4) * q + 1);
        } else if (x < 1 - 0.02425) {
                q = x - 0.5;
                r = q * q;
                y = (((((a1 * r + a2) * r + a3) * r + a4) * r + a5) * r + a6) * q / (((((b1 * r + b2) * r + b3)
* r + b4) * r + b5) * r + 1);
        } else {
                q = Math.sqrt(-2 * Math.log(1 - x));
                y = -(((((c1 * q + c2) * q + c3) * q + c4) * q + c5) * q + c6) / ((((d1 * q + d2) * q + d3) *
q + d4) * q + 1);
        }

        return y;
};

const shapiroWilkTest = (data) => {
  let result = {};
        const xx = data.sort((a, b) => a - b);
        const meanValue = data.reduce((a, b) => a + b) / data.length;
        const n = data.length;
        const u = 1 / Math.sqrt(n);

        const m = new Array();
        for (let i = 1; i <= n; i++) {
                m.push(normalInverse((i - 3/8) / (n + 1/4)));
        }

        const md = m.reduce((sum, value) => sum + Math.pow(value, 2), 0);
  const factor = 1 / Math.sqrt(md);
        const c = m.map((value) => value * factor);

        const an = -2.706056 * Math.pow(u, 5) + 4.434685 * Math.pow(u, 4) - 2.071190 *
Math.pow(u, 3) - 0.147981 * Math.pow(u, 2) + 0.221157 * u + c[n - 1];
```

```
        const ann = -3.582633 * Math.pow(u, 5) + 5.682633 * Math.pow(u, 4) - 1.752461 *
Math.pow(u, 3) - 0.293762 * Math.pow(u, 2) + 0.042981 * u + c[n - 2];

        let phi;



        if (n > 5) {
  let denominator = 1 - 2 * Math.pow(an, 2) - 2 * Math.pow(ann, 2);

  if (denominator === 0) {
    console.error("Знаменник дорівнює нулю!");
  } else {
    phi = (md - 2 * Math.pow(m[n - 1], 2) - 2 * Math.pow(m[n - 2], 2)) / denominator;
  }
        } else {
                phi = (md - 2 * Math.pow(m[n - 1], 2)) / (1 - 2 * Math.pow(an, 2));
        }

        const a = new Array();
        if (n > 5) {
                a.push(-an);
                a.push(-ann);

                for (let i = 2; i < n - 2; i++) {
                        a.push(m[i] * Math.pow(phi, -1/2));
                }

                a.push(ann);
                a.push(an);
        }
 else {
                a.push(-an);

                for (let i = 1; i < n - 1; i++) {
                        a.push(m[i] * Math.pow(phi, -1/2));
                }
```

```
            a.push(an);
        }

  let denominator = xx.reduce((sum, value) => sum + Math.pow(value - meanValue, 2), 0);
        result.w = Math.pow(a.map((aValue, index) => aValue * xx[index]).reduce((sum, value) =>
sum + value), 2) / denominator;

  let g, mu, sigma;

        if (n < 12) {
            let gamma = 0.459 * n - 2.273;
            g = - Math.log(gamma - Math.log(1 - result.w));
            mu = -0.0006714 * Math.pow(n, 3) + 0.025054 * Math.pow(n, 2) - 0.39978 * n +
0.5440;
            sigma = Math.exp(-0.0020322 * Math.pow(n, 3) + 0.062767 * Math.pow(n, 2) -
0.77857 * n + 1.3822);
        } else {
            let u = Math.log(n);
            g = Math.log(1 - result.w);
            mu = 0.0038915 * Math.pow(u, 3) - 0.083751 * Math.pow(u, 2) - 0.31082 * u - 1.5851;
            sigma = Math.exp(0.0030302 * Math.pow(u, 2) - 0.082676 * u - 0.4803);
        }

        const z = (g - mu) / sigma;
        result.p = 1 - 0.5 * (1 + erf(z  / Math.sqrt(2)));

        return result;
}

const erf = (z) => {
        let term;
        let sum = 0;
        let n = 0;
        do {
            term = Math.pow(-1, n) * Math.pow(z, 2 * n + 1) / calculateFactorial(n) / (2 * n + 1);
```

```
            sum = sum + term;
            n++;
        } while (Math.abs(term) > 0.000000000001);
        return sum * 2 / Math.sqrt(Math.PI);
};

const calculateFactorial = (n) => {
        let result = 1;

        for (let i = 2; i <= n; i++) {
            result = result * i;
        }

        return result;
}

db.startConnection();

const getIrradianceStatsForDay = async (day) => {
  const query = `
    SELECT
        "Irradiance"
    FROM
        pvgis_api
    WHERE
        "Day" = $1 AND
        "Irradiance" <> 0
  `;

  const result = await db.getColumn(query, [day]);
  return result.map(row => parseFloat(row.Irradiance));
}

const comparePandAlpha = (alpha, p) => {
  return p > alpha;
};
```

```
// main part

(async () => {
  const day = process.argv[2];
  const stats = await getIrradianceStatsForDay(day);

  const shapiroWilkResult = shapiroWilkTest(stats);
  console.log(`W: ${shapiroWilkResult.w}`);
  console.log(`p: ${shapiroWilkResult.p}`);

  const alpha = process.argv[3];

  if (comparePandAlpha(alpha, shapiroWilkResult.p)) {
    console.log(`Дані за ${day} день нормально розподілені.`);
  }
  else {
    console.log(`Нормальність даних з очікуванням похибки в ${alpha} не приймається за ${day}
день.`);
  }

  db.endConnection();
})();
```

data-corellation-command.js

```
const config = require('config');
const utils = config.get('Path.utils');
const db =  require(`${utils}database.js`);
const plots = require(`${utils}plots.js`);
const tools = require(`${utils}mathTools.js`);

db.startConnection();

(async () => {
  const baseTable = process.argv[2];
  const baseColumn = process.argv[3];
```

```
  const corelationTable = process.argv[4];
  const corelationColumn = process.argv[5];
  const startDate = process.argv[6];
  const endDate = process.argv[7];

  const dbBaseData = await db.getUsableData(baseTable, baseColumn, startDate, endDate);
  const dbCorellationData = await db.getUsableData(corelationTable, corelationColumn, startDate,
endDate);
  const baseData = db.getTimeData(dbBaseData, baseColumn);
  const corellationData = db.getTimeData(dbCorellationData, corelationColumn);
  const baseDataSpeed = tools.getDeltas(baseData);
  const corellationDataSpeed = tools.getDeltas(corellationData);

  const pearsonCoef = tools.pearsonCorelation(baseData.map(obj => obj.y), corellationData.map(obj
=> obj.y));
  const speedPearsonCoef = tools.pearsonCorelation(baseDataSpeed.map(obj => obj.y),
corellationDataSpeed.map(obj => obj.y));
  console.log("Коефіцієнт кореляції Пірсона: " + pearsonCoef);
  console.log("Коефіцієнт кореляції Пірсона швидкості змін: " + speedPearsonCoef);

  const normalPlot = new plots.corelationPlot(baseData, corellationData);
  await normalPlot.init();
  normalPlot.writeTimePlotFile("corellation");

  const speedPlot = new plots.corelationPlot(baseDataSpeed, corellationDataSpeed);
  await speedPlot.init();
  speedPlot.writeTimePlotFile("speed");

  db.endConnection();
})();
```

data-normalization-vizual.js

```
const config = require('config');
const utils = config.get('Path.utils');
const plots = require(`${utils}plots.js`);
const db = require(`${utils}database.js`);
```

```
const calculateNormalization = async (t, c, s, e) => {
  db.startConnection();
  const table = t;
  const chosenColumn = c;
  const startDate = s;
  const endDate = e;

  const dbData = await db.getUsableData(table, chosenColumn, startDate, endDate);
  const plotData = db.getTimeData(dbData, chosenColumn);
  const normalizeCheckData = db.checkNormalization(dbData, chosenColumn);

  const plot = new plots.timePlot2D(plotData);
  await plot.init();
  plot.writeTimePlotFile();

  const normPlot = new plots.linearPlot(normalizeCheckData);
  await normPlot.init(500, 500);
  normPlot.writePlotFile();

  return {plot:plot.svgTag, plotCheck:normPlot.svgTag};
};

module.exports = {
  calculateNormalization
}
```

app.py

```
import sys
import json
from PySide6.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout,
QFormLayout, QLineEdit, QPushButton, QFileDialog
import matlab.engine

class PVModelingApp(QMainWindow):
    def __init__(self):
```

```
super().__init__()

self.setWindowTitle("PV Modeling")

self.main_widget = QWidget(self)
self.setCentralWidget(self.main_widget)
self.layout = QVBoxLayout(self.main_widget)

self.form_layout = QFormLayout()
self.layout.addLayout(self.form_layout)

self.Gn_input = QLineEdit(self)
self.Tn_input = QLineEdit(self)
self.Vocn_input = QLineEdit(self)
self.Vmp_input = QLineEdit(self)
self.Imp_input = QLineEdit(self)
self.Iscn_input = QLineEdit(self)
self.Ki_input = QLineEdit(self)
self.Kv_input = QLineEdit(self)
self.Ns_input = QLineEdit(self)

self.model_name_input = QLineEdit(self)

self.form_layout.addRow("Model Name:", self.model_name_input)

self.form_layout.addRow("Nominal Irradiance (Gn):", self.Gn_input)
self.form_layout.addRow("Nominal Temperature (Tn):", self.Tn_input)
self.form_layout.addRow("Open Circuit Voltage (Vocn):", self.Vocn_input)
self.form_layout.addRow("Maximum Power Voltage (Vmp):", self.Vmp_input)
self.form_layout.addRow("Maximum Power Current (Imp):", self.Imp_input)
self.form_layout.addRow("Short Circuit Current (Iscn):", self.Iscn_input)
self.form_layout.addRow("Temperature Coefficient of Current (Ki):", self.Ki_input)
self.form_layout.addRow("Temperature Coefficient of Voltage (Kv):", self.Kv_input)
self.form_layout.addRow("Number of Cells in Series (Ns):", self.Ns_input)

self.run_button = QPushButton("Run Model", self)
```

```
        self.run_button.clicked.connect(self.run_model)
        self.layout.addWidget(self.run_button)

        self.save_button = QPushButton("Save Data", self)
        self.save_button.clicked.connect(self.save_data)
        self.layout.addWidget(self.save_button)

        self.load_button = QPushButton("Load Data", self)
        self.load_button.clicked.connect(self.load_data)
        self.layout.addWidget(self.load_button)

        self.load_data()

        self.setGeometry(100, 100, 800, 600)

    def save_data(self):
        data = {
            "Model Name": self.model_name_input.text(),
            "Gn": self.Gn_input.text(),
            "Tn": self.Tn_input.text(),
            "Vocn": self.Vocn_input.text(),
            "Vmp": self.Vmp_input.text(),
            "Imp": self.Imp_input.text(),
            "Iscn": self.Iscn_input.text(),
            "Ki": self.Ki_input.text(),
            "Kv": self.Kv_input.text(),
            "Ns": self.Ns_input.text()
        }

        options = QFileDialog.Options()
        file_path, _ = QFileDialog.getSaveFileName(self, "Save Data", "", "JSON Files (*.json);;All Files
(*)", options=options)
        if file_path:
            try:
                with open(file_path, 'w') as f:
                    json.dump(data, f, indent=4)
```

```
            print(f"Data saved to {file_path}")
        except Exception as e:
            print(f"Error saving data: {e}")

    def load_data(self):
        options = QFileDialog.Options()
        file_path, _ = QFileDialog.getOpenFileName(self, "Load Data", "", "JSON Files (*.json);;All Files (*)", options=options)
        if file_path:
            try:
                with open(file_path, 'r') as f:
                    data = json.load(f)
                    self.model_name_input.setText(data.get("Model Name", ""))
                    self.Gn_input.setText(data.get("Gn", ""))
                    self.Tn_input.setText(data.get("Tn", ""))
                    self.Vocn_input.setText(data.get("Vocn", ""))
                    self.Vmp_input.setText(data.get("Vmp", ""))
                    self.Imp_input.setText(data.get("Imp", ""))
                    self.Iscn_input.setText(data.get("Iscn", ""))
                    self.Ki_input.setText(data.get("Ki", ""))
                    self.Kv_input.setText(data.get("Kv", ""))
                    self.Ns_input.setText(data.get("Ns", ""))
                print(f"Data loaded from {file_path}")
            except Exception as e:
                print(f"Error loading data: {e}")

    def run_model(self):
        Gn = float(self.Gn_input.text())
        Tn = float(self.Tn_input.text())
        Vocn = float(self.Vocn_input.text())
        Vmp = float(self.Vmp_input.text())
        Imp = float(self.Imp_input.text())
        Iscn = float(self.Iscn_input.text())
        Ki = float(self.Ki_input.text())
        Kv = float(self.Kv_input.text())
        Ns = int(self.Ns_input.text())
```

```
        eng = matlab.engine.start_matlab()

        s = eng.genpath('.')
        eng.addpath(s, nargout=0)
        eng.method1(Gn, Tn, Vocn, Vmp, Imp, Iscn, Ki, Kv, Ns, nargout=10)
        input("Click Enter to proceed...")
        eng.quit()

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = PVModelingApp()
    window.show()
    sys.exit(app.exec())
```

get_server_calc.py

```
import sys
import requests
import json
from PySide6.QtWidgets import QApplication, QWidget, QVBoxLayout, QLabel, QComboBox,
QPushButton, QDateEdit, QTextEdit
from PySide6.QtCore import QDate

class ModelApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Параметри моделювання")
        self.setGeometry(100, 100, 400, 400)

        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()

        self.table_label = QLabel("Оберіть назву таблиці:")
        self.table_combobox = QComboBox()
```

```
        self.table_combobox.addItem("pvgis_api")
        self.table_combobox.addItem("station_data")

        self.start_date_label = QLabel("Дата початку:")
        self.start_date_edit = QDateEdit(self)
        self.start_date_edit.setDate(QDate(2020, 1, 1))
        self.start_date_edit.setCalendarPopup(True)

        self.end_date_label = QLabel("Дата кінця:")
        self.end_date_edit = QDateEdit(self)
        self.end_date_edit.setDate(QDate(2020, 1, 5))
        self.end_date_edit.setCalendarPopup(True)

        self.submit_button = QPushButton("Отримати дані")
        self.submit_button.clicked.connect(self.fetch_data)

        self.results_label = QTextEdit(self)
        self.results_label.setReadOnly(True)
        self.results_label.setPlaceholderText("Результати будуть тут")

        layout.addWidget(self.table_label)
        layout.addWidget(self.table_combobox)
        layout.addWidget(self.start_date_label)
        layout.addWidget(self.start_date_edit)
        layout.addWidget(self.end_date_label)
        layout.addWidget(self.end_date_edit)
        layout.addWidget(self.submit_button)
        layout.addWidget(self.results_label)

        self.setLayout(layout)

    def fetch_data(self):
        table = self.table_combobox.currentText()
        start_date = self.start_date_edit.date().toString("yyyy-MM-dd")
        end_date = self.end_date_edit.date().toString("yyyy-MM-dd")
```

```
        url = f"http://127.0.0.1:3000/view?columns=Irradiance-
Temperature&table={table}&startDate={start_date}&endDate={end_date}"

        try:
            response = requests.get(url)
            response.raise_for_status()
            data = response.json()
            formatted_data = json.dumps(data, indent=4)
            self.results_label.setPlainText(formatted_data)

        except requests.exceptions.RequestException as e:
            self.results_label.setText(f"Помилка при отриманні даних: {str(e)}")

    def closeEvent(self, event):
        event.accept()

def main():
    app = QApplication(sys.argv)
    window = ModelApp()
    window.show()
    sys.exit(app.exec())

if __name__ == "__main__":
    main()
```

method1.m

```
function [I, V, P, Rs, Rp, ideality_coef, T, G, Pmax_m, error] = method1(G, T, Vocn, Vmp, Imp, Iscn,
Ki, Kv, Ns)
k = 1.3806503e-23;
q = 1.60217646e-19;

G = double(G);
T = double(T) + 273.15;
Tn = 25 + 273.15;
Gn = 1000;
```

```
Vocn = double(Vocn);
Iscn = double(Iscn);
Vmp = double(Vmp);
Imp = double(Imp);
Pmax_e = Vmp * Imp;
Ki = double(Ki);
%Kv = double(Kv);
Ns = double(Ns);

algorithm_speed = 0.001;
model_prec = 0.0001;
plot_points = 100;
max_cycle_number = 500000;
isDebugging = 0;

Rs_max = (Vocn - Vmp) / Imp;
Rp_min = Vmp / (Iscn - Imp) - Rs_max;
Rs = 0;
Rp = Rp_min;
Vt  = k * T  / q;

error = Inf;
cycle_index = 0;
ideality_coef = 1;

while (error > model_prec) && (Rp > 0) && (cycle_index < max_cycle_number)
    cycle_index = cycle_index + 1;

    delta_T_Tn = T-Tn;
    nominal_Ipv = (Rs + Rp) / Rp * Iscn;
    actual_Ipv = (nominal_Ipv + Ki * delta_T_Tn) * G / Gn;
    actual_Isc = (Iscn + Ki * delta_T_Tn) * G / Gn;

    Ion = (actual_Ipv - Vocn / Rp) / (exp(Vocn / Vt / ideality_coef / Ns) - 1);
    Io = Ion;
```

```
    Rs  = Rs + algorithm_speed;
    Rp_ = Rp;
    Rp = Vmp * (Vmp + Imp * Rs) / (Vmp * actual_Ipv - Vmp * Io * exp((Vmp + Imp * Rs) / Vt / Ns /
ideality_coef) + Vmp * Io - Pmax_e);

    clear V
    clear I

    V_step = Vocn / plot_points;
    V = 0:V_step:Vocn;
    I = zeros(1, size(V, 2));
    options = optimoptions('fsolve', 'Display', 'none');

    for j = 1:size(V, 2)

        current_eq = @(I_j) actual_Ipv ...
            - Io * (exp((V(j) + I_j * Rs) / (Vt * Ns * ideality_coef)) - 1) ...
            - (V(j) + I_j * Rs) / Rp - I_j;

        I(j) = fsolve(current_eq, I(j), options);
    end



    if (isDebugging)
        figure(1)
        grid on
        hold on
        title('I-V curve - Adjusting Rs and Rp');
        xlabel('V [V]');
        ylabel('I [A]');
        xlim([0 Vocn]);
        ylim([0 Iscn]);
        plot(V,I,'LineWidth',2,'Color','k')
        plot([0 Vmp Vocn],[Iscn Imp 0],'o','LineWidth',2,'MarkerSize',5,'Color','k')

        figure(2)
```

```
        grid on
        hold on
        title('P-V curve - Adjusting peak power');
        xlabel('V [V]');
        ylabel('P [W]');
        xlim([0 Vocn])
        ylim([0 Vmp*Imp]);
    end

    P = (actual_Ipv-Io*(exp((V+I.*Rs)/Vt/Ns/ideality_coef)-1)-(V+I.*Rs)/Rp).*V;
    Pmax_m = max(P);
    error = (Pmax_m-Pmax_e);

    if (isDebugging)
        plot(V,P,'LineWidth',2,'Color','k')
        plot([0 Vmp Vocn],[0 Vmp*Imp 0],'o','LineWidth',2,'MarkerSize',5,'Color','k')
    end
end

if (Rp < 0) Rp = Rp_
end

figure(3)
grid on
hold on
title('Adjusted I-V curve');
xlabel('V [V]');
ylabel('I [A]');
xlim([0 max(V) * 1.1]);
ylim([0 max(I) * 1.1]);
plot(V,I,'LineWidth',2,'Color','k')
plot([0 Vmp Vocn ],[Iscn Imp 0 ],'o','LineWidth',2,'MarkerSize',5,'Color','k')

figure(4)
grid on
hold on
```

```
title('Adjusted P-V curve');
xlabel('V [V]');
ylabel('P [W]');
xlim([0 Vocn * 1.1]);
ylim([0 Vmp * Imp * 1.1]);
plot(V,P,'LineWidth',2,'Color','k')
plot([0 Vmp Vocn ],[0 Pmax_e 0 ],'o','LineWidth',2,'MarkerSize',5,'Color','k')

disp(sprintf('Complete model\n'));
disp(sprintf('     Rp = %f',Rp));
disp(sprintf('     Rs = %f',Rs));
disp(sprintf('      a = %f',ideality_coef));
disp(sprintf('      T = %f',T-273.15));
disp(sprintf('      G = %f',G));
disp(sprintf(' Pmax,m = %f  (model)',Pmax_m));
disp(sprintf(' Pmax,e = %f  (experimental)',Pmax_e));
disp(sprintf('    tol = %f',model_prec));
disp(sprintf('P_error = %f',error));
disp(sprintf('    Ipv = %f',actual_Ipv));
disp(sprintf('    Isc = %f',actual_Isc));
disp(sprintf('    Ion = %g',Ion));
disp(sprintf('\n\n'));

end
```

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Факультет мехатроніки та комп’ютерних технологій
Кафедра комп’ютерних наук

Тема кваліфікаційної роботи:

Розроблення програмного забезпечення для системи електропостачання з фотоелектричними панелями

студент групи МгІТ-1-23
Кравченко Микола Сергійович

Керівник: д. фіз.-мат. н., професор Краснитський С.М.

Інтелектуальна електрична система

IEC – це взаємозв'язана енергетична система з відповідними технологіями, які використовуються для апаратної автоматизації та програмного інтелектуального керування з метою підвищення ефективності, надійності, стійкості та гнучкості функціонування електричної системи при виробництві, передачі, розподілі та споживанні електроенергії

Актуальність теми

Об’єкт дослідження – процес збирання, зберігання та обробки даних для ефективного функціонування системи електропостачання з ФЕП.

Предмет дослідження – програмний додаток для збирання, зберігання та обробки даних для ефективного функціонування системи електропостачання з ФЕП.

Мета кваліфікаційної роботи – розробка програмного засобу для збирання, зберігання та обробки даних для ефективного функціонування системи електропостачання з ФЕП.

Призначення цього програмного засобу полягає в забезпеченні необхідних даних та методів для виконання оптимального функціонування фотоелектричної системи на основі технічного критерію оптимізації. Оптимізація фотоелектричної системи здійснюється в результаті моніторингу та аналізу її функціонування.

Технічний критерій оптимізації

Імовірність відключення навантаження (ІВН)

$$\text{ІВН} = \frac{\sum_{i=1}^{8760} \text{Дефіцит потужності}_i}{\sum_{i=1}^{8760} \text{Потреба потужності}_i}$$

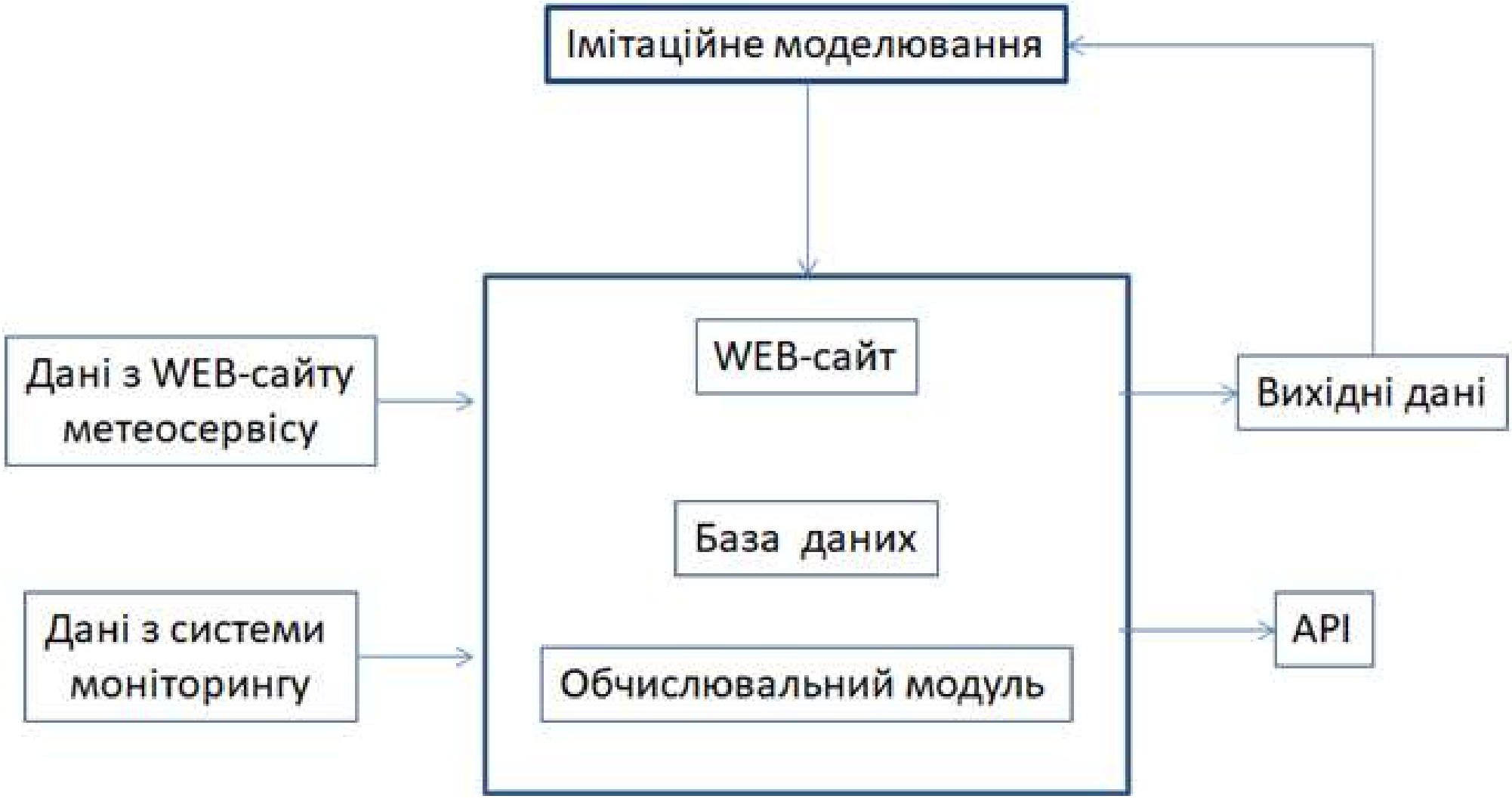
$$\text{Дефіцит потужності}_i = \sum_{i=1}^{8760} \left(E_L(i) - E_{PV}(i)\right)$$

Імовірність втрати потужності (ІВП)

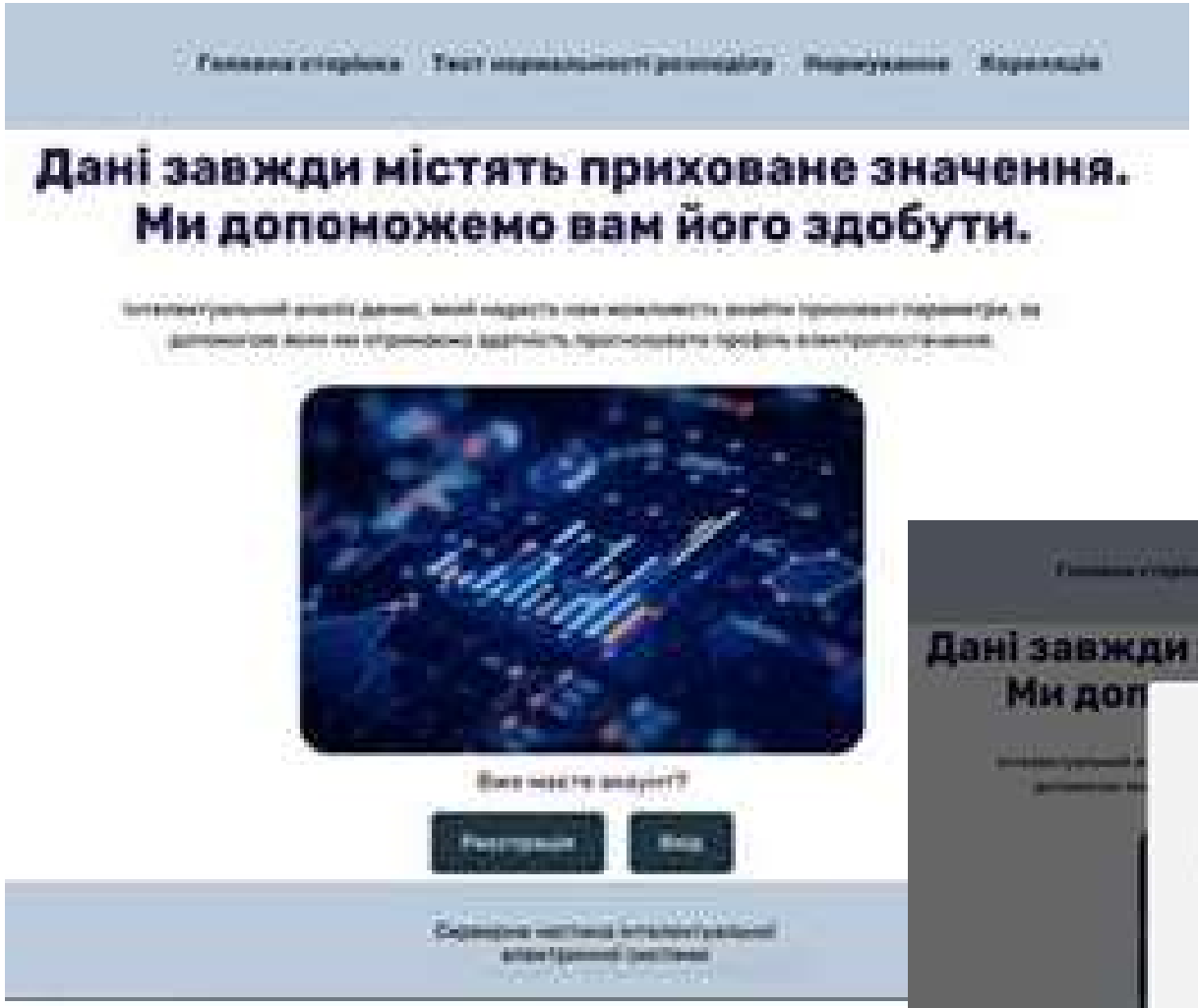
$$\text{ІВП} = \frac{\sum_{i=1}^{8760} \text{ВПЖ}(t)}{\sum_{i=1}^{8760} E(L)}$$

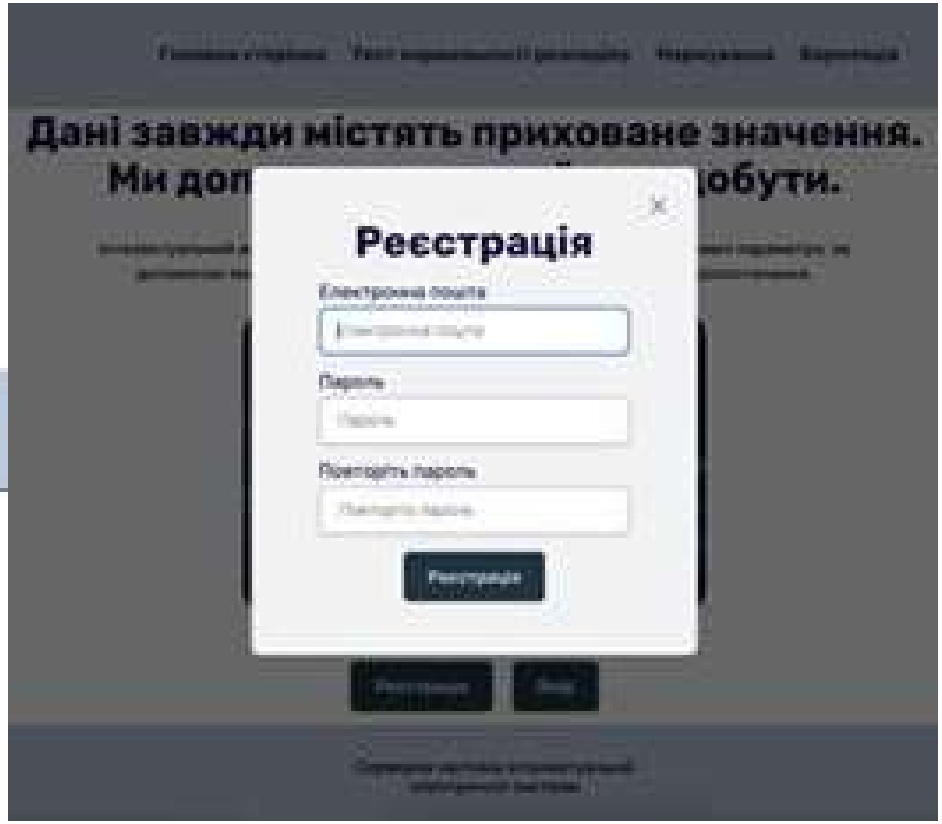
$$\text{ВПЖ}(t) = E_L(t) - [(E_{PV}(t)\eta_{bat}) + E_B(t) - E_{Bmin}] * [\eta_{inv} * \eta_{wire}]$$

Архітектура програмного забезпечення для системи електропостачання з ФЕП

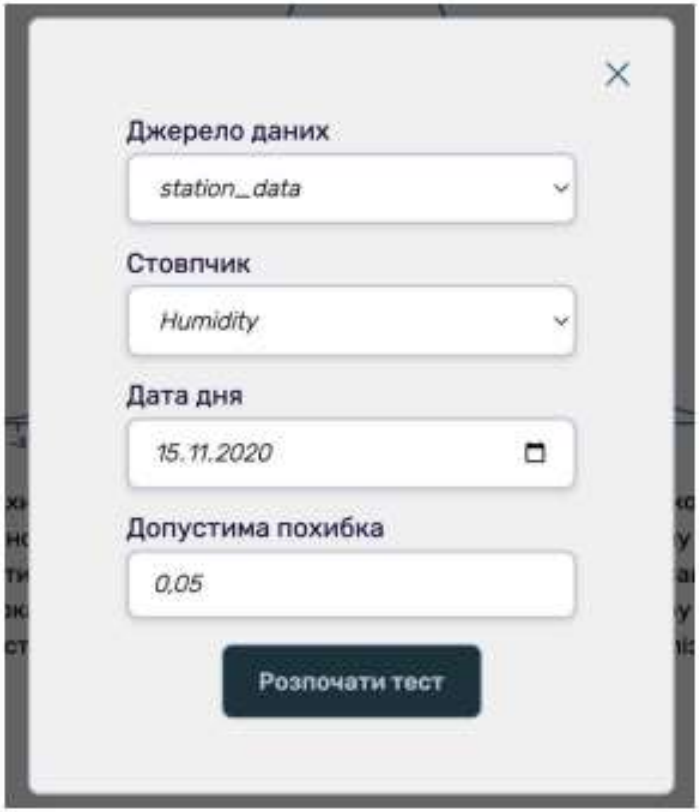


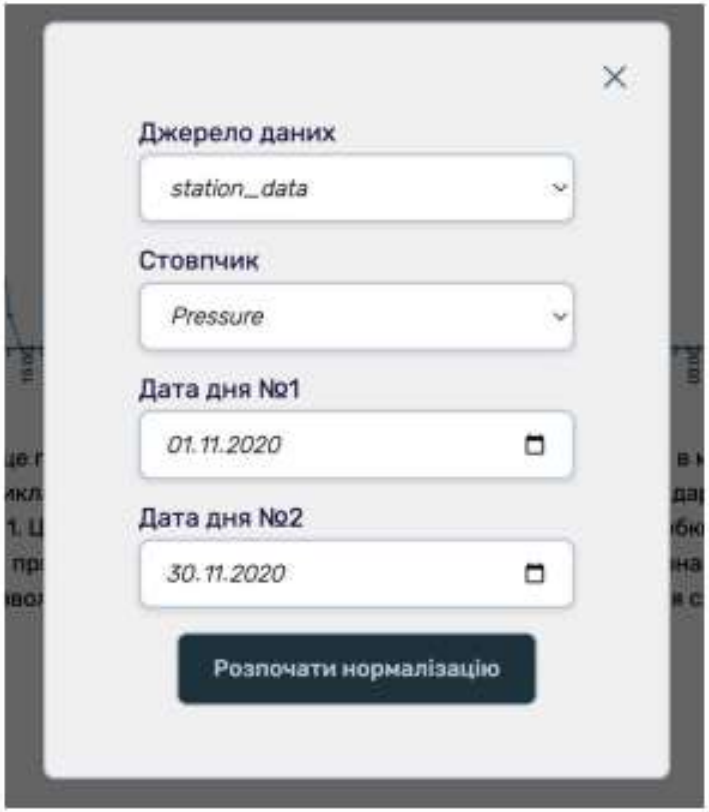
Графічний інтерфейс користувача

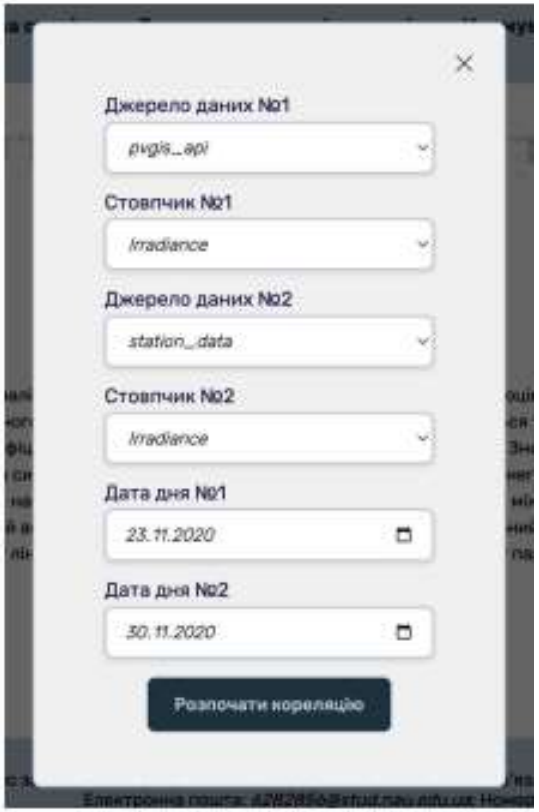




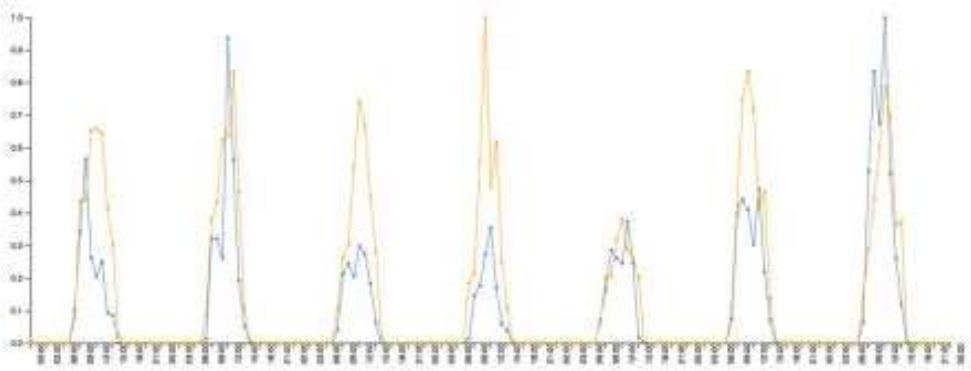
Вікна вхідних даних алгоритмів обчислень





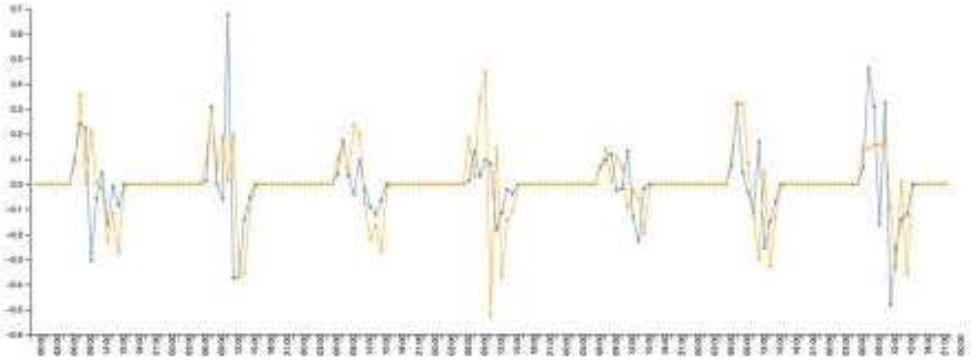


Приклад реалізації розрахунків на сервері



Порівняльний графік освітленостей двох джерел даних

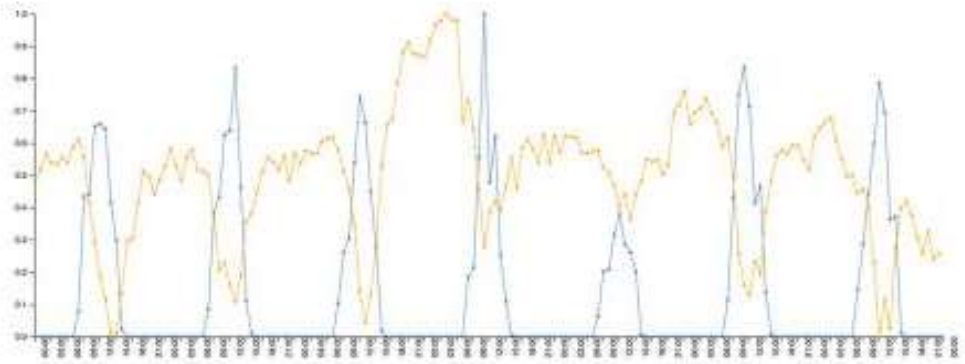
Коефіцієнт Пірсона 0.8145



Порівняльний графік зміни освітленості двох джерел даних

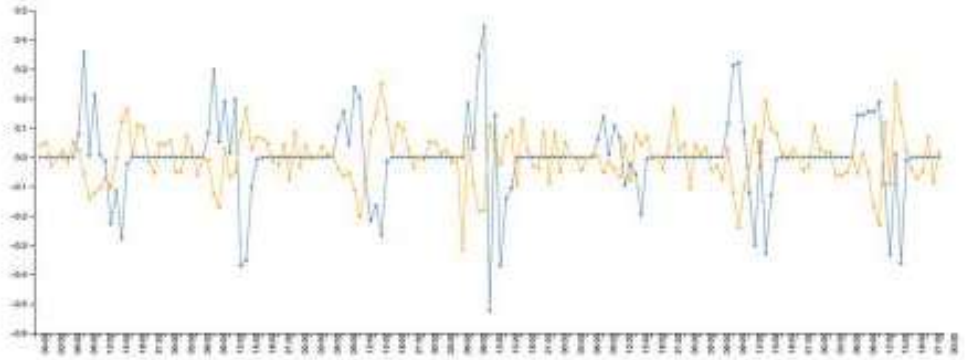
Коефіцієнт Пірсона 0.3575

Приклад реалізації розрахунків на сервері



Порівняльний графік освітленості та вологості

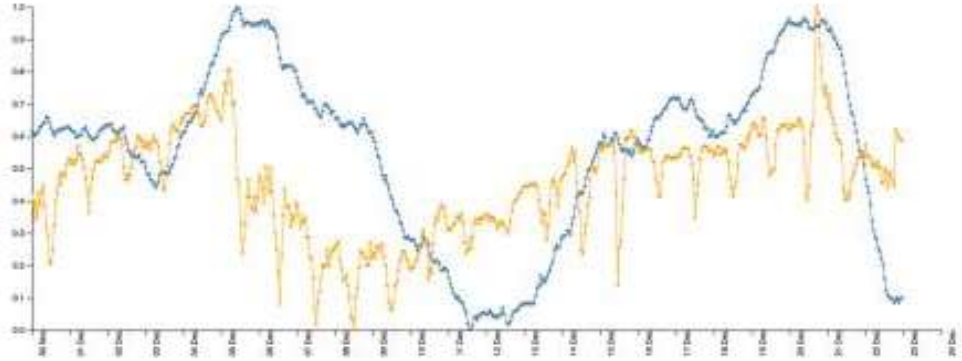
Коефіцієнт Пірсона -0.6667



Порівняльний графік зміни освітленості та вологості

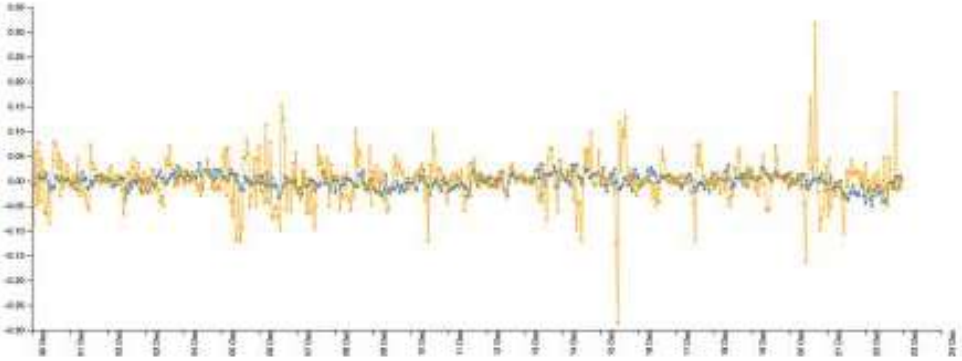
Коефіцієнт Пірсона -0.5169

Приклад реалізації розрахунків на сервері



Порівняльний графік тиску та вологості

Коефіцієнт Пірсона 0.3598

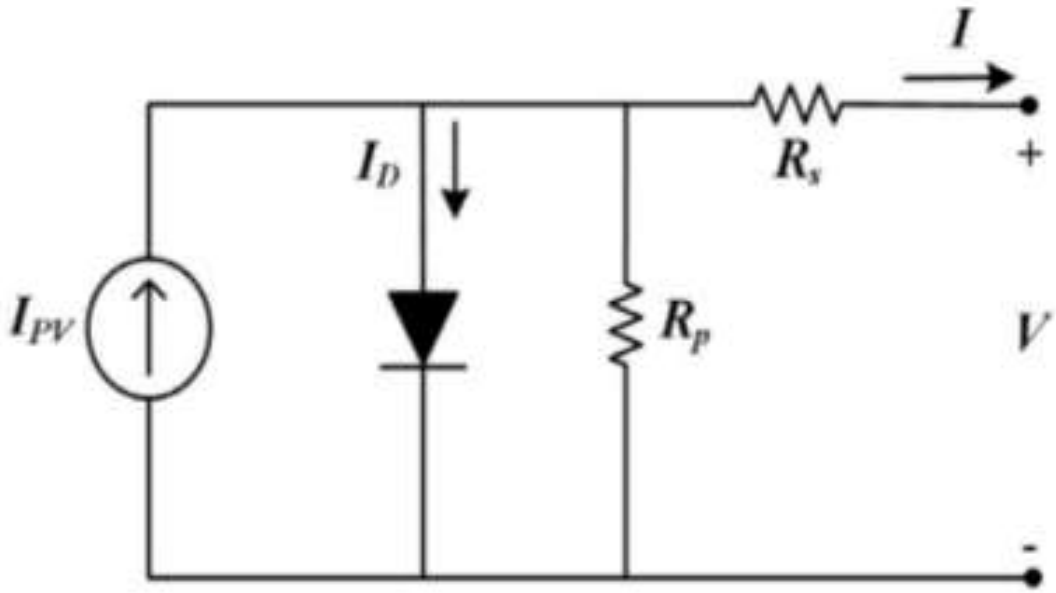


Порівняльний графік зміни тиску та вологості

Коефіцієнт Пірсона 0.0748

Модуль імітаційного моделювання

Еквівалентна електрична схема ФЕП та математична модель

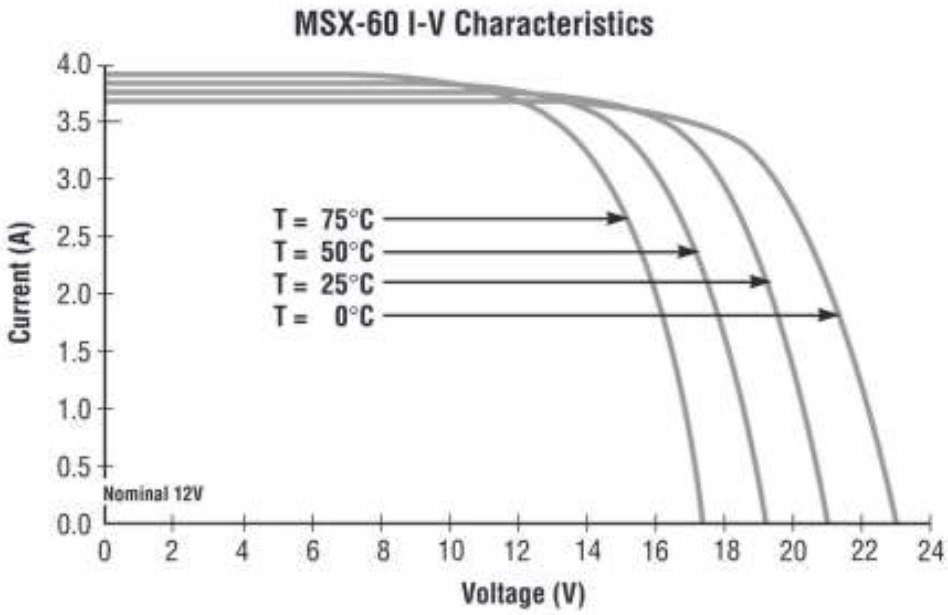


$$I = I_{PV} - I_0 \left[exp\left(\frac{q(V + IR_s)}{akT} \right) - 1 \right] - \frac{V + IR_s}{R_p}$$

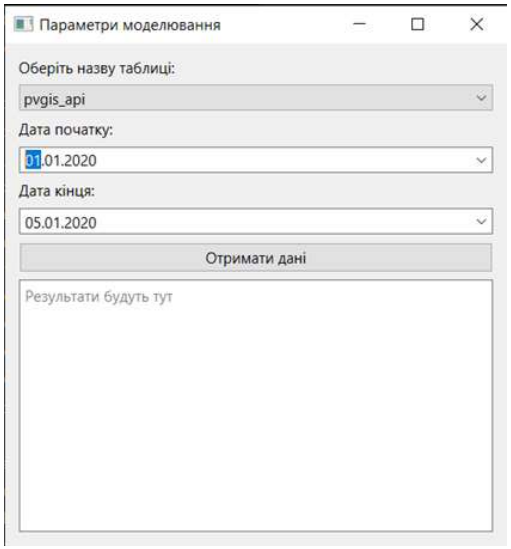
Технічні характеристики ФЕП

TYPICAL ELECTRICAL CHARACTERISTICS[1]

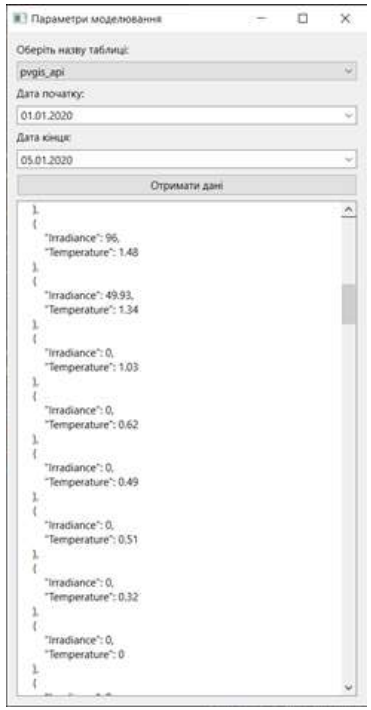
	12 VOLT CONFIGURATION[2] MSX-64	MSX-60
Typical peak power (P_p)	64W	60W
Voltage @ peak power (V_{pp})	17.5V	17.1V
Current @ peak power (1_{pp})	3.66A	3.5A
Guaranteed minimum peak power	62W	58W
Short-circuit current (1_{sc})	4.0A	3.8A
Open-circuit voltage (V_{oc})	21.3V	21.1V
Temperature coefficient of open-circuit voltage	.....–(80±10)mV/°C.....	
Temperature coefficient of short-circuit current	....(0.065±0.015)%/°C...	
Approximate effect of temperature on power	.....–(0.5±0.05)%/°C....	
NOCT[3]	..... 49°C	



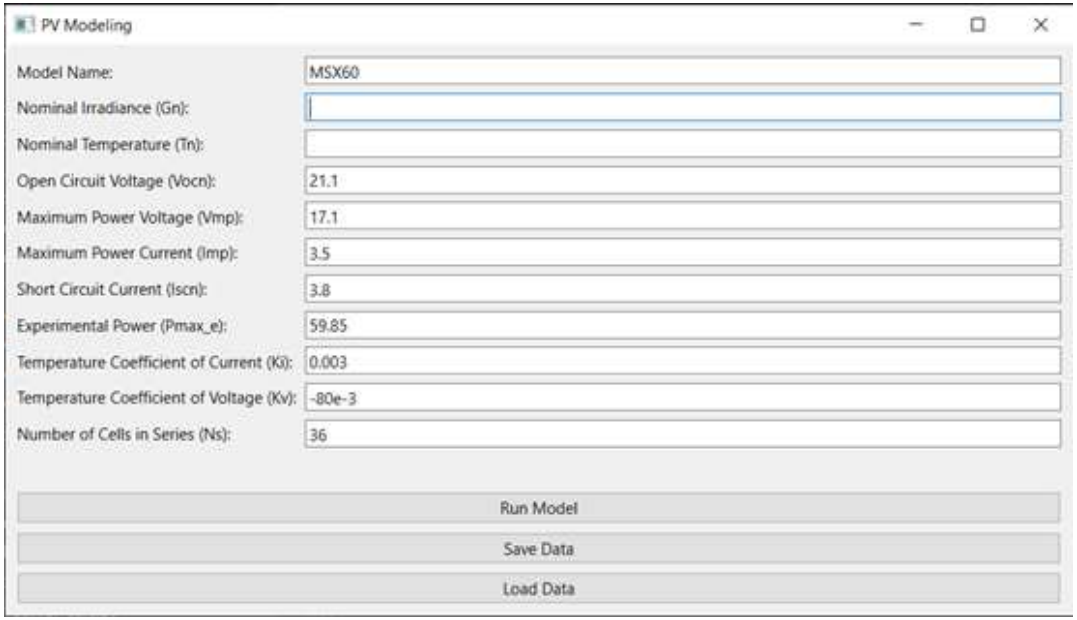
GUI модулю імітаційного моделювання



Отримання вхідних параметрів освітленості та температури

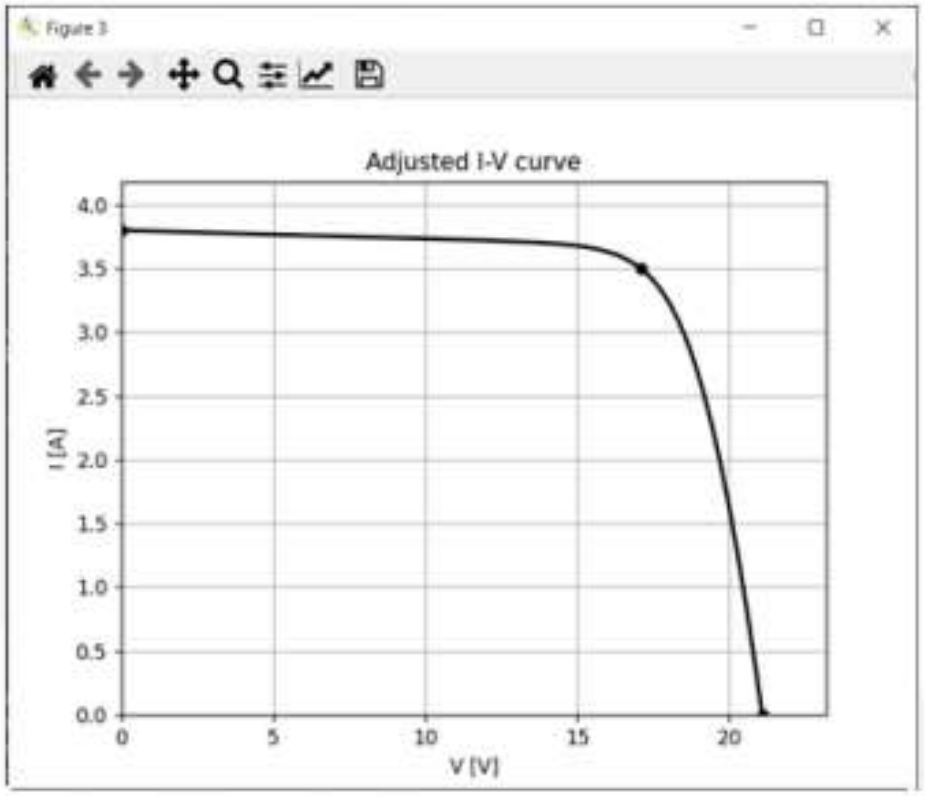


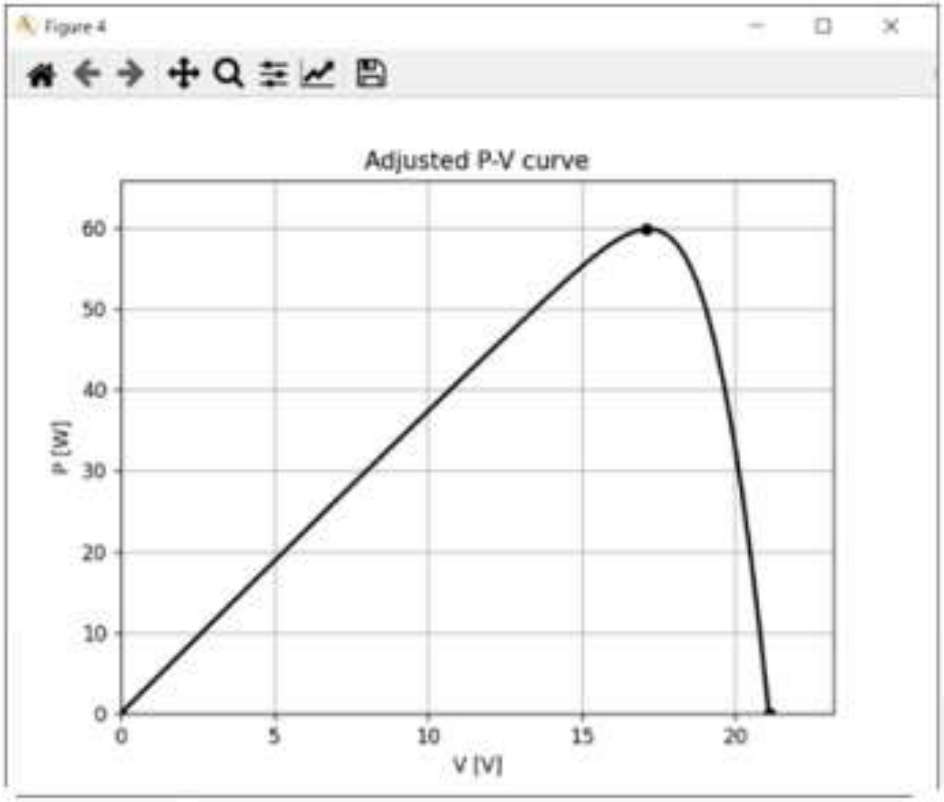
Відображення отриманих параметрів



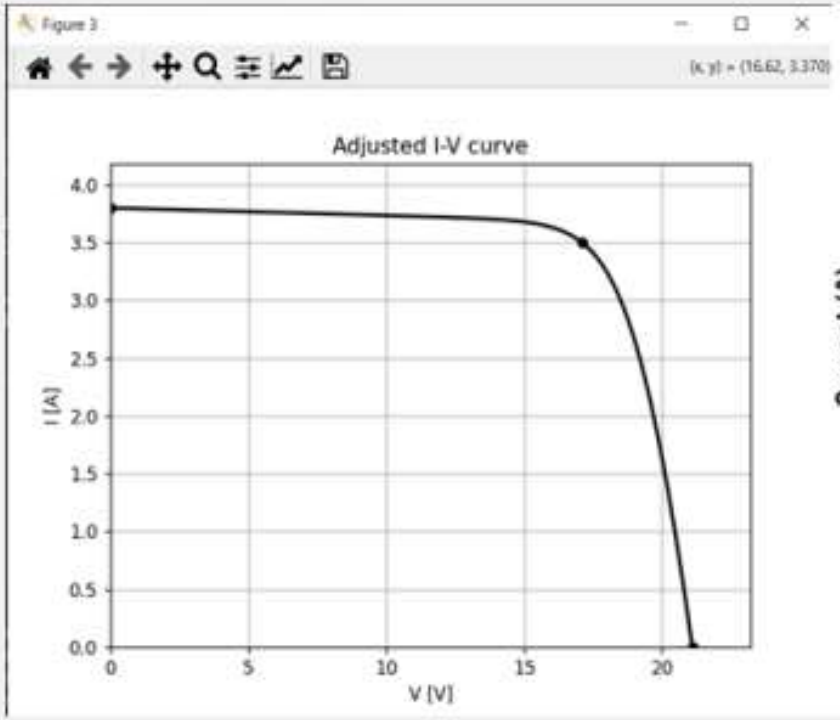
Головне вікно програми

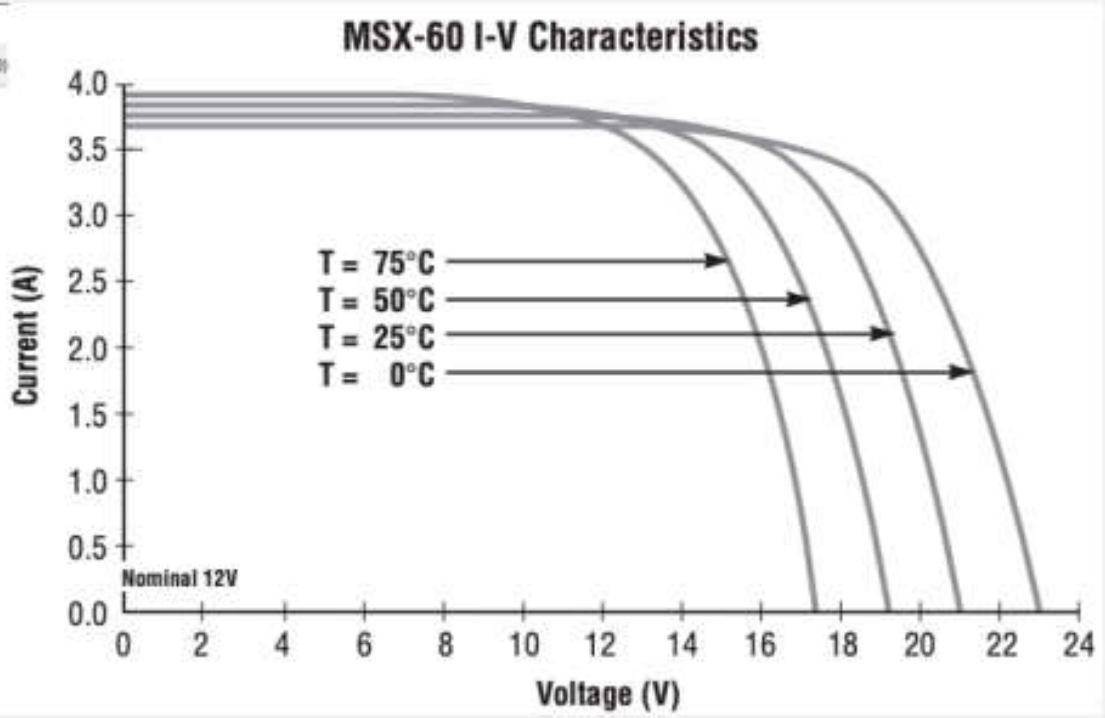
Результат імітаційного моделювання





Валідація моделі





ІМІТАЦІЙНА МОДЕЛЬ

ОГОЛОШЕНІ ХАРАКТЕРИСТИКИ ПАНЕЛІ

ВИСНОВКИ

1. Розроблення програмного забезпечення для системи електропостачання з ФЕП складається з розробки програмного засобу для імітаційного моделювання ФЕП та WEB-додатку.
2. Завдяки клієнт-серверній архітектурі, інтеграції сучасних технологій і реалізації алгоритмів аналізу, розроблений програмний засіб забезпечує високу продуктивність, масштабованість і адаптивність.
3. Розроблений програмний засіб сприятиме оптимізації енергоресурсів, підвищенню ефективності функціонування енергетичних систем і відповідає актуальним викликам цифровізації (діджиталізації) та впровадження відновлюваних джерел енергії.
4. Використання історичних даних у поєднанні з аналізом залежностей між параметрами погодних умов, які впливають на продуктивність ФЕП забезпечує точність передбачень, що дозволяє оптимізувати розподіл енергоресурсів та підвищити ефективність функціонування системи.
5. Застосування імітаційного моделювання дозволяє визначати точні характеристики ФЕП та оцінювати їх ефективність, що, в свою чергу, дозволяє прогнозувати потужність, яка буде генеруватись системою в реальних умовах експлуатації.

Це є особливо важливим для прийняття інвестиційних рішень, розробки стратегій фінансування проєктів у сфері сонячної енергетики, а також оптимального вибору конфігурації обладнання.

ДЯКУЮ ЗА УВАГУ

Kravchenko M. S., Astistova T. I.
Kyiv National University of Technologies and Design
Kravchenko O. P.
National Aviation University, Kyiv

PROCESSING DATA MODULE IN MONITORING ENVIRONMENT PARAMETERS FOR ELECTRICAL POWER GENERATION

Abstract. *The concept of "MicroGrid" is based on using automated systems carried out their own generation, monitoring and distribution of electricity taking into account the consumer requirements to achieve maximum energy consumption efficiency. The energy sector functionality is ensured by decentralized distributed electric energy resources (DER) concentrated in a certain area or local facility. Their spatial locality determines some individual weather condition differed, to some extent, from the general weather monitored by the local weather stations within the Meteorological Global Weather framework. A module for processing data coming from both the weather service website and a local environmental parameter monitoring module placed at an appropriate location within the distributed energy resources was designed and produced. The results obtained after the data processing module operation make it possible to form a corrected weather forecast in a specified DER location for achieving the optimal efficiency in the distributed electrical system.*

Keywords: *MicroGrid, distributed energy resources (DER), environmental monitoring, weather forecast, efficient operation, intelligent data analysis.*

Кравченко М. С., магістр, Астістова Т. І., доцент
Київський національний університет технологій та дизайну
Кравченко О. П., доцент
Національний авіаційний університет, м. Київ

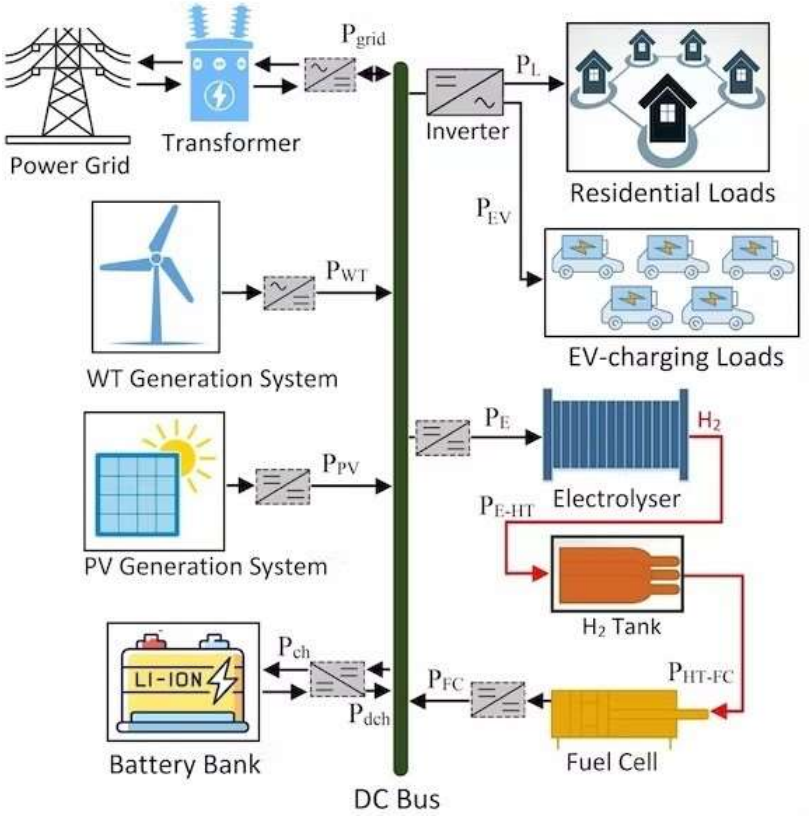
МОДУЛЬ ОБРОБКИ ДАНИХ У МОНІТОРИНГУ ПАРАМЕТРІВ СЕРЕДОВИЩА ДЛЯ ВИРОБНИЦТВА ЕЛЕКТРОЕНЕРГІЇ

Анотація. *Концепція «Інтелектуальні мережі» базується на використанні автоматизованих систем, які здійснюють власну генерацію, моніторинг та розподіл потоків електричної енергії з урахуванням вимог споживача для досягнення максимальної ефективності енергоспоживання. Функціонування енергетичного сектору забезпечується за рахунок децентралізованих розподілених джерел енергії, де джерела генерації електроенергії розосереджені на певній території або локальному об'єкті. Просторова локальність визначає індивідуальні властивості погодних умов, які в деякій мірі відрізняються від загального погодного стану, що відстежується локальними метеостанціями в рамках Метеорологічної Глобальної Погоди. Був розроблений та виготовлений модуль обробки даних, що надходять з веб-сайтів метеосервісів та локального модулю моніторингу параметрів навколишнього середовища, що розміщений у визначеній локації розподілених джерел енергії. Результати, що отримуються після обробки даних, дають можливість формувати уточнений прогноз погоди у визначеній просторовій локації з метою оптимального функціонування розподіленої електричної системи.*

Ключові слова*: локальна електрична система, розподілені джерела енергії, моніторинг параметрів оточуючого середовища, прогноз погоди, ефективне функціонування, інтелектуальний аналіз даних.*

Introduction. The concept of "SmartGrid" is based on using automated systems carried out their own generation, monitoring and distribution of electricity taking into consideration the consumer requirements to achieve maximum efficiency in energy consumption [1]. The

functionality within the energy sector is ensured by both a centralized facility and decentralized distributed energy resources (DER). These DERs scattered over an appropriate territory or local object are combined into a single microsystem (viz. MicroGrid). The system has some advantages compared to the central power supply, since there is no need to build new generating capacities, long power transmission lines and distribution networks which, in turn, require significant capital investments and additional electricity losses. The MicroGrid power supply is organized by integrating low-power energy resources to maximize their adaptation to the necessary power consumption. In general, the MicroGrid consists of several electricity resources, a battery for power accumulation and the means for regulating electricity flow (Fig. 1) [2]. At the same time, in the MicroGrid framework there is the possibility for any consumer to be quickly connected to the central power grid in the case of overload or voltage fluctuation, which results in the significant increasing the reliability of power supply.



Source: [10].

Fig. 1. **The MicroGrid Framework**

The modern development in the electrical power industry is characterized by the growing DER application used the power generated by solar panels and wind generators [3]. The productivity of such renewable energy sources depends on the weather condition in the area where these are operating. Illumination level and ambient temperature are the main environmental parameters affected the electrical energy generation by photovoltaic panels. Therefore, the electrical power produced by solar panels has been increased with the increasing in the solar radiation intensity on the one hand, and decreased with increasing in the panel temperature on the other hand [4]. The photovoltaic panel temperature has been determined by the environmental conditions where this panel is operating, which, in addition to the solar radiation mentioned above, include the ambient temperature and humidity. Similarly, the wind generator power depends on the wind speed, formed, in turn, by the atmospheric factors such as the pressure and the air humidity [5].

The MicroGrid is characterized by the fact that its generation resources are distributed over a certain territory or local object i.e. it is determined by the specified locality. Such a spatial locality has certain individual feature formed a local weather profile slightly different from the general weather profile determined by the local weather stations. The particular area with specific vegetation landscape and accessible water resource determines the environmental difference where DERs are located. In order to optimize the DER operation, the weather forecast should be performed in the appropriate location during a time. The task has been resolved within the Meteorological Global Weather framework comprised the weather probes, meteorological satellites and local weather stations. All these facilities provide the necessary information used in creating an appropriate meteorological model. Further, the probable weather parameter values occurred in the area during a time based on this model are formed [6]. But in reality with a high probability degree, the weather condition in the DER location would, to some extent, differ from the ones determined by the Meteorological Global Weather framework affected the DER efficiency. The problem resolution is, perhaps, to use a local measuring module monitored the environment in order to provide data used in correcting the predicted current weather for improving the appropriate efficiency of electrical generation resource [7].

The problem announcement. The purpose of the work is to design and produce the module for processing data coming from both the weather service website and a local portable electronic environment monitoring module placed at the appropriate DER location. The results obtained after the data processing make it possible to form a corrected weather forecast in a specified location for the MicroGrid optimal operation purpose.

Results of the research. In the Fig. 2 is shown the structure of the data processing module.

Source: designed by authors.

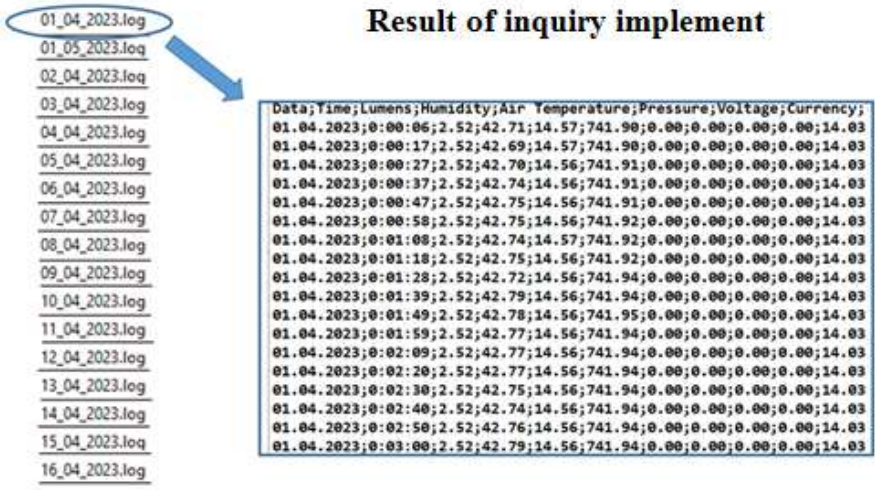
Fig. 2. **The data processing module**

Data on weather parameters are obtained from the great number of public and private professional weather stations (more than 40 thousand items) [8]. These weather stations are connected to the global weather monitoring network, the each one contributed additional data sets, and the computational models are refined on the basis of such sets to forecast more carefully the weather for the nearest time intervals [9]. The weather service (or the weather website) usually specializes in providing detailed weather reports getting access to its own API and database with the weather reports and correspondent geolocations. Weather APIs provide the access to both real-time weather data and forecasts, at the same time allowing for users also getting access to hourly and daily forecasts, as well as long-term forecasts.

Data on the weather parameters occurred in a certain area are obtained using a portable electronic module for environmental monitoring consisted of a microcontroller and sensors that collect the necessary data from the environment and transfer them to the microcontroller's memory. Communication between sensors, as well as with microcontrollers, is provided by

appropriate radio modules. Access to the data stored in the microcontroller's memory, their exchange and general storage is provided by web-server connected to a cloud data storage service.

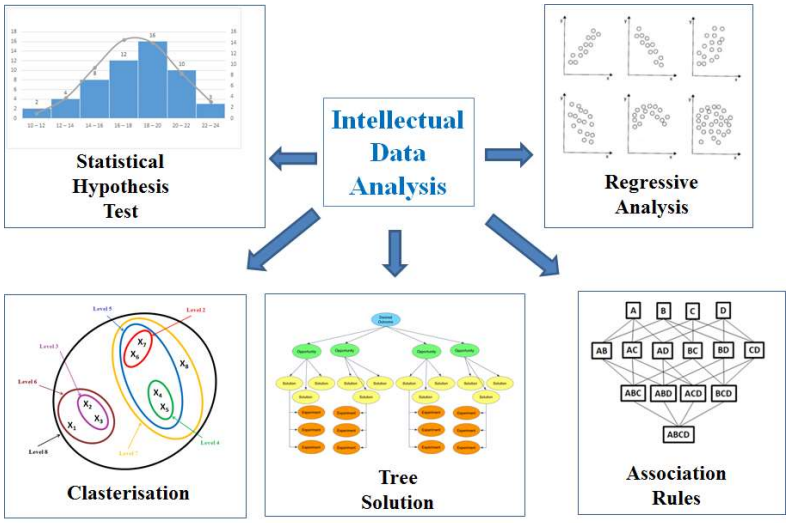
In the Fig. 3 is shown the data transmitted from the local environment monitoring module to web-server after inquiring. Module's microcontroller forms a virtual COM-port thereby the necessary data from sensors are transmitted.



Source: designed by authors.

Fig. 3. **Result of inquiry implementation**

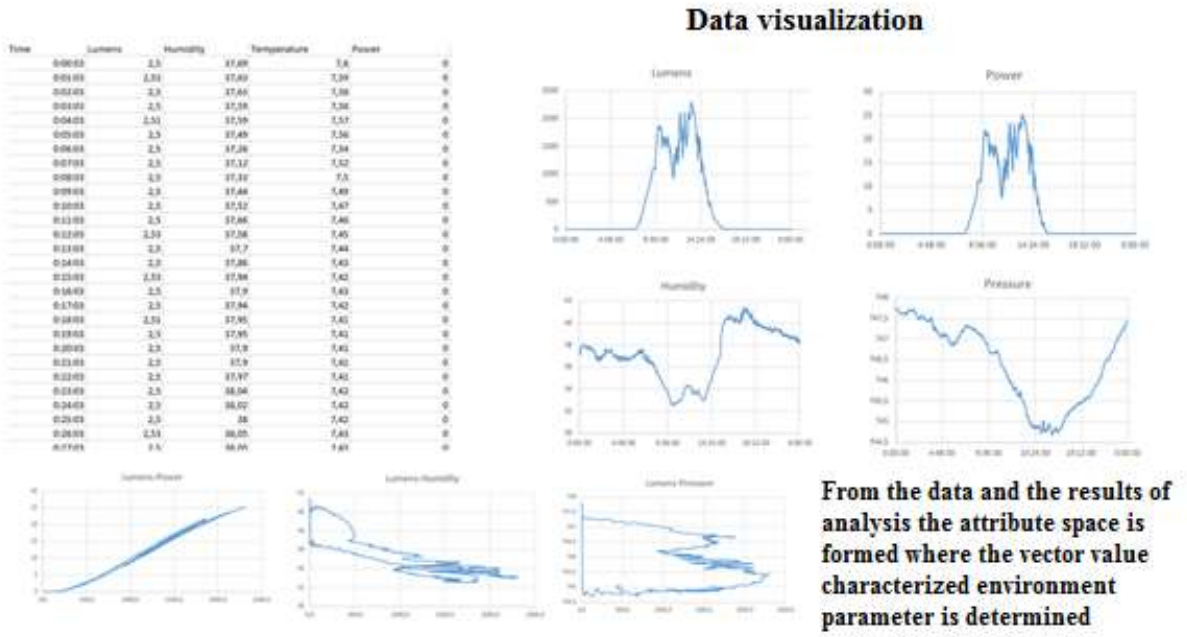
The information processing module performs preliminary processing to the received data that includes determining the value set of all available features in each data point from the data massive. Each feature expresses a certain value of the feature space, which places the data point in a certain location of such space. The dynamic weather evolution occurred during a time is described with an attribute set. Based on the feature space vector value, the data intellectual analysis is carried out with subsequent visualization for the obtained results (Fig. 4). The main techniques used in the processing module are as follows: 1) At the beginning Statistical Hypothesis Test and Regressive Analysis is implemented to ensure the correctness into input data and assign the appropriate attributes; 2) On the base of attribute value the feature space is formed using Clasterisation, Tree Solution and Assotiation Rules Formation methods.



Source: designed by authors.

Fig. 4. **The data intellectual analysis**

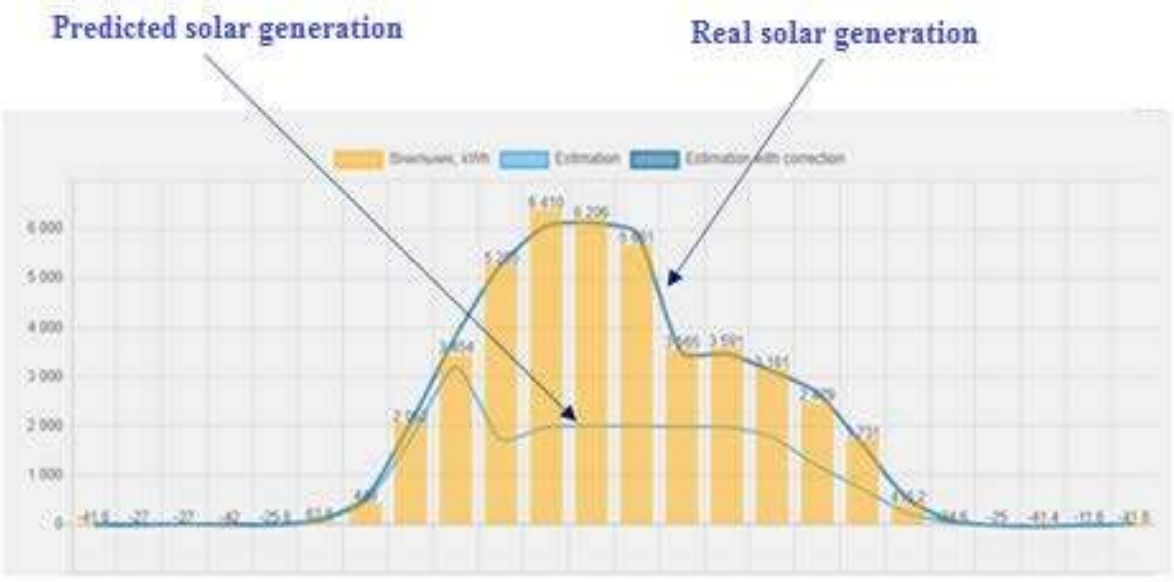
In the Fig. 5 is shown the initial steps in forming the feature space along with finding an attributive value.



Source: designed by authors.

Fig. 5. **Finding an attributive value to form the feature space from data**

In fact, predicted solar generation from solar panels installed in determined location does not often coincide with real solar generation because the predicted generation has been evaluated from meteorological conditions for necessary location based on the general weather within the Meteorological Global Weather framework (Fig. 6). So, the presented fact confirms the purpose for using additional complementary facilities to correct the current weather forecast for appropriate space location where the generation resources are distributed.



Source: [11].

Fig. 6. **The predicted and real solar generation in determined location**

Conclusion. The MicroGrid is characterized by the fact that electricity generation resources are located in a distributed area of DER facilities. DER's spatial locality determines the appearance some different weather conditions compared to the general weather condition, implemented by the local weather stations within the Meteorological Global Weather framework. A module for processing data coming from both the weather service website and a portable electronic local environmental monitoring module, located at an appropriate location, was designed and produced. The results obtained after the data processing make it possible to form the corrected weather forecast in a specified location for the MicroGrid optimal operation.

References

1. Grid 2030: A National Version for Electricity's Second 100 Years. *Office of Electric Transmission and Distributon Unnited State Departament of Energy*. URL: https://www.energy.gov/oe/articles/grid-2030-national-vision-electricitys-second-100-years.
2. Uddin, M., Mo, H., Dong, D., Elsawah, S., Zhu, J., Guerrero, J. M. (2023). Microgrids: A review, outstanding issues and future trends. *Energy Strategy Reviews*, Vol. 49, P. 101–127.
3. Babaremu, K., Olumba, N., Chris-Okoro, I., Chuckwuma, K., Jen, T.-C., Oladijo, O., Akinlabi, E. (2022). Overview of Solar–Wind Hybrid Products: Prominent Challenges and Possible Solutions. *Energies*, Vol. 15, P. 6014.
4. Bagalini, V., Zhao, B. Y., Wang, R. Z., Desideri, U. (2019). Solar PV-Battery-Electric Grid-Based Energy System for Residential Applications: System Configuration and Viability. *Research*, Vol. 2019.
5. Rao, K. R. (2019). Wind Energy for Power Generation. Springer Nature: Switzerland AG. 1443 p.
6. World Weather Information Service. URL: https://worldweather.wmo.int/en/home.html.
7. Кравченко О. П., Манойлов Е. Г., Арзікулов Т. С. Моніторинг та управління параметрами в Смарт-системах електропостачання. *Інноватика в освіті, науці та бізнесі: виклики та можливості: матеріали I Всеукраїнської конференції здобувачів вищої освіти і молодих учених*. Київ: КНУТД, 2020. С. 262–266.
8. World Meteorological Organization. URL: https://public.wmo.int/en.
9. Holton, J. R., Hakim, G. J. (2013). An Introduction to Dynamic Meteorology. Elsevier: Academic Press. 532 p.
10. World Economic Forum Here's how microgrids could keep the power on during extreme weather events, Feb 15, 2023. URL: https://www.weforum.org/agenda/2023/02/microgrids-energy-electricity-weather/
11. Міжнародна група компаній народжена в Україні, яка розробляє технології та втілює проекти відновлюваної і традиційної енергетики. URL: https://kness.energy/

UDC 004.75

THE USE OF RS-485 INTERFACE FOR CONNECTING SENSORS IN AGRICULTURAL APPLICATION

M.S. Kravchenko, graduate student
Kyiv National University of Technologies and Design
T.I. Astistova, PhD, Associate Professor
Kyiv National University of Technologies and Design

Keywords: greenhouse complex, climate control system, environmental parameters monitoring, RS-485 interface.

In agriculture, greenhouse complexes have been becoming increasingly popular, due to providing an opportunity to supply vegetables and fruits to consumers in cities every day, as well as silage grass to livestock farms throughout all the year, both in summer and winter. In greenhouses, plants need to be provided with optimal range of climatic conditions in order to ensure the maximum product yield. The high agricultural crops yield of is provided by: 1) ambient temperature higher than 14 ^{0}C and irradiance during at least 8 hours a day [1]. Optimal conditions for plant growth are maintained by a climate control system, which includes the environmental parameters monitoring unit and appropriate actuators controlling necessary heating, an appropriate humidity level, both in the air and in the soil. It is known that modern greenhouse agricultural complexes can be localized in the area up to 30 hectares [2], which causes corresponding difficulties in using the climate control system, since the signal power decreases with increasing the cable length to provide the signal transmission in system.

To solve the problems associated with the optimal climate control system operation, the RS-485 interface was used, which is characterized by the possibility for two-way data exchange between several units using a single two-wire communication line maintained with twisted wire pair in half-duplex mode. The hardware implementation of the interface is the receiver chips with differential inputs / outputs (to the line) and digital ports (to the UART controller ports) [3] (Fig.1).

In the configuration studied, temperature and humidity sensors were installed both inside and outside the greenhouse. Inside the greenhouse, the sensors were placed one pair at ground level, the second pair at the level of the greenhouse ceiling and the third pair of sensors measured the moisture and temperature in the soil where the crops were being grown. In addition, irradiance sensors were installed at the ceiling and ground level in the greenhouse, and a light sensor was also installed outside the greenhouse. Temperature, humidity and irradiance level are monitored online with data being transmitted to a server where the information is stored and properly processed.

The temperature of the outside air significantly affects the yield of crops in the greenhouse due to the large heat losses that occur on virtue of the heat exchange between the greenhouse external and internal space the through its walls. Large sensor number appeared in the case of great agricultural complex

where large number of greenhouses were located has been found not to be connected to a network with a simple architecture. Such a consideration, in turn, has limit the choice in technical solutions to advanced interfaces. Therefore, for the case described, it is the RS-485 interface that is the most suitable, because it allows to connect a sufficient sensors number within a radius of 1200 meters around the main controller (server).

The use of RS485 interface allows in creating a reliable system resistant to external interference and malfunctions, as well as with a minimum output electromagnetic background due to differential signal transmission. Due to the "twisted pair", RS485 provides high stability to external electromagnetic interference, which can be critical in environment where many sources of noise are presented, and makes the system invisible from the outside. This ensures continuous monitoring and protection, minimizing the risk of unauthorized access. The RS485 interface allows data to be transmitted over a distance up to 1200 meters, which is ideal for greenhouses with large area. Also, the system provided with the RS485 interface can easily adapt to changing conditions. The ability to be connected up to 32 devices in one bus makes the system flexible and expandable one. This interface ensures the optimal integration of different units into one holistic system, maintaining automatic control and efficient monitoring.

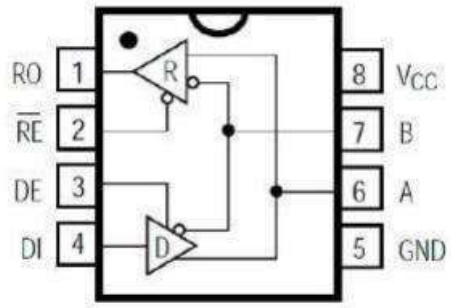


Figure 1 – Hardware implementation of the RS-485 interface

D - driver; R – receiver, DI - driver input, RO - receiver output, DE - driver enable, RE - receiver enable), A - direct differential input/output, B - inverse differential input/output.

References

1. Vaičiulytė V. Effects of meteorological conditions and plant growth stage on the accumulation of carvacrol and its precursors in Thymus pulegioides / Vaičiulytė V., Butkienė R., Ložienė K.// Phytochemistry. – 2016. – V.128. – P. 20-26.

2. Greenhouse farms in Ukraine. https://agri-gator.com.ua/category/teplichni-gospodarstva/ (current information at October 27, 2024).

3. Protocols, interfaces, technologies. Description of RS-485. https://vkmodule.com.ua/Description/Description2_ua.html