

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

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО

В.о. завідувача кафедри
Олександр КОВАЛЬ

«__» _____ 2026 р.

Дипломна робота

**на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»
спеціальності 121 Інженерія програмного забезпечення
на тему: «Програмний модуль автоматизації режимів зарядки
акумуляторних батарей лабораторного стенду з сонячною генерацією
енергії»**

Виконав:

студент IV курсу, групи ТВ-22

Полупан Михайло Валерійович

(прізвище, ім'я, по батькові)

(підпис)

Керівник:

доктор технічних наук, професор,

професор кафедри ІПЗЕ Федорова Н. В.

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

(підпис)

Рецензент:

доцент, к.т.н. кафедри ТАЕ, Сірий О.А.

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

(підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.

Студент

(підпис)

Київ— 2026

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

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри
Олександр КОВАЛЬ

« » 2026р.

ЗАВДАННЯ

на дипломну роботу студенту

Полупану Михайлу Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного модуля автоматизації режимів зарядки
аккумуляторних батарей лабораторного стенду з сонячною генерацією енергії.
керівник роботи доктор технічних наук, професор кафедри ІПЗЕ Федорова Н. В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» травня 2026 р. № 1992-с

2. Строк подання студентом роботи «11» червня 2026 р.

3. Вихідні дані до роботи: Python, фреймворк FastAPI, бібліотеки NumPy, pandas,
pvlb, matplotlib, SQLAlchemy, Pydantic, PyYAML, база даних SQLite, frontend-
бібліотека React, інструменти PyCharm, DataGrip, Docker, Docker Compose.

4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно
розробити): аналіз існуючих рішень; вибір інструментів розробки; розробка
алгоритму автоматичного заряджання батарей на основі наявної інформації про
стан системи; тестування системи.

5. Перелік ілюстративного матеріалу: діаграма прецедентів; блок-схеми
алгоритмів; графіки тестування симуляційних даних; таблиці параметрів моделей;
скріншоти веб-інтерфейсу та результатів роботи програмної системи.

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	31.10.2025	
2	Дослідження предметної області	01.11.2025– 31.11.2025	
3	Дослідження існуючих рішень	01.12.2025– 31.12.2025	
4	Постановка вимог до проектування системи	01.01.2026– 31.01.2026	
5	Розробка програмного продукту	01.02.2026– 03.05.2026	
6	Тестування	04.05.2026– 10.05.2026	
7	Захист програмного продукту	11.05.2026– 15.05.2026	
8	Оформлення дипломної роботи	18.05.2026– 31.05.2026	
9	Передзахист	01.06.2026– 05.06.2026	
10	Захист	15.06.2026– 19.06.2026	

Студент

(підпис)

Михайло ПОЛУПАН

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Наталія ФЕДОРОВА

(ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 91 сторінку, 16 рисунків, 10 таблиць, 2 додатки та 25 посилань.

Метою роботи є розробка програмного модуля автоматизації режимів заряджання акумуляторних батарей лабораторного стенду з сонячною генерацією енергії.

Для досягнення поставленої мети було розроблено програмну систему, яка моделює роботу лабораторного стенду SmartEnergy Lab. Система містить модулі симуляції сонячної генерації, доступності централізованої електромережі, стану акумуляторної батареї та споживання електроенергії. На основі даних від цих модулів EMS формує рішення щодо режиму роботи системи, розподілу енергії між сонячною генерацією, електромережею, батареєю та навантаженням, а також переходу до захисних режимів за умов відключення, критичного заряду батареї або перевищення можливостей інвертора. Результати симуляції зберігаються в базі даних і відображаються у веб-інтерфейсі.

Практичне значення одержаних результатів полягає у можливості демонстрації та перевірки алгоритмів автоматизації режимів заряджання акумуляторних батарей у системі з сонячною генерацією, змінним споживанням та нестабільною електромережею. Розроблена система може використовуватися як навчальний і демонстраційний прототип для подальшої інтеграції з реальними датчиками та апаратними компонентами лабораторного стенду.

Ключові слова: програмна система, база даних, API, веб-інтерфейс, сонячна генерація, акумуляторна батарея, EMS, автоматизація заряджання, симуляція, електромережа

ABSTRACT

Structure and scope of the thesis. The thesis comprises 91 pages, 16 figures, 10 tables, 2 appendices and 25 references.

The aim of this thesis is to develop a software module for automating the battery charging modes of a laboratory test bench with solar power generation.

To achieve this objective, a software system was developed that simulates the operation of the SmartEnergy Lab test bench. The system comprises modules for simulating solar power generation, the availability of the centralised electricity grid, the state of the battery, and electricity consumption. Based on data from these modules, the EMS makes decisions regarding the system's operating mode, the distribution of energy between solar generation, the grid, the battery and the load, as well as the transition to protective modes in the event of a power cut, critical battery charge or exceeding the inverter's capacity. The simulation results are stored in a database and displayed in a web interface.

The practical significance of the results lies in the ability to demonstrate and verify algorithms for automating battery charging modes in a system with solar generation, variable consumption and an unstable grid. The developed system can be used as a training and demonstration prototype for subsequent integration with real sensors and hardware components of a laboratory test bench.

Keywords: software system, database, API, web interface, solar power generation, battery, EMS, charging automation, simulation, power grid

ЗМІСТ

Перелік умовних позначень, скорочень і термінів	8
ВСТУП.....	9
1 АКТУАЛЬНІСТЬ, МЕТА ТА ПОСТАНОВКА ЗАВДАННЯ	10
1.1 Актуальність теми	10
1.2 Мета та завдання роботи.....	11
1.3 Постановка завдання до розробки програмного забезпечення	12
Висновки до розділу 1.....	13
2 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ДО АВТОМАТИЗАЦІЇ РЕЖИМІВ ЗАРЯДКИ ТА СТВОРЕННЯ СПРОЩЕНОГО ЦИФРОВОГО ДВІЙНИКА.....	14
2.1 Підходи до симуляції сонячної генерації	15
2.2 Підходи до моделювання акумуляторних батарей	16
2.3 Підходи до автоматизації режимів зарядки, EMS та BMS	17
2.4 Підходи до моделювання споживання електроенергії	19
2.5 Моделювання нестабільної електромережі та аварійних режимів	20
Висновки до розділу 2.....	21
3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ІНСТРУМЕНТІВ РОЗРОБКИ .	23
3.1 Вибір серверної частини.....	24
3.2 Засоби роботи з даними та конфігурацією	24
3.3 Бібліотеки для симуляції та обробки даних	25
3.4 Вибір frontend-інструментів	25
3.5 Контейнеризація проєкту	26
Висновки до розділу 3.....	26
4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ	28
4.1 Програмна реалізація симуляції сонячної генерації.....	28
4.1.1 Конфігураційні параметри сонячної станції.....	29
4.1.2 Розрахунок ідеальної сонячної генерації	31
4.1.4 Отримання погодних даних і формування погодного коефіцієнта.....	32
4.1.5 Формування історичної та прогнозовної генерації.....	33
4.1.6 Допоміжна оцінка напруги та струму сонячного масиву	35
4.1.7 тестування роботи модулі.....	36

4.2 Програмна реалізація симуляції наявності електромережі за умов обстрілів	39
4.2.1 Вихідні припущення та параметри моделі	40
4.2.2 Формування добового сценарію обстрілу	42
4.2.3 Моделювання пошкоджень енергетичної інфраструктури.....	46
4.2.4 Перехід від пошкоджень до точок доступності електромережі	50
4.2.5 Тестування роботи модуля	54
4.3.1 Конфігурація батареї та параметри моделі.....	56
4.3.2 Алгоритм оновлення стану батареї	57
4.3.3 Облік SoH, напруги та статусу батареї	59
4.4 Програмна реалізація симуляції споживання електроенергії.....	60
4.4.1 Вхідні припущення та контекст симуляції	60
4.4.2 Каталог пристроїв та груп навантаження	61
4.4.3 Формування сценаріїв споживання та реакція на стан системи.....	64
4.5 Програмна реалізація модуля автоматизації режимів заряджання	70
4.5.1 Вхідні дані, конфігурація та режими EMS	71
4.5.2 Розрахунок ризику та вибір автоматичного режиму	74
4.5.3 Формування рішення про розподіл енергії.....	76
4.5.4 Робота EMS під час відключення та захисні режими.....	77
4.5.5 Передача рішення EMS до батареї та збереження результатів	79
Висновки до розділу 4.....	80
5 Опис графічного інтерфейсу програмної системи	82
5.1 Призначення графічного інтерфейсу	82
5.2 Загальний вигляд інтерфейсу	84
5.3 Перегляд додаткових модулів системи.....	85
Висновки до розділу 5.....	86
ВИСНОВКИ.....	87
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	89
ДОДАТОК А Лістинг розробленого модуля.....	92
ДОДАТОК Б Презентація.....	115

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	Програмний інтерфейс для обміну даними між frontend та backend частинами системи.
Backend	Серверна частина програмної системи з симуляційними модулями, API та базою даних.
BMS	Battery Management System, система керування акумуляторною батареєю.
DoD	Depth of Discharge, глибина розряду акумуляторної батареї.
EMS	Energy Management System, система керування енергетичними потоками.
FastAPI	Python-фреймворк, використаний для реалізації серверного API.
Frontend	Клієнтська частина програмної системи з графічним веб-інтерфейсом.
MPPT	Maximum Power Point Tracking, пошук точки максимальної потужності сонячної панелі.
PV	Photovoltaic, фотоелектрична система сонячної генерації.
React	JavaScript-бібліотека для створення графічного веб-інтерфейсу.
Rule-based EMS	EMS, побудована на заздалегідь заданих правилах і порогових значеннях.
SoC	State of Charge, поточний рівень заряду акумуляторної батареї.
SoH	State of Health, показник технічного стану акумуляторної батареї.
SQLite	Локальна файлова база даних для збереження симуляційних та історичних даних.
Uvicorn	ASGI-сервер, використаний для запуску FastAPI-застосунку.
YAML	Формат конфігураційних файлів із параметрами лабораторного стенду.
Інвертор	Пристрій для перетворення постійної напруги батареї у змінну напругу навантаження.

ВСТУП

Повномасштабне вторгнення росії в Україну призвело до змін у загальному рівні обізнаності населення в сфері енергонезалежності. Якщо для європейців сонячні панелі та альтернативні джерела енергії зазвичай асоціюються з «зеленою» енергією, то в українських умовах це стало питанням диверсифікації джерел постачання, локальної енергетичної незалежності та стійкості всієї електромережі. З початком масованих обстрілів енергетики питання особистої енергонезалежності стало не абстрактним, а цілком практичним: чи зможе кожна окрема людина та підприємство забезпечити нормальну життєдіяльність та працездатність за умов регулярних відключень енергопостачання.

Це стало причиною для встановлення в межах SmartEnergy Lab на кафедрі ІПЗЕ міні-сонячної електростанції, яка дозволяє покривати частину споживання завдяки сонячній генерації та забезпечує безперебійну роботу критичного обладнання за умов відсутності централізованого енергопостачання завдяки накопиченню енергії в акумуляторі.

Хоча встановлене апаратне забезпечення й не передбачає модифікацію встановленого програмного забезпечення чи отримання даних через API воно стало важливою відправною точкою для розуміння як функціонують подібні системи. Завдяки стенду з сонячними панелями вдалось наочно побачити як себе поводить система, які має недоліки та переваги та яким чином її робота могла би бути покращена в межах теми дипломної роботи.

Завданням дипломної роботи стало створення програмного забезпечення для автоматизації режимів зарядки, який, на основі даних про сонячну генерацію, кількість відключень, споживання та стан акумулятора вибирає оптимальну стратегію розподілу енергії яка забезпечує баланс між економією грошей та безперебійністю електропостачання.

1 АКТУАЛЬНІСТЬ, МЕТА ТА ПОСТАНОВКА ЗАВДАННЯ

Розділ присвячено обґрунтуванню актуальності теми, формулюванню мети дипломної роботи та визначенню основних завдань розробки програмного забезпечення. У межах розділу розглянуто, чому автоматизація режимів заряджання акумуляторних батарей є важливою для систем із сонячною генерацією, які проблеми виникають під час роботи таких систем в умовах нестабільного електропостачання та які функції має виконувати розроблювана програмна система.

1.1 Актуальність теми

Враховуючи контекст, в умовах якого виконується дана робота, а саме триваюче повномасштабне вторгнення, регулярні обстріли енергетики та загальносвітова нестабільність актуальність цієї теми існує як у короткостроковій, так і в довгостроковій перспективі не тільки в Україні, але й у світі загалом. В Україні актуальність системи пояснюється поточними руйнуваннями енергетичної інфраструктури та, як наслідок, значним попитом на системи резервування і генерації електроенергії та оптимізації їх роботи як у малих, так і в середньо-великих масштабах. Для решти світу це питання поки що більше пов'язане з довгостроковим переходом до нейтральних викидів CO₂ та другорядним, але також важливим, ефектом у вигляді більш стійкої та диверсифікованої енергетичної системи.

Оптимізація роботи таких систем у короткостроковій перспективі підвищує ефективність їх використання та збільшує час стабільної роботи до відмови. Як наслідок, зменшуються витрати на обслуговування, підвищується кількість виробленої енергії на одиницю площі, що зайнята засобами генерації, а також зменшується обсяг ресурсів, необхідних для вироблення однієї кіловат-години електроенергії. Це, у свою чергу, впливає і на сумарний рівень викидів, пов'язаних

з одиницею згенерованої енергії що є важливим в контексті глобального потепління.

Враховуючи все перелічене, оптимізація етапів функціонування альтернативних джерел електроенергії та засобів накопичення енергії є нагальною проблемою і водночас довгостроковою метою розвитку енергетичних систем. Актуальність автоматизації вибору режимів зарядки впливає з необхідності оптимізації процесу розподілу енергії без постійного втручання людини.

1.2 Мета та завдання роботи

Метою роботи є розроблення програмного забезпечення для моделювання роботи лабораторного стенду SmartEnergy Lab та автоматизації режимів заряджання акумуляторних батарей.

Розроблене програмне забезпечення має забезпечувати:

- симуляцію основних процесів у системі, зокрема сонячної генерації, споживання електроенергії, стану акумуляторної батареї та доступності централізованої мережі електропостачання;
- зберігання історичних даних, отриманих від змодельованих підсистем;
- відображення поточного та історичного стану системи у веб-інтерфейсі;
- підтримку автоматичного й ручного вибору режимів роботи системи;
- розподіл доступної енергії між акумуляторною батареєю та навантаженням з урахуванням вибраного режиму роботи, історичних даних і поточного стану системи.

1.3 Постановка завдання до розробки програмного забезпечення

Завданням роботи є розроблення програмної системи яка моделює роботу лабораторного стенду SmartEnergy Lab та основного модуля автоматизації режимів заряджання акумуляторних батарей. Система має відтворювати основні процеси, що відбуваються у стенді з сонячною генерацією, зокрема генерацію електроенергії сонячними панелями, зміну стану акумуляторної батареї, споживання енергії приладами, що використовують працівники та студенти через інвертор та доступність централізованої мережі електропостачання.

Оскільки наявне апаратне забезпечення лабораторного стенду не забезпечує повного зовнішнього програмного керування роботою фізичної системи, розроблене програмне забезпечення орієнтоване на відтворення роботи стенду, перевірку алгоритмів автоматизації та демонстрацію можливих режимів автоматизації заряджання акумуляторів.

Для досягнення поставленої мети необхідно розробити такі складові програмної системи:

- модуль симуляції сонячної генерації, який формує дані про потужність генерації з урахуванням часу доби, сезону, погодних умов та параметрів розміщення сонячних панелей;
- модуль симуляції доступності централізованої електромережі, який відтворює періоди наявності та відсутності мережевого живлення, а також сценарії нестабільної роботи енергосистеми;
- модуль симуляції споживання електроенергії, який описує навантаження лабораторного стенду та зміну споживаної потужності в часі;
- модуль моделювання стану акумуляторної батареї, який розраховує рівень заряду, режим заряджання, розряджання та захисні стани батареї;
- модуль автоматизації режимів заряджання, який приймає рішення щодо використання доступних джерел енергії та вибору режиму роботи системи;

- серверну частину з API для обміну даними між підсистемами, збереження історичних даних і надання доступу до поточного стану системи;
- графічний веб-інтерфейс для перегляду поточних і історичних даних, вибору режиму роботи, запуску симуляції та аналізу рішень системи автоматизації.

Розроблені модулі мають працювати як єдина програмна система, у якій симуляційні дані формуються окремими підсистемами, зберігаються у базі даних, обробляються алгоритмами автоматизації та відображаються у веб-інтерфейсі. Результатом виконання роботи має бути програмне забезпечення, яке демонструє принципи автоматизованого керування режимами заряджання акумуляторних батарей у системі з сонячною генерацією, акумуляторним накопичувачем, навантаженням та централізованим електропостачанням.

Висновки до розділу 1

У першому розділі було обґрунтовано актуальність розроблення програмного забезпечення для автоматизації режимів заряджання акумуляторних батарей. Показано, що така задача є актуальною як у короткостроковій перспективі, з огляду на пошкодження енергетичної інфраструктури України та потребу в резервних джерелах живлення, так і в довгостроковій перспективі, у контексті підвищення ефективності використання відновлюваних джерел енергії та систем накопичення.

Було сформульовано мету роботи, яка полягає у створенні програмної системи для симуляції основних процесів лабораторного стенду та підтримки автоматизованого вибору режимів роботи системи. Визначено, що програмне забезпечення має забезпечувати моделювання сонячної генерації, споживання електроенергії, стану акумуляторної батареї та доступності централізованої електромережі, а також збереження, відображення й подальшу обробку отриманих даних.

2 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ДО АВТОМАТИЗАЦІЇ РЕЖИМІВ ЗАРЯДКИ ТА СТВОРЕННЯ СПРОЩЕНОГО ЦИФРОВОГО ДВІЙНИКА

При розгляді задачі розроблення програмного модуля автоматизації зарядки акумуляторів швидко виникає розуміння, що цей модуль є глибоко залежним від вхідних даних, а отже задачу не можна розглядати окремо від решти системи та даних. Щоб продемонструвати здатність алгоритму приймати обґрунтовані рішення на основі даних про стан системи, важливо мати якісні вхідні дані. У кіберфізичній системі лабораторного стенду такі дані генеруються природним чином завдяки фізичним пристроям, датчикам та взаємодії людей із системою.

Враховуючи, що в межах лабораторії немає доступу до таких даних із реальної системи, виникає необхідність в їх якісній симуляції. Модуль автоматизації зарядки батарей має спиратися на інформацію про доступну сонячну генерацію, оцінювати ризик відключень на основі даних про наявність електромережі, запас енергії в акумуляторі та поточне споживання. На основі цих даних він має обирати оптимальний режим заряджання акумуляторів та розподілу енергії в системі, щоб мінімізувати вартість спожитої енергії, пріоритизувати використання сонячної генерації та водночас забезпечити достатній запас енергії в акумуляторі на випадок відключень.

Беручи ці факти до уваги, основні зусилля було спрямовано на аналіз моделей сонячної генерації, підходів до симуляції акумуляторів, систем EMS/BMS, моделей споживання, сценаріїв роботи за нестабільної електромережі та симуляції самої мережі й її доступності. Такий огляд потрібен, щоб зрозуміти, які частини задачі вже добре досліджені, які методики використовуються для схожих задач та чи підходять готові рішення для нашої роботи загалом.

2.1 Підходи до симуляції сонячної генерації

Першим пунктом розробки було заплановано модуль симуляції сонячної генерації, метою якого є отримання реалістичних даних на основі орієнтації та розміщення сонячних панелей у просторі. У найпростішому вигляді графік генерації можна представити як параболічну криву з піком у середині світлового дня, однак таке відображення не демонструє добових коливань залежно від сезону, орієнтації панелей та погоди.

У повноцінних симуляторах поведінки фотоелектричних систем розрахунок виконується через побудову детальної програмної моделі сонячної установки. Наприклад, NREL SAM не обмежується простим множенням номінальної потужності панелей на умовний коефіцієнт погоди. У цій системі продуктивність PV-установки розраховується як повний ланцюг перетворень: використовуються погодні дані для конкретної місцевості, зокрема пряме, розсіяне та сумарне сонячне випромінювання, температура повітря і швидкість вітру. Далі модель визначає положення Сонця, перераховує випромінювання на площину панелі з урахуванням її нахилу та азимута, оцінює температуру модуля і переходить до електричної моделі PV-системи. На цьому етапі враховуються характеристики модулів, схема їх з'єднання, робота інвертора, DC/AC-перетворення та різні типи втрат: затінення, забруднення, втрати в кабелях, невідповідність параметрів модулів, деградація та інші фактори [1].

Такий підхід робить SAM корисним інструментом для проєктування реальних фотоелектричних установок, однак водночас показує складність повноцінного моделювання сонячної генерації. Для коректної роботи такої моделі потрібен повний набір вхідних параметрів, частина яких у межах навчального стенду або відсутня, або не має принципового впливу на точність симуляції тих даних, які необхідні нам. Тому для SmartEnergy Lab використання моделі рівня SAM було б надмірним: вона спеціалізується на точному енергетичному

розрахунку СЕС, тоді як у межах роботи достатньо наближених до реальності даних моментальної генерації.

Аналогічною за ідеєю є Python-бібліотека `pvl`. Це бібліотека моделей для сонячної геометрії, опромінення, температури модуля і вихідної потужності PV-системи [2]. Вона добре підходить для програмної інтеграції, оскільки дозволяє використовувати лише потрібні частини моделі. Для цієї роботи важливим є сам підхід: сонячну генерацію доцільно розраховувати не як випадкове значення, а як результат залежності від часу доби, сезону, географічного положення, орієнтації панелей і погодного коефіцієнта.

2.2 Підходи до моделювання акумуляторних батарей

Наступною частиною системи виступає акумулятор. Він є резервним сховищем енергії, яке дозволяє зберігати надлишок сонячної енергії або енергію з мережі й віддавати цей запас, коли мережа зникає або генерації недостатньо для покриття навантаження.

Найпростіші моделі акумулятора базуються на балансі енергії та визначенні поточного стану акумулятора за його фізичними параметрами. Параметри нормальної роботи залежать від будови та внутрішньої хімії батареї. Контроль її стану в подібних системах зазвичай виконується окремим модулем управління батареєю, відомим як BMS, або battery management system.

Найпростішим параметром у моделі акумулятора є SoC, тобто State of Charge, поточний рівень заряду батареї. У спрощених системах його приблизно визначають за напругою акумулятора або через баланс отриманої та витраченої енергії. Окремо використовують SoH, тобто State of Health, стан зносу батареї. Його можна оцінити, порівнявши паспортну ємність із фактичною кількістю енергії, яку акумулятор здатний віддати до досягнення критичного рівня заряду. Ці показники використовуються в BMS-системах для контролю безпечної роботи батареї, обмеження глибокого розряду та оцінювання залишкового ресурсу [3], [4].

Більш точними за системи енергетичного балансу є моделі еквівалентних електричних схем. У них акумулятор представлений як джерело напруги з внутрішнім опором і додатковими конденсаторно-резистивними ланками, які наближено описують процеси накопичення та віддачі енергії. Такі моделі не відтворюють повну хімію акумулятора, але дозволяють точніше демонструвати зміну напруги під навантаженням і реакцію батареї на заряд або розряд. Через це вони часто використовуються в задачах оцінювання SoC і SoH як компроміс між точністю та складністю [4].

Найскладнішими є фізико-хімічні моделі акумуляторів. Вони описують не тільки зовнішні електричні параметри батареї, а й процеси всередині її елементів: рух іонів, дифузію, електрохімічні реакції, нагрівання та деградацію. Такі моделі точні, але потребують великої кількості параметрів і зазвичай застосовуються для розробки або оптимізації акумуляторів. Прикладом такого інструмента є PyBaMM, який дозволяє будувати математичні моделі батарей різного рівня деталізації [5]. Для більш тривіальних задач рівень такої деталізації є надмірним.

Для симуляційної моделі лабораторного стенду достатньо припущення, що акумулятор має рівномірний процес зношення в межах заявленої виробником кількості циклів заряду-розряду та притаманну своєму типу криву розряду. Це дозволяє отримати достатньо наближену до реальності модель, на поведінку якої можна спиратися під час розрахунків.

2.3 Підходи до автоматизації режимів зарядки, EMS та BMS

У системах СЕС існують три основні компоненти: MPPT, BMS та EMS-контролери. Це можуть бути окремі фізичні пристрої, але все частіше функціонал усіх трьох об'єднують у межах одного гібридного інвертора, де кожна підсистема виконує свою роль.

MPPT-контролер, або Maximum Power Point Tracking контролер, шукає таку робочу точку сонячної панелі, у якій добуток її напруги та струму є максимальним.

Для цього він через DC-DC перетворювач змінює навантаження, вимірює поточні значення напруги й струму, розраховує потужність та поступово підлаштовує режим роботи так, щоб отримати найбільшу доступну генерацію [6].

BMS, або Battery Management System, є системою керування батареєю. Вона відстежує напругу комірок, струм, температуру і на основі цих даних виконує заряджання, балансування елементів і захищає батарею від критичних станів, які виводять її за межі нормальних діапазонів напруги, температури або струму. Також BMS може тестувати стан “здоров’я” батареї, помічаючи відхилення параметрів від паспортних [3], [4].

EMS, або Energy Management System, працює вже на рівні всієї енергетичної системи. Вона розподіляє енергетичні потоки та на основі параметрів користувача і поточного стану вирішує, що робити з енергією: заряджати батарею, жити навантаження, в яких пропорціях і з якого джерела це робити.

За своєю суттю програмний модуль автоматизації режимів зарядки належить саме до EMS-рівня. Він аналізує доступну генерацію, стан акумулятора, навантаження та доступність мережі, після чого вибирає доцільний режим роботи.

Найпростішим і практично зручним варіантом EMS є rule-based алгоритми, тобто алгоритми на основі правил. Вони не завжди виглядають так “розумно”, як прогнозні або оптимізаційні системи, але в реальних малих сонячних системах їхня ефективність може бути близькою до складніших підходів. Причина проста: складні моделі сильно залежать від якості прогнозу, точності моделі батареї та коректності вхідних даних. Якщо ці дані неточні, перевага складного алгоритму швидко зменшується, тоді як rule-based логіку легше перевірити, пояснити і стабільно використовувати в навчальному стенді [7], [8]. Саме тому на багатьох комерційних гібридних інверторах реалізовано кілька режимів з пріоритетами розподілу енергії.

2.4 Підходи до моделювання споживання електроенергії

Окремою частиною симуляційної системи є моделювання споживання електроенергії. Для модуля автоматизації режимів заряджання акумуляторів споживання є одним із ключових вхідних параметрів, оскільки саме воно визначає, скільки енергії потрібно подати на навантаження в поточний момент і скільки енергії необхідно залишити в акумуляторі для резервної роботи системи.

Найпростішим підходом є використання фіксованого або усередненого профілю навантаження. У такому випадку споживання задається як набір типових значень для різних періодів доби. Наприклад, вдень навантаження може бути вищим через роботу лабораторного обладнання, а вночі знижуватися до мінімального рівня. Такий підхід простий у реалізації, але погано відображає випадкові короточасні піки, які виникають під час увімкнення окремих приладів.

Більш реалістичним є підхід на основі профілів кінцевого споживання. У таких моделях загальне навантаження розкладається на окремі групи споживачів або приладів. Для кожного з них можна задати номінальну потужність, черговий режим, активний режим, пікове споживання та ймовірність увімкнення. Подібна логіка використовується в наборах *end-use load profiles*, де споживання подається з часовою дискретизацією та розподілом за категоріями кінцевого використання енергії [9].

Ще точнішим є стохастичне моделювання навантаження. У цьому випадку робота споживачів визначається не лише фіксованим розкладом, а й випадковими подіями: появою користувача, запуском обладнання, зміною режиму роботи або короточасним піковим навантаженням. Такий підхід дозволяє краще відтворювати поведінку малих енергетичних систем, де споживання часто залежить не від стабільного виробничого графіка, а від дій людей та нерегулярного використання окремих пристроїв [10].

У межах цієї роботи для лабораторного стенду SmartEnergy Lab було обрано саме спрощений стохастичний підхід до моделювання споживання. Навантаження

описується через набір типових приладів, які можуть бути підключені до інвертора. Для кожного приладу задаються режими споживання, зокрема очікування, типовий режим, активний режим і пікове споживання, а також імовірність появи події його використання. Завдяки цьому модель формує не сталу лінію навантаження, а змінний у часі профіль споживання з випадковими піками, які краще відповідають поведінці малої лабораторної системи.

Таким чином, у межах роботи моделювання споживання виконується як комбінація базового добового профілю та стохастичних подій увімкнення окремих приладів. Такий підхід залишається достатньо простим для реалізації та перевірки, але водночас формує дані, які реально впливають на рішення EMS-модуля щодо заряджання акумулятора, живлення навантаження та використання централізованої електромережі.

2.5 Моделювання нестабільної електромережі та аварійних режимів

У багатьох класичних моделях систем із сонячною генерацією та акумулятором зовнішня електромережа розглядається як стабільне джерело живлення або звичайний канал обміну енергією. Для сучасних українських умов таке припущення є хибним. Через пошкодження енергетичної інфраструктури мережа може бути недоступною, працювати з обмеженнями або поступово відновлюватися після аварій. Це є однією з причин появи малих СЕС в університетах, квартирах та приватних будинках.

У літературі ця тема зазвичай розглядається через моделювання стійкості енергосистем і аварійних відключень. Такі підходи аналізують поведінку системи під час тривалих перебоїв, дефіциту генерації, пошкоджень обладнання та подальшого відновлення. У роботах про децентралізовані енергетичні системи підкреслюється, що довгі аварійні події не можна нормально описати лише середнім відсотком доступності мережі [11]. Для цього потрібні сценарії, у яких

враховується тривалість події, інтенсивність, момент початку та період повернення до нормальної роботи.

В Україні існують відкриті аналітичні звіти про пошкодження енергетичної інфраструктури, дефіцит генерації та потребу в розподілених джерелах енергії. Наприклад, ІЕА у 2024 році розглядала питання енергетичної безпеки України та роль локальних джерел генерації у підвищенні стійкості системи [12], [13]. Також існують моделі дефіциту електроенергії для України на рівні всієї енергосистеми, зокрема з використанням PyPSA [14].

Однак такі джерела не дають готової простої моделі для невеликого лабораторного стенду, якому в основі потрібні стан наявності чи відсутності електромережі та тривалість цих станів. Вони працюють на рівні всієї енергосистеми, окремих регіонів або великих сценаріїв розвитку. Для SmartEnergy Lab така деталізація є надмірною, а частина реальних даних про стан інфраструктури або недоступна, або не може використовуватися відкрито. Тому в межах цієї роботи доцільніше використати сценарний підхід: нормальна робота мережі, обмежена доступність, повне відключення та поступове відновлення.

Така модель додає ще одну важливу частину системи, яка впливає на споживання, використання акумулятора і поведінку програмного модуля автоматизації зарядки. Якщо мережа нестабільна, система має інакше оцінювати запас енергії в батареї, пріоритети навантаження та момент, коли доцільно заряджати або зберігати акумулятор для резерву.

Висновки до розділу 2

У другому розділі було розглянуто основні підходи до моделювання складових системи з сонячною генерацією, акумуляторним накопичувачем, навантаженням та централізованою електромережею. Проведений аналіз показав, що для кожного окремого компонента вже існують спеціалізовані методи та програмні інструменти. Сонячну генерацію можна моделювати за допомогою

систем на кшталт SAM або бібліотек на зразок pvlib. Поведінку акумуляторної батареї можна описувати через простий баланс енергії, моделі еквівалентних електричних схем або складні фізико-хімічні моделі. Для керування енергетичними потоками застосовуються EMS-рішення, алгоритми на основі правил, прогнозні та оптимізаційні підходи. Окремо було розглянуто моделювання споживання електроенергії та нестабільної роботи централізованої електромережі.

Основна проблема полягає не у відсутності технічних рішень, а в тому, що більшість із них спеціалізовані на окремій сфері задач. Повноцінні інструменти для розрахунку сонячної генерації потребують великої кількості вхідних параметрів і більше орієнтовані на проєктування реальних СЕС, ніж на демонстрацію логіки роботи навчального стенду. Моделі акумуляторів високого рівня деталізації потребують значної кількості параметрів і є надмірними для задачі перевірки алгоритму автоматизації заряджання. Готові EMS-рішення часто розраховані на конкретну апаратну конфігурацію або промислове застосування, тому їх важко безпосередньо використати в межах лабораторного стенду SmartEnergy Lab.

З огляду на це, у межах роботи є доцільним розроблення власних спрощених симуляційних модулів для моделювання сонячної генерації, акумуляторного накопичувача, навантаження та стану електромережі. Такий підхід дозволяє забезпечити необхідний рівень деталізації без надмірної складності та зосередитися на основній задачі дослідження – створенні rule-based системи автоматизації режимів заряджання з урахуванням ризиків та поточного стану енергетичної системи.

3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ІНСТРУМЕНТІВ РОЗРОБКИ

Для реалізації програмного модуля було обрано стек інструментів, який дозволяє швидко розробити демонстраційну вебсистему, розділити логіку симуляції та інтерфейс користувача, а також забезпечити відтворюваний запуск проєкту. Основними вимогами були підтримка API, зручна робота з часовими даними, можливість побудови графіків, контейнеризація та подальше розширення системи.

У проєкті використано такі основні засоби розробки:

- Python мова програмування серверної частини;
- FastAPI– фреймворк для створення API;
- Uvicorn– сервер для запуску FastAPI-застосунку;
- Pydantic та Pydantic Settings– опис і перевірка структур даних
- SQLite– збереження симуляційних, погодних та історичних даних;
- PyYAML– робота з конфігураційними файлами;
- pandas та numpy– обробка часових рядів і числових даних;
- pvlib– розрахунок сонячної геометрії;
- requests– звернення до зовнішніх API;
- matplotlib– побудова тестових графіків та попередній перегляд даних;
- pytest– перевірка роботи окремих частин backend;
- React– побудова вебінтерфейсу;
- Vite– запуск і збирання frontend-застосунку;
- Recharts– побудова графіків у вебінтерфейсі;
- Docker та Docker Compose– контейнеризація і запуск усієї системи.

3.1 Вибір серверної частини

Для серверної частини було обрано Python – мову програмування, яка дозволяє швидко працювати та ефективніше інтегрувати системи [15], оскільки він добре підходить для задач симуляції, роботи з даними та швидкого створення прототипів. У межах цієї роботи Python використовується для розрахунку сонячної генерації, моделювання стану електромережі, обробки погодних даних, збереження результатів і підготовки відповідей для інтерфейсу.

FastAPI було обрано як основний backend-фреймворк, оскільки він дозволяє швидко створювати API та зручно передавати дані між серверною частиною і frontend. В офіційній документації FastAPI описано як сучасний і швидкий вебфреймворк для створення API з використанням Python та стандартних анотацій типів [16]. Це важливо для проєкту, тому що вебінтерфейс не працює напряму з базою даних, а отримує всі значення через окремі API-запити.

Для запуску FastAPI використовується Uvicorn. Він виконує роль сервера застосунку та забезпечує обробку запитів від frontend-частини. В офіційній документації Uvicorn визначено як реалізацію ASGI вебсервера для Python [17].

3.2 Засоби роботи з даними та конфігурацією

Для опису структур даних використовуються Pydantic та Pydantic Settings. Вони потрібні для того, щоб дані між частинами системи передавалися у зрозумілому форматі, а помилки конфігурації або некоректні значення було легше виявити ще на етапі виконання програми.

Для збереження даних використано SQLite разом із SQLAlchemy. SQLite достатньо для демонстраційної системи, оскільки проєкт не потребує окремого сервера бази даних. У локальній базі зберігаються погодні дані, історична та прогнозна сонячна генерація, а також дані, необхідні для швидкого відображення графіків.

Файл конфігурації з параметрами стенду зберігається у форматі YAML, тому для його зчитування використовується PyYAML. Через конфігурацію можна змінювати параметри системи без прямого редагування програмної логіки.

3.3 Бібліотеки для симуляції та обробки даних

Для роботи з числовими значеннями та часовими рядами використовуються `pandas` і `numpy`. Вони застосовуються при підготовці історичних даних, обробці проміжків часу, розрахунках генерації та підготовці значень для графіків.

Бібліотека `pvlib` використовується для розрахунку положення Сонця [2]. У межах цієї роботи вона не застосовується як повний готовий симулятор сонячної електростанції, а використовується лише там, де це справді потрібно: для визначення сонячної геометрії. Подальший розрахунок генерації виконується спрощеною локальною моделлю.

Для отримання погодних даних використовується `requests`. Через нього `backend` звертається до зовнішнього погодного сервісу, отримує історичні та прогнозні значення, після чого зберігає їх у локальній базі. Для попередньої перевірки згенерованих даних використовувався `matplotlib`, який дозволив швидко побудувати графіки та перевірити правдоподібність симуляції до виведення її у вебінтерфейс.

3.4 Вибір frontend-інструментів

Для клієнтської частини було обрано `React`. `React` дозволяє будувати інтерфейс користувача з окремих частин, які називаються компонентами [18]. Це добре підходить для структури цієї системи: віджети сонячної генерації, віджети електромережі, погодний блок і майбутні блоки акумулятора та автоматизації зарядки можуть існувати як окремі частини сторінки.

Vite використовується як засіб запуску та збирання frontend-застосунку. В офіційній документації він описується як швидкий інструмент збирання для вебзастосунків [19]. Його перевага для цієї роботи полягає у простому налаштуванні та швидкому локальному запуску, що важливо під час розробки демонстраційного прототипу.

Для побудови графіків у вебінтерфейсі використано Recharts. Графіки є важливою частиною системи, оскільки саме через них користувач бачить зміну сонячної генерації, стан електромережі, історичні дані та інші часові залежності.

3.5 Контейнеризація проєкту

Для запуску системи використано Docker та Docker Compose. Docker Compose є інструментом для визначення та запуску багатоконтейнерних застосунків [20]. Backend, frontend і службові процеси оновлення даних запускаються як окремі сервіси. Це спрощує запуск проєкту, оскільки не потрібно вручну налаштовувати всі залежності та запускати кожну частину окремо.

Контейнеризація також робить систему зручнішою для демонстрації та передачі іншим користувачам. Достатньо мати встановлений Docker, після чого проєкт можна запустити однією командою. Для практичної роботи це важливо, бо головною метою є не складне адміністрування середовища, а стабільна демонстрація роботи програмного модуля.

Висновки до розділу 3

У третьому розділі було обґрунтовано вибір програмних засобів, використаних для реалізації системи. Обраний стек відповідає задачам роботи, оскільки дозволяє розділити серверну логіку, симуляційні модулі, збереження даних, вебінтерфейс та запуск системи в окремому середовищі.

Python було використано як основну мову серверної частини, оскільки він підходить для задач симуляції, обробки часових рядів, роботи з числовими даними та швидкого створення прототипів. FastAPI забезпечує побудову API, через яке frontend отримує дані від backend, а Uvicorn використовується для запуску серверного застосунку.

Для збереження даних було обрано SQLite, оскільки для демонстраційної системи не потрібен окремий сервер бази даних. Такий підхід спрощує розгортання проєкту та дозволяє зберігати історичні, погодні й симуляційні дані локально. Конфігураційні параметри винесено у YAML-файл, що дає можливість змінювати параметри стенду без прямого редагування програмної логіки.

Для клієнтської частини було обрано React, оскільки він дозволяє будувати інтерфейс із незалежних компонентів. Це відповідає структурі розроблюваної системи, де окремі віджети відповідають за відображення сонячної генерації, стану електромережі, погодних даних та інших модулів. Для побудови графіків використано Recharts, оскільки часові залежності є основним способом відображення результатів симуляції.

Docker та Docker Compose забезпечують відтворюваний запуск системи й дозволяють запускати backend, frontend та службові процеси як окремі сервіси. Це робить проєкт зручнішим для демонстрації, перевірки та подальшої передачі іншим користувачам.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У цьому розділі описано програмну реалізацію основних модулів системи, які забезпечують моделювання роботи лабораторного стенду SmartEnergy Lab та автоматизацію режимів заряджання акумуляторної батареї. Розглядаються модулі симуляції сонячної генерації, доступності централізованої електромережі, стану акумуляторної батареї, споживання електроенергії та EMS. Кожен із цих модулів формує окрему частину стану системи, а їхня спільна робота дозволяє виконувати інтегровану симуляцію та перевіряти логіку автоматичного розподілу енергії.

4.1 Програмна реалізація симуляції сонячної генерації

Модуль симуляції сонячної генерації призначений для симуляції даних сонячної генерації на основі спрощеної моделі розрахунку продукованої потужності фотоелектричних панелей лабораторного стенду SmartEnergy Lab. Його основна задача полягає не в інженерному проєктуванні сонячної електростанції, а в отриманні достатньо правдоподібних даних, які можуть використовуватися іншими модулями програмної системи.

У дипломній роботі цей модуль розглядається детальніше як частина симуляційного ядра, тобто як альтернативне джерело вхідної енергії для подальшої автоматизації режимів заряджання акумуляторної батареї.

Необхідність такого модуля пояснюється, по-перше, тим, що створена система відтворює реальний лабораторний стенд де такі панелі присутні. По-друге, що алгоритм EMS не може приймати обґрунтовані рішення без інформації про доступну сонячну потужність. Якщо генерація перевищує поточне навантаження, надлишок може бути спрямований на заряджання батареї. Якщо генерація низька, система має вирішувати, чи варто використовувати мережу, розряджати акумулятор або зберігати його резерв. Тому сонячна генерація є одним із базових вхідних параметрів для всієї логіки автоматизації.

Розроблений модуль формує два основні типи даних: історичні та прогнозовані точки сонячної генерації. Це пов'язано з тим, що система працює не з реальними даними зі стенду, а з інформацією про погоду через OpenMeteo API, що в свою чергу унеможливорює гарантований безперервний потік даних, тож сонячна генерація спирається як на історичні дані так і на прогноз, але лише перші зберігаються довгостроково.

4.1.1 Конфігураційні параметри сонячної станції

Вхідні параметри сонячної станції задаються у YAML-конфігурації. Це дозволяє змінювати модель стенду без редагування програмного коду. У конфігурації описуються географічні координати встановлення, часовий пояс, орієнтація панелей, кут нахилу, тип панелей, їх кількість і основні електричні характеристики. Такий підхід робить модуль гнучким: за потреби його можна адаптувати під іншу станцію або іншу конфігурацію панелей.

У поточній реалізації моделюється сонячна станція SmartEnergy Lab, розташована в Києві. Для розрахунків використовується часовий пояс Europe/Kyiv, широта 50.448997 та довгота 30.464642 що відповідає координатам даху кафедри ІПЗЕ. Панелі орієнтовані на південь, тобто мають азимут 180° , і встановлені під кутом 35° . Така орієнтація є типовою для стаціонарної сонячної установки в північній півкулі, де максимальна генерація очікується в середині дня.

Модель використовує два фотоелектричні модулі AXIOMA Energy 200W, з'єднані в одну послідовну збірку. Номінальна потужність однієї панелі становить 200 Вт, тому сумарна встановлена потужність масиву дорівнює 400 Вт. Номінальна напруга однієї панелі задана як 35 В, а для двох панелей у послідовному з'єднанні номінальна напруга масиву становить приблизно 70 В.

Таблиця 4.1– Основні параметри сонячної станції, використані в симуляції

Назва станції	SmartEnergy Lab	Ідентифікатор лабораторного стенду, для якого формується симуляція
Місце розташування	Київ	Використовується для прив'язки до реальної географічної локації
Широта	50,448997	Вхідний параметр для розрахунку положення Сонця
Довгота	30,464642	Вхідний параметр для розрахунку положення Сонця
Часовий пояс	Europe/Kyiv	Використовується для переходу між UTC та локальним часом
Тип панелі	AXIOMA Energy 200W	Тип фотоелектричного модуля в конфігурації стенду
Кількість панелей	2	Кількість модулів у сонячному масиві
Схема з'єднання	Послідовне з'єднання	Визначає сумарну напругу сонячного масиву
Потужність однієї панелі	200 Вт	Номінальна максимальна потужність одного модуля
Сумарна встановлена потужність	400 Вт	Верхня межа ідеальної потужності масиву
Номінальна напруга однієї панелі	35 В	Базове значення для допоміжної оцінки напруги масиву
Орієнтація панелей	Південь	Основний напрямок, у який орієнтована площа панелей
Азимут панелей	180°	Використовується під час розрахунку коефіцієнта падіння випромінювання
Кут нахилу панелей	35°	Визначає нахил площини панелей відносно горизонту

Параметри з таблиці 4.1 використовуються як основа для всіх подальших розрахунків. Географічні координати та часовий пояс потрібні для визначення положення Сонця. Азимут і кут нахилу панелей визначають, наскільки ефективно випромінювання потрапляє на площину модулів. Номінальна потужність і кількість панелей задають верхню межу доступної генерації, яку модель не повинна перевищувати.

4.1.2 Розрахунок ідеальної сонячної генерації

Першим етапом роботи модуля є формування ідеальної сонячної генерації. Під ідеальною генерацією в цьому випадку розуміється потужність, яку панелі могли б виробити за заданої геометрії Сонця та панелей без урахування фактичної погоди. Такі дані необхідні як базові, до яких пізніше застосовується погодне коригування.

Для розрахунку положення Сонця використовується бібліотека `pvl`. Вона дозволяє для кожного моменту часу визначити висоту Сонця над горизонтом і його азимут. Ці параметри залежать від дати, часу, географічних координат і часового поясу. У результаті для кожної часової точки система отримує не умовні параметри, а реальну сонячну геометрію для конкретної локації стенду.

Після отримання сонячної геометрії модуль розраховує коефіцієнт падіння випромінювання на площину панелей. Для цього напрямки на Сонце та нормаль до площини панелі подаються як просторові вектори. Їхній скалярний добуток показує, під яким кутом Сонце розташоване відносно площини панелей. Якщо Сонце світить перпендикулярно до площини панелі, коефіцієнт наближається до одиниці. Якщо Сонце знаходиться збоку або нижче горизонту, коефіцієнт зменшується або дорівнює нулю.

Крім прямої складової, у моделі враховано невелику умовну фонову складову. Вона потрібна для наближеного врахування розсіяного світла, яке може давати малу частину генерації навіть тоді, коли пряме падіння випромінювання на панелі не є максимальним. Ця складова залежить від висоти Сонця і обмежена невеликим значенням, щоб не спотворювати загальний профіль генерації.

Ідеальна потужність для кожного моменту часу розраховується як сума прямої та фонових-складових, але не може перевищувати сумарну встановлену потужність масиву. Якщо Сонце перебуває нижче горизонту, усі складові генерації примусово встановлюються рівними нулю. Завдяки цьому модель природно

формує нічний нульовий період, ранкове зростання, денний максимум і вечірній спад.

Результатом цього етапу є набір 15-хвилинних точок ідеальної генерації. Для кожної точки зберігаються UTC-час, локальний час, висота Сонця, азимут Сонця, коефіцієнт падіння, фоновий-коефіцієнт, пряма складова потужності, фонові-складова та сумарна ідеальна потужність. Така деталізація дозволяє не тільки використовувати готове значення потужності, а й аналізувати, чому в певний момент генерація має саме такий рівень.

4.1.4 Отримання погодних даних і формування погодного коефіцієнта

Ідеальна генерація відображає лише геометрично можливу потужність панелей. У реальних умовах фактична генерація суттєво залежить від погоди. Хмарність, туман, дощ, сніг або гроза можуть зменшувати потужність у кілька разів, навіть якщо геометричне положення Сонця є сприятливим. Тому другим важливим етапом роботи модуля є погодні коригування.

Погодні дані отримуються з Open-Meteo. Для роботи модуля використовуються як історичні дані, так і прогноз. До набору погодних параметрів входять температура повітря, відсоток хмарності, опади, дощ, сніг, та відповідні погодні коди.

Погодний код перетворюється на узагальнений стан погоди. У моделі використовуються такі стани: `clear`, `partly_cloudy`, `cloudy`, `fog`, `drizzle`, `rain`, `snow`, `thunderstorm` та `unknown`. Для кожного стану задано діапазон погодного коефіцієнта. Коефіцієнти та описи станів наведені у таблиці 4.2. Цей коефіцієнт множиться на ідеальну генерацію та зменшує її відповідно до погодних умов.

Для станів, пов'язаних з опадами, коефіцієнт вибирається безпосередньо з відповідного діапазону. Для станів `clear`, `partly_cloudy` і `cloudy` додатково враховується відсоток хмарності. Чим більша хмарність, тим вища ймовірність

того, що для точки буде застосовано коефіцієнт з нижчого діапазону. Це дозволяє отримати більш нерівномірний та наближений до реальності профіль генерації.

Таблиця 4.2– Діапазони погодніх коефіцієнта для різних погодніх станів

Погодний стан	Діапазон коефіцієнта	Опис стану	Вплив на генерацію
clear	0.85–1.00	Ясна погода	Мінімальне зменшення генерації
partly_cloudy	0.55–0.85	Мінлива хмарність	Помірне зменшення потужності
cloudy	0.25–0.55	Хмарна погода	Суттєве зменшення генерації
fog	0.10–0.35	Туман	Сильне розсіювання та поглинання світла
drizzle	0.20–0.50	Мряка	Помітне зниження генерації
rain	0.10–0.40	Дощ	Значне зниження доступної потужності
snow	0.05–0.30	Сніг	Дуже низька генерація
thunderstorm	0.05–0.25	Гроза	Мінімальна доступна генерація
unknown	0.50–0.80	Невизначений стан	Середній умовний коефіцієнт

Щоб результат симуляції залишався відтворюваним, випадкова складова не є повністю випадковою. Для кожної часової точки формується детермінований генератор на основі часу, погодніх коду, хмарності та seed-значення. Це дозволяє гарантувати що у випадку повторного заповнення бази даних на іншому пристрої дані симуляції співпадуть за умови однакових вхідних даних.

4.1.5 Формування історичної та прогнозної генерації

Після розрахунку ідеальної генерації та погодніх коефіцієнта модуль формує погוזозалежну сонячну генерацію. Для кожної часової точки ідеальна потужність множиться на погодніх коефіцієнт, після чого отримується значення

доступної потужності панелей (рис. 4.1). Саме це значення надалі використовується в інтегрованій симуляції як доступна сонячна потужність.

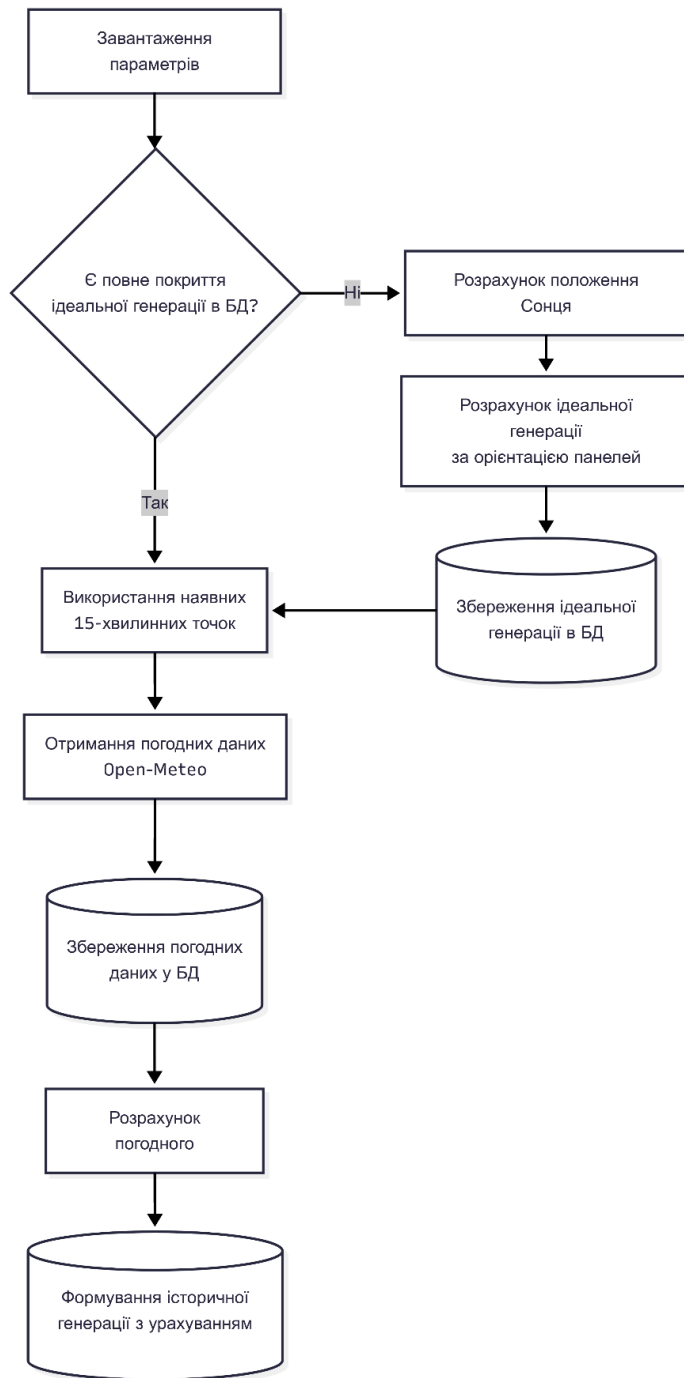


Рисунок 4.1 – Блок-схема алгоритму формування історичної та прогнозної сонячної генерації

Модуль формує два основні набори даних: історичну генерацію та прогнозну генерацію. Історична генерація використовується для аналізу вже сформованого

періоду та відтворення минулого стану системи. Прогнозна генерація формується на основі прогнозу погоди й дозволяє оцінити майбутнє надходження сонячної енергії яке необхідне для безшовної роботи веб-інтерфейсу.

З технічної точки зору модуль спочатку перевіряє, чи є в базі даних уже розраховані точки ідеальної генерації для потрібного періоду. Якщо вони є, система використовує їх повторно. Якщо ні, виконується розрахунок положення Сонця та формування нових точок. Такий підхід зменшує обсяг повторних обчислень, оскільки ідеальна генерація залежить тільки від геометрії станції та часу, а не від поточної погоди.

Як видно з рисунка 4.1, модуль спочатку працює з конфігурацією та базовим розрахунком ідеальної генерації, а вже після цього переходить до погодного коригування. Це важливий архітектурний поділ. Ідеальна генерація є стабільною основою, яку можна зберігати й використовувати повторно, а погодозалежна генерація є змінним шаром, який залежить від конкретних погодних даних.

Окремо на схемі показано формування історичної та прогнозованої генерації. Обидва набори даних мають однакову загальну логіку розрахунку, але різні джерела погодної інформації. Для історичного періоду використовуються фактичні погодні спостереження, а для майбутнього періоду – прогноз. Це дозволяє модулю однаково працювати як з минулими, так і з майбутніми часовими проміжками.

Збереження результатів у базі даних потрібне не лише для швидкого доступу до значень. Воно також дозволяє іншим модулям працювати з єдиним джерелом даних. Інтегрована симуляція не повинна щоразу самостійно перераховувати сонячну генерацію. Вона отримує вже підготовлені точки та використовує їх у розрахунку енергетичного балансу.

4.1.6 Допоміжна оцінка напруги та струму сонячного масиву

Крім основного значення доступної потужності, у модулі передбачено допоміжну оцінку електричних параметрів сонячного масиву: напруги та струму.

Цей розрахунок не є повною фізичною моделлю PV-модуля або MPPT-контролера. Його задача – отримати наближені електричні значення, узгоджені з поточною змодельованою потужністю.

Для цього використовується сумарна встановлена потужність масиву, напруга в точці максимальної потужності та напруга холостого ходу панелі. Потужність обмежується діапазоном від нуля до максимальної встановленої потужності масиву. Якщо потужність дуже мала, напруга та струм приймаються рівними нулю. Якщо потужність додатна, модель оцінює робочу напругу масиву та визначає струм як відношення потужності до напруги.

У поточній конфігурації дві панелі з'єднані послідовно, тому напруга масиву приблизно відповідає сумі напруг окремих панелей. При цьому струм визначається вже поточною потужністю генерації. Такий підхід дозволяє отримати правдоподібні значення напруги та струму без надмірного ускладнення моделі.

Важливо підкреслити, що основним параметром для EMS залишається не напруга або струм, а доступна потужність у ватах. Саме вона визначає, яку частину навантаження може покрити сонячна генерація та скільки енергії потенційно можна спрямувати на заряджання акумуляторної батареї. Напруга та струм у цьому модулі виконують допоміжну інформаційну роль.

4.1.7 тестування роботи модулі

Для перевірки роботи моделі доцільно використовувати допоміжні графіки, сформовані на основі збережених симуляційних даних. Першим таким графіком є порівняння ідеальної та погодозалежної генерації протягом одного дня. Ідеальна генерація має плавну форму, яка визначається геометрією Сонця та панелей. Погодозалежна генерація повторює загальну форму ідеальної кривої, але має провали та коливання, спричинені хмарністю й опадами.

Приклад порівняння ідеальної та погодозалежної сонячної генерації наведено на рисунку 4.2.

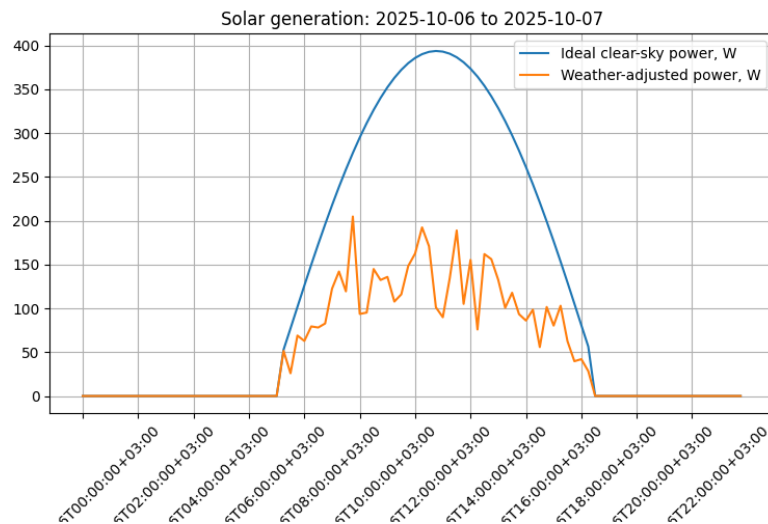


Рисунок 4.2 – Порівняння ідеальної та погодозалежної сонячної генерації

На рисунку 4.2 видно різницю між теоретично можливою та скоригованою генерацією. Ідеальна крива відображає максимально сприятливий випадок без урахування фактичної погоди. Погодозалежна крива краще відповідає умовам реальної експлуатації, оскільки враховує хмарність, опади та інші погодні обмеження.

Такий графік є важливим для обґрунтування моделі. Якщо б система використовувала лише ідеальну генерацію, EMS працював би з надто оптимістичними даними. У такому випадку алгоритм міг би частіше очікувати надлишок сонячної енергії, хоча в реальних умовах хмарність або опади зменшили б доступну потужність. Погодозалежна генерація створює більш складні й корисні сценарії для перевірки роботи EMS.

Другим корисним графіком є довший історичний профіль генерації. Він показує не лише форму одного дня, а й повторюваність добових циклів, зміну тривалості світлового дня, вплив погоди та сезонну різницю між періодами.

На рисунку 4.3 можна простежити, що сонячна генерація має виражену добову структуру, але її амплітуда змінюється від дня до дня. У сонячні дні пікова потужність наближається до максимальної встановленої потужності масиву, а в похмурі або дощові дні генерація може залишатися низькою протягом усього світлового періоду.

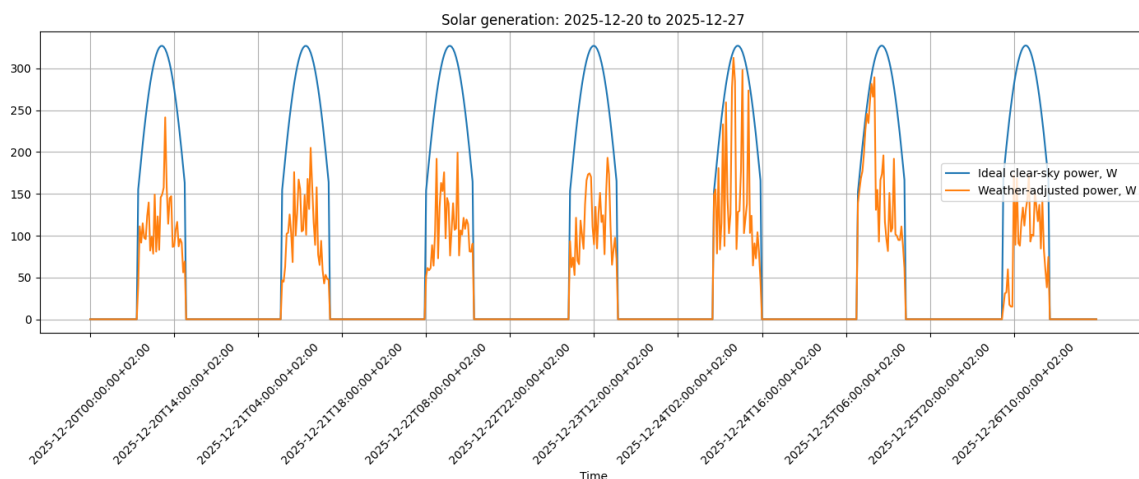


Рисунок 4.3 – Приклад історичного графіка сонячної генерації

Окремо доцільно показати графік добової згенерованої енергії за вибраний період (рис 4.4). На відміну від миттєвої потужності, цей показник характеризує сумарний енергетичний результат за добу. Він дозволяє оцінити, скільки енергії сонячна станція фактично могла б надати системі протягом дня.

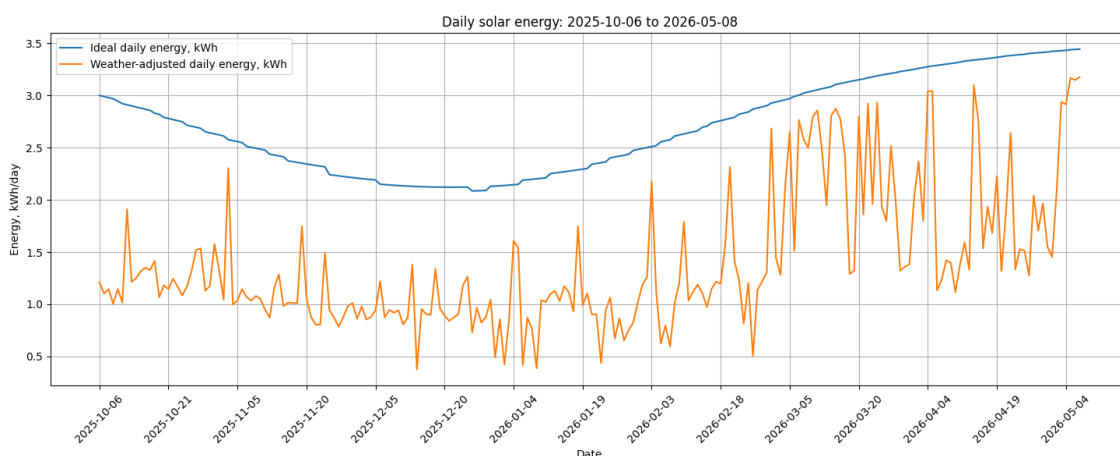


Рисунок 4.4 – Приклад добової згенерованої енергії за історичний період

На рисунку 4.4 видно, що добова генерація може суттєво змінюватися навіть у межах одного сезону. Для EMS це важливо, оскільки рішення про розподіл енергії в системі батареї залежить не лише від поточної миттєвої потужності, а й від загального енергетичного балансу та залежного від нього стану батареї за попередній період.

4.2 Програмна реалізація симуляції наявності електромережі за умов обстрілів

Модуль симуляції наявності електромережі призначений для формування даних доступності централізованого електропостачання для лабораторного стенду SmartEnergy Lab. Його задача полягає не у точному моделюванні всієї енергосистеми України, але у створенні правдоподібного зовнішнього контексту, у якому має працювати система з сонячною генерацією, акумуляторною батареєю, навантаженням та EMS.

Необхідність такого модуля пов'язана з тим, що в реальних українських умовах централізована електромережа не може вважатися гарантовано стабільним джерелом живлення. Через обстріли енергетичної інфраструктури, дефіцит генерації, пошкодження елементів передачі та подальше відновлення система може перебувати в різних станах: від повністю стабільної роботи до тривалих аварійних або планових відключень. Для задачі автоматизації режимів заряджання це має принципове значення, оскільки поведінка EMS повинна змінюватися залежно від поточної та попередньої доступності мережі.

У спрощеному варіанті стан електромережі можна було б задати одним бінарним параметром та випадковими відключеннями. Такий підхід є занадто обмеженим для цієї роботи оскільки він не відображає реалії в яких довелось би функціонувати модулю в реальних умовах де наявність мережі є нестабільною й залежить від обстрілів та можливості покрити потреби користувачів без каскадного руйнування системи. Тому в реалізованому модулі формується розширений набір параметрів: умовна цілісність генерації, умовна цілісність доставки електроенергії, ефективна цілісність мережі, дефіцит, рівень відключень, добова тривалість відключень, напруга та поточні або наступні вікна недоступності.

Таким чином, модуль електромережі створює не просто випадкові проміжки “світло є / світла немає”, а сценарну модель роботи енергосистеми в умовах пошкоджень. Отримані дані надалі використовуються інтегрованою симуляцією та

EMS для прийняття рішень щодо заряджання акумуляторної батареї, збереження резерву, використання мережі та роботи системи під час відключень.

4.2.1 Вихідні припущення та параметри моделі

Основним припущенням моделі є розділення стану електромережі на дві складові: стан генерації та стан доставки електроенергії. Стан генерації описує умовну здатність енергосистеми виробляти достатню кількість електроенергії. Стан доставки описує здатність передати цю електроенергію до споживача через підстанції, лінії та інші елементи мережевої інфраструктури.

Таке розділення важливе, оскільки в реальній системі дефіцит може виникати з різних причин. Електроенергія може бути вироблена, але не доставлена до споживача через пошкодження мережі. Або навпаки, мережа доставки може бути умовно працездатною, але генерації недостатньо для покриття попиту. У моделі ефективний стан електромережі визначається слабшою з двох складових. Якщо генерація має умовну цілісність 120 % - тобто є резерв для покриття всього поточного навантаження, а доставка лише 80 %, то ефективним станом системи буде 80 %, оскільки саме доставка стає обмежуючим фактором.

Початкові значення стану генерації та доставки задано на рівні 130 %. Це означає, що система має умовний запас міцності й не переходить до дефіциту після незначного пошкодження. Межа нормальної роботи проходить через 100 %. Якщо ефективна цілісність мережі дорівнює або перевищує 100 %, дефіциту немає. Якщо вона падає нижче 100 %, різниця між 100 % і поточним ефективним станом інтерпретується як дефіцит.

Як видно з таблиці 4.3, модель має як числові параметри конфігурації, так і шкалу якісного стану мережі. Параметри 130 %, 150 % та 0 % задають допустимий діапазон зміни умовної цілісності системи. Черга відключень 3.1 використовується для зміщення вікон недоступності в межах доби, а seed-значення забезпечує повторюваність симуляції.

Таблиця 4.3 – Базові параметри та рівні стану електромережі

Параметр / рівень	Значення	Пояснення
base_generation_health_percent	130 %	Початковий стан цілісності генерації
base_delivery_health_percent	130 %	Початковий стан цілісності доставки
regeneration_cap_percent	150 %	Верхня межа цілісності системи після пошкоджень.
minimum_health_percent	0 %	Нижня межа стану генерації та доставки.
outage_queue	3.1	Умовна черга відключень
outage_schedule_seed	20260513	Seed-значення для детермінованого формування сценаріїв і графіків.
local_timezone	Europe/Kyiv	Часовий пояс для локального часу симуляції.
Крок точок доступності	30 хв	Інтервал, з яким формується часовий ряд стану електромережі.
stable	дефіцит = 0 %	Стабільний режим, відключення не формуються.
strained	дефіцит = 0–10 %	Напружений режим з малим дефіцитом.
partial_outage	дефіцит = 10–40 %	Часткові відключення через помітний дефіцит.
severe_outage	дефіцит = 40–80 %	Важкий режим з тривалими відключеннями.
blackout	дефіцит = > 80 %	Критичний стан, більша частина доби може бути без електроенергії.

Рівні стану електромережі визначаються за величиною дефіциту. За відсутності дефіциту система вважається стабільною. Невеликий дефіцит переводить її в напружений режим. За більшого дефіциту формуються часткові або важкі відключення. Якщо дефіцит перевищує 80 %, модель переходить у режим blackout, тобто критичного стану з дуже низькою доступністю зовнішнього живлення.

4.2.2 Формування добового сценарію обстрілу

Першим етапом моделювання є формування добового сценарію обстрілу. У моделі не використовується повністю випадковий генератор подій, оскільки такий підхід не відображав би реальну логіку атак, які впливають на українську енергосистему. Натомість добовий сценарій формується як спрощене відтворення типових патернів російських обстрілів України на основі відкритих повідомлень, статистики застосування різних типів засобів ураження та особистих спостережень автора за регулярністю таких подій.

У відкритих джерелах неодноразово описувалися комбіновані удари, у яких одночасно застосовувалися ударні БПЛА, крилаті ракети, балістичні ракети та інші засоби ураження. Наприклад, у повідомленнях про великі атаки по енергетичній інфраструктурі фіксувалися десятки ракет і десятки або сотні БПЛА, а також значні наслідки для електропостачання [21], [22]. Через це в моделі окремо виділено сценарії рутинної атаки БПЛА, щільної атаки БПЛА та комбінованого удару.

Стан `quiet` відповідає спокійному дню, коли значного впливу на енергосистему внаслідок обстрілів не очікується, тобто рідкісному дню без суттєвих повітряних атак. Стан `routine_uav` описує рутинну атаку безпілотниками. Стан `dense_uav` використовується для щільної атаки БПЛА з підвищеною кількістю засобів ураження. Стан `combined` описує найважчий сценарій, у якому одночасно можуть бути присутні БПЛА, крилаті ракети та балістичні ракети. Стан `cooldown` використовується для періоду після значної атаки, коли ймовірність повторного масованого сценарію знижується.

Вибір стану доби не є повністю випадковим. Модель враховує попередній стан та сезонну інтенсивність. Попередній стан потрібен для того, щоб події не були ізольованими одна від одної. Наприклад, після комбінованої атаки підвищується ймовірність переходу до `cooldown`, тобто до періоду відносного зниження інтенсивності. Після щільної атаки БПЛА може зростати ймовірність

важчого сценарію, оскільки в реальних умовах окремі хвилі атак часто є частиною ширшої кампанії тиску.

Сезонна інтенсивність використовується для наближеного відтворення того, що атаки на енергетичну інфраструктуру мають більший ефект у холодний період року. У зимові місяці коефіцієнт інтенсивності дорівнює 1.0, у літні – 0.0, а навесні та восени змінюється поступово. Таке зміщення ймовірностей відображає припущення, що зимові атаки по енергетиці мають більшу стратегічну цінність для противника, оскільки електро- й теплопостачання стає критично важливим для населення, промисловості та роботи міської інфраструктури [23].

Після вибору стану доби модель формує загальнонаціональну кількість засобів ураження трьох типів: БПЛА, крилатих ракет і балістичних ракет. Для кожного сценарію задано свої середні значення та діапазони. У стані *quiet* кількість БПЛА низька, а крилаті та балістичні ракети відсутні. У стані *routine_uav* основним засобом ураження є БПЛА, а ракети можуть з'являтися лише з малою ймовірністю. У стані *dense_uav* кількість БПЛА суттєво зростає, а ракетні засоби можуть доповнювати атаку. У стані *combined* формується найважчий сценарій, у якому присутні всі три типи засобів ураження.

Оскільки лабораторний стенд SmartEnergy Lab прив'язаний до Київського регіону, загальнонаціональна атака не повинна автоматично створювати локальне пошкодження в моделі. Для цього введено режим фокусу атаки на Київ: *no_kyiv*, *spillover*, *secondary* та *primary*. Якщо вибрано *no_kyiv*, атака не впливає на локальну модель. Якщо вибрано *spillover*, до локальної моделі потрапляє лише мала частка загальної атаки. Режим *secondary* відповідає ситуації, коли Київський регіон є другорядним напрямом атаки, а *primary* – коли він є одним з основних напрямів.

Таблиця 4.4 показує, що модель не використовує один універсальний сценарій для всіх днів. Добова симуляція складається з кількох послідовних рішень: вибору стану атаки, формування кількості засобів ураження, визначення локального фокусу та переходу від загальнонаціональної кількості до локальної кількості подій.

Таблиця 4.4 – Параметри формування добового сценарію атаки

Група параметрів	Варіанти / діапазони	Призначення
Стан атаки	quiet	Спокійний або малопомітний день без суттєвого впливу на енергосистему.
Стан атаки	routine_uav	Рутинна атака БПЛА з переважно малою або середньою кількістю локальних ризиків.
Стан атаки	dense_uav	Щільна атака БПЛА з підвищеною кількістю засобів ураження.
Стан атаки	combined	Комбінована атака з БПЛА, крилатими та балістичними ракетами.
Стан атаки	cooldown	Період після значної атаки з нижчою ймовірністю повторної комбінованої події.
Сезонна інтенсивність	зима = 1.0; літо = 0.0; березень, вересень, жовтень – перехідні значення	Підвищує ймовірність інтенсивних сценаріїв у холодний період.
quiet: UAV	середнє 10; діапазон 0–50	Фонові або поодинокі події без крилатих і балістичних ракет.
routine_uav: UAV	середнє 120 + 80·winter; діапазон 40–300	Типовий сценарій регулярної атаки БПЛА.
routine_uav: cruise	середнє 3; діапазон 1–8; ймовірність появи 10 %	Рідкісне додавання крилатих ракет до рутинної атаки.
routine_uav: ballistic	середнє 2; діапазон 1–5; ймовірність появи 5 %	Рідкісне додавання балістичних ракет до рутинної атаки.
dense_uav: UAV	середнє 300 + 120·winter; діапазон 180–700	Масована атака БПЛА.
dense_uav: cruise	середнє 5; діапазон 1–15; ймовірність появи 20 %	Можливе посилення атаки крилатими ракетами.
dense_uav: ballistic	середнє 3; діапазон 1–8; ймовірність появи 10 %	Можливе посилення атаки балістичними ракетами.

Продовження таблиці 4.4

combined: UAV	середнє 430 + 150·winter; діапазон 250–950	Велика кількість БПЛА в комбінованій атаці.
combined: cruise	середнє 20 + 15·winter; діапазон 5–90	Основна ракетна складова комбінованої атаки.
combined: ballistic	середнє 8 + 8·winter; діапазон 2–40	Найважча для нейтралізації складова комбінованої атаки.
cooldown: UAV	середнє 70 + 60·winter; діапазон 0–220	Залишкова активність після значної атаки.
Фокус на Київ	no_kyiv	Загальнонаціональна подія не впливає на локальну зону стенду.
Фокус на Київ	spillover	Побічний вплив; локальна частка атаки приблизно 0.02–0.12.
Фокус на Київ	secondary	Київський регіон є другорядною ціллю; локальна частка приблизно 0.12–0.45.
Фокус на Київ	primary	Київський регіон є основною ціллю; локальна частка приблизно 0.55–1.00.
Імовірності фокусу для combined	no_kyiv 0.15; spillover 0.15; secondary 0.30; primary 0.40	Комбінована атака частіше має значний локальний вплив.
Імовірності фокусу для dense_uav	no_kyiv 0.30; spillover 0.25; secondary 0.30; primary 0.15	Щільна атака БПЛА має середній шанс локального впливу.
Імовірності фокусу для quiet/cooldown	no_kyiv 0.65; spillover 0.22; secondary 0.11; primary 0.02	У спокійні або післяатакові дні переважає відсутність локального впливу.
Імовірності фокусу для routine_uav	no_kyiv 0.45; spillover 0.30; secondary 0.20; primary 0.05	Рутинна атака найчастіше не є основною локальною подією для стенду.

Такий підхід дозволяє наближено відтворити поточні сценарії обстрілів, які застосовуються проти України: рутинні атаки БПЛА, щільні хвилі безпілотників, комбіновані удари з використанням різних засобів ураження та періоди зниження інтенсивності після великих атак. При цьому модель не претендує на прогнозування реальних обстрілів. Її задача – створити правдоподібне симуляційне середовище для перевірки роботи EMS в умовах нестабільної електромережі.

4.2.3 Моделювання пошкоджень енергетичної інфраструктури

Після формування локальної кількості засобів ураження система переходить до моделювання пошкоджень енергетичної інфраструктури. На цьому етапі важливо, що не кожна повітряна ціль автоматично створює пошкодження електромережі. Реальні обстріли можуть бути спрямовані на різні типи об'єктів: енергетику, промисловість, військову інфраструктуру, логістику або цивільні об'єкти. Тому в моделі спочатку визначається, чи стосується конкретна подія саме енергетичної інфраструктури.

Імовірність вибору енергетичної цілі також залежить від сезону. У літній період вона нижча, оскільки наслідки пошкодження енергосистеми, хоча й залишаються суттєвими, не створюють такого ж рівня гуманітарного тиску, як у морозний період. У зимові місяці ця ймовірність зростає, оскільки пошкодження електро- й теплопостачання напряду впливає на стійкість міст, роботу підприємств, побут населення та здатність системи витримувати пікові навантаження [23].

Другим фільтром є умовна ефективність протиповітряної оборони. Значення ефективності ППО в моделі не є довільними: вони задані з урахуванням відкритих повідомлень про роботу української ППО проти різних типів цілей. У публічних повідомленнях регулярно видно, що БПЛА типу Shahed загалом перехоплюються частіше, ніж складніші ракетні цілі, хоча їхня ефективність також змінюється залежно від модернізацій дронів, погоди, маршруту польоту та насичення атаки. Для ракетних цілей ситуація складніша: крилаті ракети можуть ефективно перехоплюватися спеціалізованими системами ППО, тоді як балістичні ракети потребують значно складніших систем і мають нижчу середню ймовірність нейтралізації [24], [25].

З огляду на це для БПЛА в моделі задано найвищу ймовірність нейтралізації. Для крилатих ракет ймовірність нижча, оскільки їх перехоплення потребує спеціалізованих зенітних ракетних систем і залежить від наявності боєкомплекту. Для балістичних ракет у моделі задано найнижчу ймовірність нейтралізації,

оскільки такі цілі складніші для перехоплення, особливо за умови маневрування або використання модернізованих ракет.

Подія пошкодження створюється тільки в тому випадку, якщо ціль пов'язана з енергетикою і засіб ураження не був нейтралізований. Після цього модель визначає клас пошкодження та його величину. Для БПЛА переважають легкі або обмежені пошкодження, оскільки вони частіше моделюють ураження локальних елементів доставки електроенергії. Для крилатих ракет можливі швидкі, середні та важкі пошкодження. Для балістичних ракет передбачено середні, важкі та структурні пошкодження, які мають найтриваліші наслідки.

Окремо визначається, чи пошкодження впливає на генерацію або доставку електроенергії. Це розділення було введено з огляду на те, що атаки можуть мати різну природу наслідків. Ураження генерувальних потужностей зменшує здатність системи виробляти електроенергію. Ураження підстанцій, трансформаторів або ліній впливає на здатність доставити електроенергію до споживача навіть тоді, коли виробництво формально можливе.

У моделі БПЛА частіше впливають на доставку електроенергії. Це відповідає припущенню, що такі атаки можуть уражати локальні підстанції, трансформаторне обладнання або об'єкти розподільчої інфраструктури. Крилаті ракети мають більш збалансований розподіл між генерацією та доставкою. Балістичні ракети частіше відносяться до важчих подій і можуть сильніше впливати на генерацію або великі об'єкти енергетичної інфраструктури.

Таблиця 4.5 об'єднує основні приховані параметри цього етапу: імовірність вибору енергетичної цілі, ефективність ППО, класи пошкоджень, розподіл шкоди між генерацією та доставкою, а також час відновлення. Завдяки цьому можна показати внутрішню логіку модуля без надмірного розширення текстового опису.

Таблиця 4.5 – Параметри моделювання пошкоджень енергетичної інфраструктури

Група параметрів	Значення / діапазон	Пояснення
Імовірність енергетичної цілі влітку	середнє 0.20; діапазон 0.00–0.40	У теплий період модель задає меншу ймовірність атаки саме на енергетичну інфраструктуру.
Імовірність енергетичної цілі взимку	середнє 0.70; діапазон 0.40–1.00	У холодний період ймовірність енергетичної цілі зростає.
Ефективність ППО для UAV	середнє 0.90; діапазон 0.78–0.95	Висока ймовірність нейтралізації БПЛА.
Ефективність ППО для крилатих ракет.	середнє 0.78; діапазон 0.60–0.92	Середньо-висока ймовірність нейтралізації крилатих ракет.
Ефективність ППО для балістичних цілей	середнє 0.35; діапазон 0.20–0.45	Найнижча ймовірність нейтралізації серед типів засобів ураження.
UAV: fast	імовірність 80 %; пошкодження 0.05–0.80 %, середнє 0.30 %	Невелике швидке пошкодження, характерне для більшості успішних уражень БПЛА.
UAV: medium_low	імовірність 20 %; пошкодження 0.50–2.50 %, середнє 1.20 %	Сильніше, але все ще обмежене пошкодження від БПЛА.
Cruise: fast	імовірність 20 %; пошкодження 0.30–1.80 %, середнє 0.90 %	Порівняно легкий наслідок від крилатої ракети.
Cruise: medium	імовірність 70 %; пошкодження 1.20–6.50 %, середнє 3.20 %	Типове середнє пошкодження від крилатої ракети.
Cruise: heavy	імовірність 10 %; пошкодження 4.00–12.00 %, середнє 7.00 %	Важке пошкодження від крилатої ракети.
Ballistic: medium	імовірність 55 %; пошкодження 2.00–9.00 %, середнє 5.00 %	Середнє пошкодження від балістичної ракети.
Ballistic: heavy	імовірність 35 %; пошкодження 5.00–17.00 %, середнє 10.00 %	Важке пошкодження від балістичної ракети.
Ballistic: structural	імовірність 10 %; пошкодження 8.00–28.00 %, середнє 16.00 %	Структурне довготривале пошкодження.

Продовження таблиці 4.5

Ballistic: structural	імовірність 10 %; пошкодження 8.00– 28.00 %, середнє 16.00 %	Структурне довготривале пошкодження.
Розподіл пошкоджень UAV	70 % у доставку; 30 % у генерацію	БПЛА частіше моделюються як загроза локальним елементам доставки.
Розподіл пошкоджень cruise	50 % у доставку; 50 % у генерацію	Крилаті ракети мають збалансований вплив на генерацію і доставку.
Розподіл пошкоджень ballistic	35 % у доставку; 65 % у генерацію	Балістичні ракети частіше моделюються як причина важчих генераційних пошкоджень.
Відновлення fast	2–7 днів	Швидке відновлення після легкого пошкодження.
Відновлення medium	7–45 днів	Середній ремонтний цикл.
Відновлення heavy	30–180 днів	Тривале відновлення після важкого пошкодження.
Відновлення structural	180–540 днів	Дуже довготривале відновлення після структурного пошкодження.
Швидке первинне відновлення fast	до 40 % протягом першої доби	Частина наслідків швидко компенсується аварійними роботами.
Швидке первинне відновлення medium	до 10 % протягом першої доби	Для середніх пошкоджень первинна компенсація менша.
Швидке первинне відновлення heavy	до 2 % протягом першої доби	Важкі пошкодження майже не компенсуються швидко.
Швидке первинне відновлення structural	0 % протягом першої доби	Структурні пошкодження не мають швидкого відновлення в моделі.

Після визначення окремих подій пошкодження модель застосовує обмеження сумарної шкоди атаки. Це потрібно для того, щоб один масований сценарій не створював необмежено велике пошкодження лише через велику кількість засобів ураження. Такий підхід відповідає логіці насичення: зі зростанням кількості влучань сумарна шкода зростає, але не нескінченно лінійно. Інакше одна велика

атака могла б повністю обнулити модель електромережі та зробити подальшу симуляцію непридатною.

Відновлення після пошкоджень також моделюється поступово. Для кожного класу пошкодження задається діапазон часу відновлення. Легкі пошкодження можуть бути усунені протягом кількох днів, середні – протягом тижнів, важкі – протягом місяців, а структурні можуть впливати на систему значно довше. У перший день частина наслідків може компенсуватися аварійними роботами, після чого залишкова шкода зменшується повільніше протягом усього періоду відновлення.

Отже, у моделі пошкодження електромережі не виникає як довільне число. Воно є результатом послідовності наближених до реальних умов рішень: сценарій атаки, тип засобу ураження, локальний фокус на Київ, імовірність енергетичної цілі, результат ППО, клас пошкодження, розподіл між генерацією та доставкою і тривалість відновлення. Саме ця послідовність дозволяє отримати графіки відключень, які мають внутрішню причину, а не є випадковим набором проміжків недоступності.

4.2.4 Перехід від пошкоджень до точок доступності електромережі

Після формування подій пошкодження система переходить від опису атак до власне стану електромережі. Для кожного моменту часу визначаються активні пошкодження, тобто ті події, які вже відбулися, але ще не були повністю відновлені. Для кожної активної події розраховується залишкова частка пошкодження, після чого окремо підсумовується вплив на генерацію та доставку.

Стан генерації розраховується як початковий запас генерації мінус активні пошкодження генерації. Аналогічно стан доставки розраховується як початковий запас доставки мінус активні пошкодження доставки. Обидва значення обмежуються мінімальною та максимальною межами, заданими в конфігурації.

Після цього ефективний стан системи визначається як мінімум між станом генерації та станом доставки.

Як видно з рисунка 4.5, модуль спочатку формує сценарні пошкодження, після чого оцінює активний стан генерації та доставки електроенергії. Від цих двох значень залежить ефективна цілісність мережі. Саме вона визначає, чи виникає дефіцит, який рівень відключень має система та скільки годин на добу буде недоступне зовнішнє живлення.

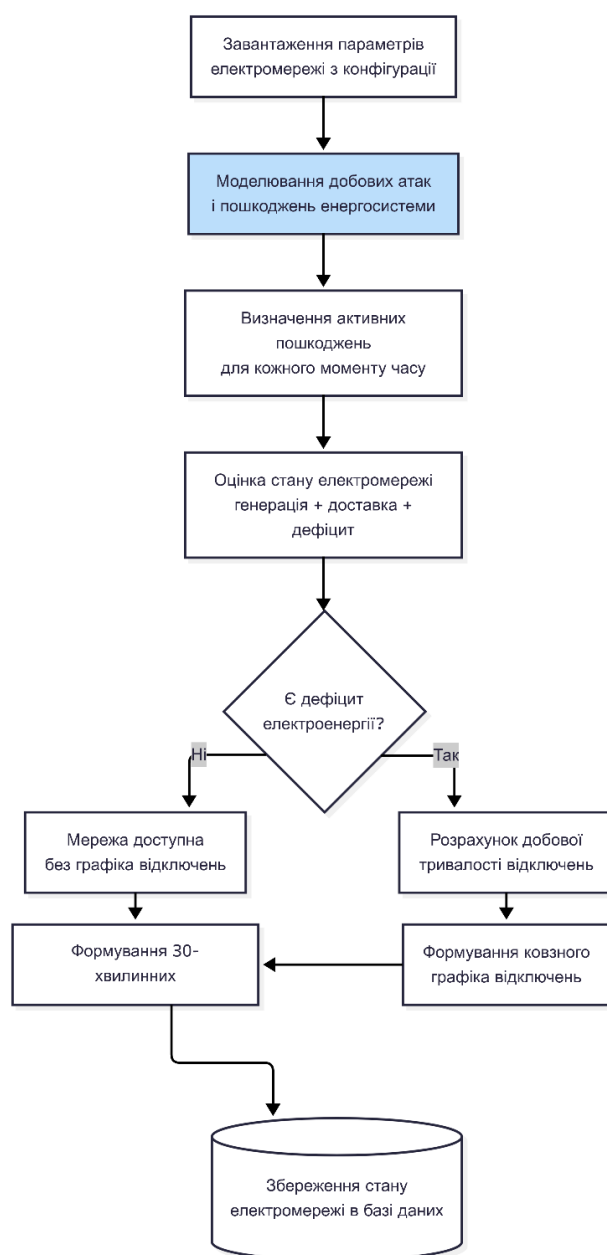


Рисунок 4.5 – Блок-схема алгоритму симуляції стану електромережі

Дефіцит визначається як нестача ефективного стану до 100 %. Якщо ефективний стан становить 100 % або більше, дефіцит дорівнює нулю. Якщо ефективний стан становить 75 %, дефіцит дорівнює 25 %. Далі цей дефіцит переводиться в добову кількість годин відключень. Для цього використовується пропорція: 24 години множаться на відсоток дефіциту. Отримане значення округлюється до половини години.

Після розрахунку добової тривалості відключень система формує конкретні вікна недоступності. Добова кількість годин розбивається на окремі блоки. Для малих значень використовуються коротші блоки, для важких дефіцитів – довші. Черга відключень зміщує початок вікон у межах доби, а невелике відхилення початку та кінця робить графік менш механічним.

На основі сформованих вікон система генерує часові точки доступності з кроком 30 хвилин. Для кожної точки визначається, чи триває зараз відключення, чи доступна мережа, яка умовна напруга має бути в цей момент, коли почалося або закінчиться поточне вікно відключення та коли очікується наступне вікно. Саме ці точки надалі використовуються інтегрованою симуляцією.

Рисунок 4.6 деталізує внутрішній блок моделювання атак. На ньому показано, що пошкодження не створюється безпосередньо з факту атаки. Спочатку визначається сезонна інтенсивність і стан доби, потім формується загальна кількість засобів ураження, оцінюється фокус на Київський регіон, перевіряється енергетична спрямованість цілі та результат роботи ППО.

Лише після проходження цих етапів формується подія пошкодження. Для неї визначаються тип засобу ураження, клас пошкодження, величина шкоди, розподіл між генерацією та доставкою, а також тривалість відновлення. Завдяки цьому модель створює не випадковий графік відключень, а послідовність причинно пов'язаних подій: атака → пошкодження → дефіцит → графік відключень → відновлення.

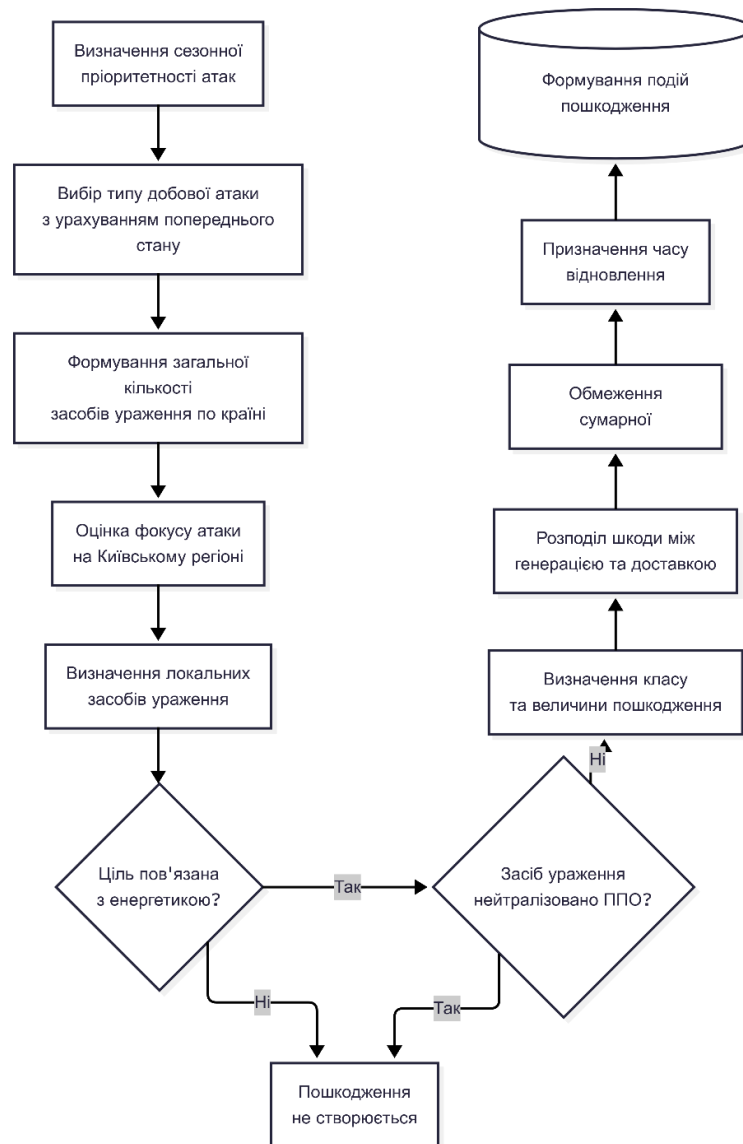


Рисунок 4.6 – Блок-схема алгоритму моделювання атак і пошкоджень енергосистеми

Окремо моделюється умовна напруга мережі. Якщо в поточній часовій точці триває відключення, напруга встановлюється рівною нулю. Якщо мережа доступна, але система має дефіцит, напруга може коливатися або просідати. У стабільному режимі вона залишається близькою до 230 В. Цей параметр не є повною електротехнічною моделлю якості напруги, але дає додатковий показник стану зовнішнього живлення.

4.2.5 Тестування роботи модуля

Для перевірки роботи модуля використовуються графіки, побудовані на основі збережених симуляційних даних. Вони потрібні для того, щоб оцінити не окрему точку розрахунку, а поведінку моделі в часі. Оскільки модуль електромережі має інерцію через пошкодження та відновлення, перевірка лише одного дня не є достатньою.

Першим корисним графіком є зміна добової кількості годин відключень за історичний період. Він дозволяє побачити, як окремі сценарії атак впливають на тривалість відключень, чи виникають пікові дні та чи відбувається поступове повернення до стабільнішого режиму після пошкоджень.

На рисунку 4.7 має бути видно, що відключення не виникають як повністю незалежний шум. Після сильніших подій кількість годин без електроенергії зростає, а після завершення активного періоду поступово зменшується. Це відповідає закладеній логіці відновлення, де пошкодження мають залишковий вплив протягом певного часу.

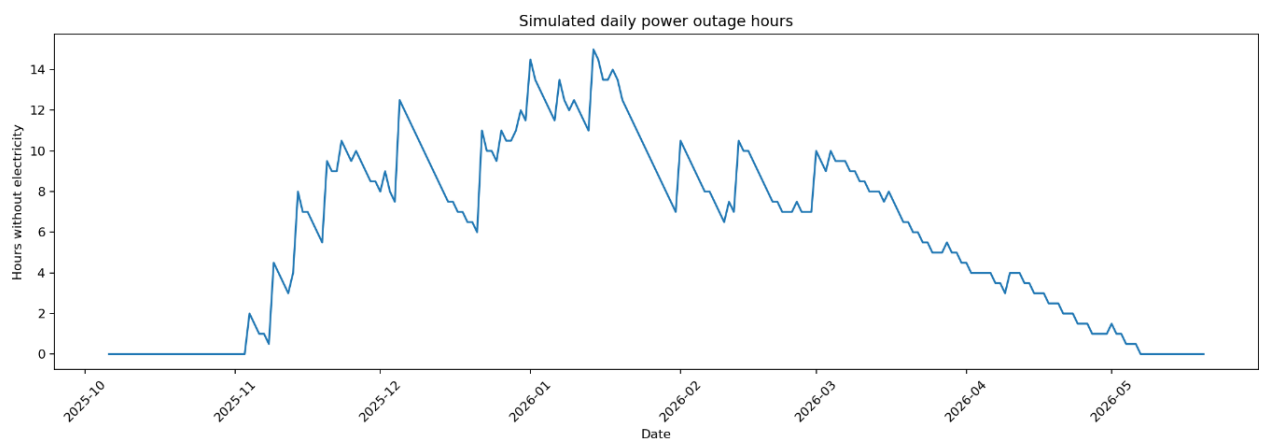


Рисунок 4.7 – Приклад змодельованої кількості годин відключень за добу

Такий графік важливий для перевірки EMS, оскільки алгоритм заряджання має враховувати не лише поточний стан мережі, а й повторюваність та тривалість відключень. Якщо модель показує кілька днів поспіль з великою кількістю годин без живлення, EMS повинен поводитися обережніше та підтримувати більший резерв акумуляторної батареї.

Другим показником для перевірки є ефективна цілісність електромережі. Вона показує не самі вікна відключень, а внутрішній стан моделі, з якого ці відключення виникають. Це дозволяє оцінити, чи правильно пошкодження та відновлення перетворюються в дефіцит.

На рисунку 4.8 можна перевірити, що цілісність мережі не перемикається різко між нулем і ста відсотками. Вона змінюється поступово, оскільки активні пошкодження накопичуються, частково компенсуються та з часом відновлюються.

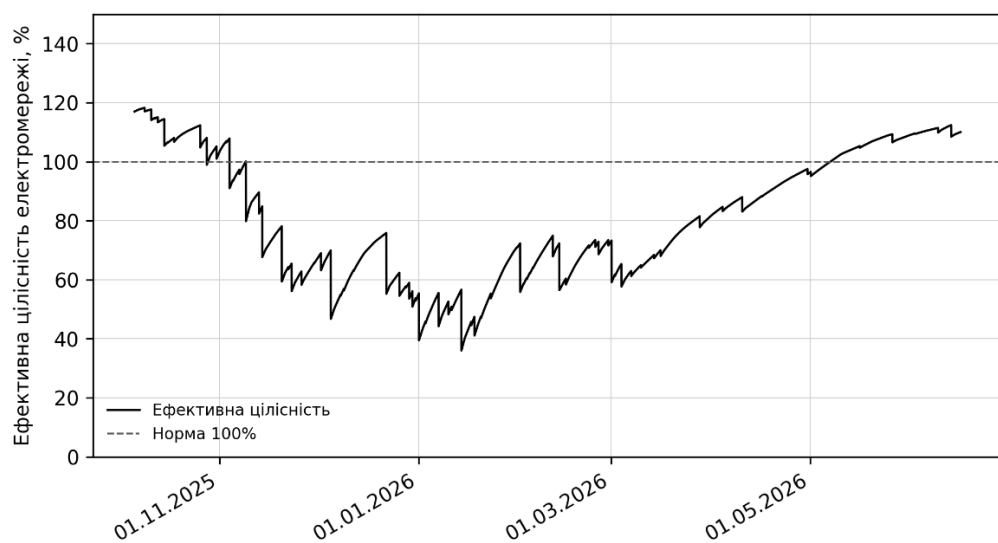


Рисунок 4.8 – Приклад зміни ефективної цілісності електромережі за історичний період

Отже, модуль симуляції наявності електромережі формує для EMS ризиковий контекст, у якому враховуються не лише поточний факт наявності або відсутності електроживлення, а й ефективна цілісність мережі, дефіцит, тривалість відключень, поточне та наступне вікно недоступності. Завдяки цьому алгоритм автоматизації режимів заряджання може працювати не зі спрощеним припущенням про стабільну мережу, а з наближеними до реальних умов сценаріями: стабільною роботою, підготовкою до можливого відключення, активним відключенням, післяаварійним відновленням та необхідністю збереження резерву акумуляторної батареї.

4.3 Програмна реалізація симуляції акумуляторної батареї

Модуль симуляції акумуляторної батареї призначений для надання даних про стан батареї та буферу енергії в системі. У цій роботі батарея розглядається як SoC/SoH модель та буфер енергії з обмеженою корисною ємністю

Такий рівень деталізації є достатнім для задачі автоматизації режимів заряджання. EMS не потребує повної фізико-хімічної моделі акумулятора, але повинна знати, скільки енергії доступно в батареї, чи можна її заряджати/розряджати, чи не досягнуто межі SoC.

У модулі передбачено підтримку кількох типів акумуляторних батарей: свинцево-кислотний, LiFePO₄ та Li-Ion. Для кожного типу задано власний профіль, який визначає допустиму глибину розряду, максимальний C-rate заряду, орієнтовний ресурс до зниження SoH до 80 % та криву залежності напруги від SoC. У поточній конфігурації стенду використовується свинцево-кислотна батарея.

4.3.1 Конфігурація батареї та параметри моделі

Вхідні параметри акумуляторної батареї задаються в YAML-конфігурації стенду. Параметри поточної батареї в стенді наведені в таблиці 4.6.

Параметри з таблиці 4.6 визначають межі роботи модуля. Номінальна напруга та ємність задають загальну енергетичну основу батареї. Параметри DoD обмежують доступну для симуляції частину цієї енергії. Максимальний C-rate заряду визначає верхню межу зарядної потужності, а ефективність заряду й розряду враховує втрати під час накопичення та віддачі енергії.

Таблиця 4.6 – Основні параметри моделі акумуляторної батареї

Параметр	Значення	Пояснення
Тип батареї	lead_acid	Тип акумулятора, заданий у конфігурації стенду
Номінальна напруга	12 V	Базова напруга акумуляторної батареї
Ємність	200 A·год	Конфігураційна ємність акумулятора

Продовження таблиці 4.5

Ємність	200 А·год	Конфігураційна ємність акумулятора
Номінальна енергія	2400 Вт·год	Розраховується як $12 \text{ В} \times 200 \text{ А} \cdot \text{год}$
Корисна частка DoD	0,50	Для свинцево-кислотної батареї використовується не вся номінальна ємність
Робочий коефіцієнт DoD	0,80	Додаткове обмеження експлуатаційної глибини розряду
Ефективна корисна ємність	960 Вт·год	$2400 \text{ Вт} \cdot \text{год} \times 0,50 \times 0,80$
Початковий стан заряду	50 %	Початковий стан батареї за відсутності попередніх даних
Максимальний C-rate заряду	0,20C	Обмеження зарядного струму для профілю lead acid
Максимальна потужність заряду	480 Вт	$12 \text{ В} \times 200 \text{ А} \cdot \text{год} \times 0,20$
Ефективність заряду	0,85	Частина енергії заряду, що зберігається в батареї
Ефективність розряду	0,85	Частина енергії батареї, доступна навантаженню
Ресурс до 80 % SoH	500 циклів	Опорне значення для розрахунку деградації

Початковий стан заряду батареї за відсутності попередніх даних приймається рівним 50 % від доступної корисної ємності. Це дозволяє запускати симуляцію з нейтрального стану, у якому батарея не є ні повністю зарядженою, ні повністю розрядженою. Надалі її стан змінюється залежно від рішень EMS, сонячної генерації, доступності мережі та навантаження.

4.3.2 Алгоритм оновлення стану батареї

Оновлення стану акумуляторної батареї виконується на кожному кроці інтегрованої симуляції. Модуль отримує поточний стан батареї та команду від EMS. Ця команда може містити запит на розряд батареї для живлення навантаження або запит на заряджання від сонячної генерації чи електромережі.

Як видно з рисунка 4.9, на початку роботи модуль завантажує параметри акумуляторної батареї та початковий стан. Якщо попередній стан уже існує в історії або кеші, він може бути використаний як стартовий. Якщо таких даних немає, формується початковий стан з SoC 50 % та SoH 100 %.

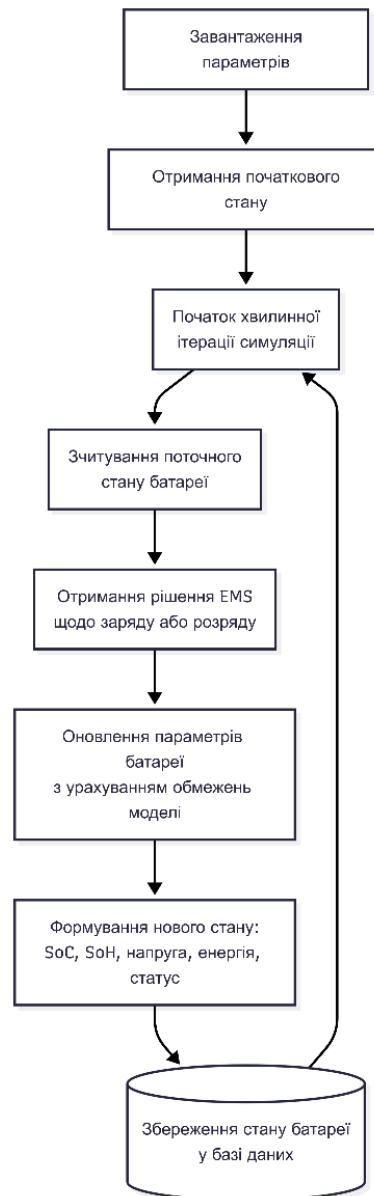


Рисунок 4.9 – Блок-схема алгоритму роботи модуля симуляції акумуляторної батареї

Після виконання заряду або розряду модуль перераховує кількість енергії в батареї, SoC, напругу, еквівалентні цикли за поточну добу та статус. Рівень заряду визначається як відношення поточної енергії до поточної корисної ємності батареї:

$$\text{SoC} = \text{energy_wh} / \text{current_usable_capacity_wh} \times 100 \, \%.$$

У результаті кожного кроку формується новий стан батареї, який містить поточну енергію, SoC, SoH, доступну ємність, напругу, кількість еквівалентних циклів і статус. Цей стан зберігається та використовується на наступному кроці симуляції.

4.3.3 Облік SoH, напруги та статусу батареї

Окрім поточного рівня заряду, модуль враховує спрощену деградацію батареї через показник SoH. У межах моделі SoH не перераховується після кожної хвилини окремо. Натомість протягом доби накопичується кількість енергії, відібраної з батареї, після чого під час фіналізації доби вона переводиться в еквівалентні цикли.

Втрата SoH розраховується відносно опорного ресурсу батареї до 80 % SoH. Для lead_acid профілю це значення становить 500 еквівалентних циклів. Якщо за добу батарея пройшла певну частку циклу, SoH зменшується пропорційно цій частці. Такий підхід не є точною моделлю старіння акумулятора, але дозволяє врахувати поступове зменшення доступної ємності під час довготривалої симуляції.

Напруга батареї також моделюється спрощено. Для кожного типу хімії задано криву залежності напруги від SoC у вигляді набору контрольних точок. Якщо поточний SoC знаходиться між двома контрольними точками, значення напруги визначається інтерполяцією. Це дозволяє отримати правдоподібну зміну напруги без використання складної електрохімічної моделі.

Статус батареї визначається за командою поточного кроку від EMS: заряджати/розряджати, режим очікування. При досягненні межі доступної ємності батарея переходить у стани full або empty.

Таким чином, модуль симуляції акумуляторної батареї виконує роль буферу енергії для EMS. EMS визначає бажаний напрямок енергетичного потоку, але саме модуль батареї перевіряє, чи може цей запит бути виконаний з урахуванням SoC, SoH, доступної ємності, ефективності заряду/розряду та максимальної потужності.

4.4 Програмна реалізація симуляції споживання електроенергії

Модуль симуляції споживання електроенергії призначений для формування змінного у часі профілю навантаження лабораторного стенду SmartEnergy Lab. Покривання споживання та реакція на його зміни є однією з головних задач EMS, оскільки саме поточний запит навантаження визначає, скільки енергії потрібно подати від сонячної генерації, електромережі або акумуляторної батареї. Тому споживання в системі не може бути задане одним сталим значенням потужності.

Для перевірки повноти логіки EMS та автоматизації режимів заряджання було прийнято рішення створити стохастичну модель споживання. Вона формує правдоподібний профіль навантаження лабораторії, який змінюється залежно від часу доби, присутності викладачів і студентів, типу активності, погодних умов, стану електромережі та рівня заряду акумуляторної батареї. Такий підхід дозволяє перевіряти EMS у більш реалістичних умовах, ніж модель з фіксованим навантаженням.

Модуль не призначений для точного прогнозування реального споживання конкретної лабораторії у конкретний день. Його задача полягає у створенні достатньо різноманітного та логічно обґрунтованого набору вхідних даних для інтегрованої симуляції. Завдяки цьому EMS отримує змінний запит споживання, який може містити базове нічне навантаження, робочі періоди, заняття, випадкові відвідування, освітлення, заряджання персональних пристроїв і короткі високопотужні піки.

4.4.1 Вхідні припущення та контекст симуляції

Вхідними даними для модуля споживання є часовий контекст, параметри лабораторії та поточний стан інших частин системи. На кожному кроці симуляції модуль отримує момент часу, часовий пояс, стан електромережі, факт доступності мережі, рівень заряду акумуляторної батареї, погодний стан і параметри

освітленості. Також у модулі передбачені допоміжні параметри для тестування, зокрема примусове задання кількості присутніх викладачів або студентів, увімкнення чайника, високопотужних пристроїв чи режиму роботи лише з робочими місцями.

Модуль споживання не працює ізольовано. У межах інтегрованої симуляції він отримує стан електромережі з відповідного модуля, SoC акумуляторної батареї з модуля батареї та погодний стан із модуля сонячної генерації. Отже, споживання реагує не тільки на внутрішній розклад лабораторії, а й на зовнішні умови роботи всієї системи.

Базова конфігурація модуля описує типову лабораторну структуру: кількість викладачів або працівників, кількість робочих місць, кількість паяльних місць, розклад планових занять, можливість випадкових студентських візитів і seed-значення для відтворюваності стохастичних сценаріїв. За однакових вхідних параметрів модель формує однакову послідовність подій, що спрощує тестування та порівняння результатів.

4.4.2 Каталог пристроїв та груп навантаження

Споживання електроенергії в моделі формується на основі каталогу пристроїв. Кожен пристрій має ідентифікатор, групу, кількість, типові рівні потужності та поведінковий тип. Такий підхід дозволяє моделювати не абстрактне навантаження, а набір споживачів, характерних для лабораторного стенду та приміщення, у якому він використовується.

До каталогу входить постійне критичне навантаження, робочі місця, освітлення, паяльне обладнання, лабораторні блоки живлення, персональні зарядні пристрої, ноутбуки студентів і короткочасні високопотужні пристрої. Критичне навантаження створюють ONU, маршрутизатор і мережевий комутатор. Воно присутнє незалежно від кількості людей у лабораторії, оскільки моделює базову мережеву інфраструктуру.

Робочі місця формуються моноблоками та настільними лампами. Паяльні місця включають паяльні станції, лабораторні блоки живлення, локальне освітлення та допоміжні пристрої на зразок клейового пістолета. Окремо враховуються студентські ноутбуки різних профілів: умовно легкі, звичайні офісні та продуктивніші ігрові ноутбуки. Такий поділ потрібен, оскільки потужність різних типів ноутбуків суттєво відрізняється.

Таблиця 4.7 показує, що профіль споживання формується з кількох різних груп. Частина з них створює стабільне або майже стабільне навантаження, наприклад мережеве обладнання. Інші групи залежать від присутності людей або типу активності. Найбільші короточасні піки можуть створювати чайник, будівельний фен, дріль або одночасна робота кількох паяльних місць.

Таблиця 4.7 – Каталог основних груп електричного навантаження лабораторного стенду

Група навантаження	Пристрої	Кількість / масштаб	Типові рівні потужності	Роль у моделі
1	2	3	4	5
Критична мережна інфраструктура	ONU, маршрутизатор, мережевий комутатор	Постійні споживачі	Одиниці або десятки ват	Формує базове навантаження, яке присутнє незалежно від кількості людей
Робочі місця	Моноблоки, настільні лампи	До 5 робочих місць	Від режиму очікування до активної роботи ПК	Моделює роботу викладачів або студентів за комп'ютерами
Освітлення приміщення	Стельові LED-лампи	Залежить від кількості людей і умов освітлення	Додається за темного часу або поганої погоди	Враховує додаткове споживання, пов'язане не з обладнанням, а з умовами роботи

Продовження таблиці 4.7

1	2	3	4	5
Паяльні місця	Паяльна станція, лабораторний БЖ, локальна лампа, клейовий пістолет	До 6 паяльних місць	Від підтримання температури до активного нагріву	Формує навантаження під час електронної практики, пайки або проєктних робіт
Персональні зарядні пристрої	Зарядки телефонів	Залежить від кількості активних людей	Приблизно 5–15 Вт на пристрій	Імітує заряджання персональних пристроїв перед відключеннями та після них
Студентські ноутбуки	Легкі, офісні та продуктивні ноутбуки	За кількістю активних студентів	Приблизно 30–160 Вт залежно від профілю	Створює основну змінну складову під час занять і випадкових візитів
Високопотужні пристрої	Чайник, будівельний фен, ручний дріль	Короткочасні події	Чайник 1200 Вт; фен 1500 Вт; дріль 400–450 Вт	Створює короткі піки споживання та перевіряє реакцію EMS на різке збільшення навантаження

Завдяки такому каталогу модуль може формувати різні сценарії без ручного задання кожного графіка споживання. Якщо в лабораторії активні лише один-два викладачі, навантаження буде відносно невеликим. Якщо одночасно проходить заняття, працюють паяльні місця та заряджаються ноутбуки, загальна потужність зростає. Якщо мережа недоступна і батарея розряджена, високопотужні пристрої обмежуються або не вмикаються.

4.4.3 Формування сценаріїв споживання та реакція на стан системи

Основою модуля є формування хвилинного значення споживання. Для кожної хвилини симуляції визначається поточний контекст, після чого модель послідовно додає активні складові навантаження. Результатом є точка споживання, яка містить часову мітку, сумарну потужність, кількість активних викладачів, кількість активних студентів і набір тегів активних подій.

Сценарії споживання формуються з кількох джерел активності. Першим джерелом є присутність викладачів або працівників лабораторії. Для них моделюється календарний стан: присутність, хвороба, відпустка, польові роботи або регулярна відсутність. Якщо викладач присутній, для нього формується часовий проміжок перебування в лабораторії, а потім поточна активність: робота за комп'ютером, пайка, консультація, коротка відсутність, перерва або неактивна присутність.

Другим джерелом є планові студентські заняття. У моделі вони проводяться у визначені дні тижня, починаються о 14:00 і тривають 90 хвилин. Тип заняття вибирається зі сценаріїв `debugging_session`, `electronics_practice` або `soldering_practice`. Від типу заняття залежить, яка частка студентів використовує ноутбуки, лабораторні блоки живлення, паяльні станції, клейовий пістолет або високопотужне обладнання.

Третім джерелом є випадкові студентські візити. Вони не прив'язані до планового заняття та моделюють консультації, короткі візити, роботу над паяльним проєктом або налагодження програмного забезпечення. Такі візити частіше виникають у будні дні, можуть тривати від однієї до кількох годин і включати від одного до трьох студентів. Це додає до профілю навантаження природну нерівномірність, якої не було б у моделі з одним фіксованим розкладом. Перелік джерел споживання та сценаріїв впливу наведено в таблиці 4.8

Таблиця 4.8 – Параметри сценаріїв активності лабораторії

Джерело сценарію	Параметри формування	Типові варіанти активності	Вплив на навантаження
1	2	3	4
Викладачі / працівники лабораторії	Календар присутності, робочий день, випадкові відсутності	Робота за ПК, пайка, консультація, перерва, коротка відсутність	Додає робочі станції, локальне освітлення, зарядки та частину лабораторного обладнання
Планові студентські заняття	Вівторок і четвер, початок 14:00, тривалість 90 хв	debugging_session, electronics_practice, soldering_practice	Формує передбачуване збільшення споживання у визначені години
Випадкові студентські візити	Імовірність 0,48 у будні та 0,10 у вихідні; 1–3 студенти; 60–480 хв; старт 09:00–17:00	laptop_debugging, soldering_project, consultation, quick_visit	Додає нерівномірність профілю та неповторювані піки навантаження
Освітлення	Активується лише за присутності людей; залежить від темного часу, погоди, сходу й заходу Сонця	Додаткове кімнатне та локальне освітлення	Збільшує споживання вранці, ввечері та за несприятливої погоди
Заряджання персональних пристроїв	Імовірність і кількість залежать від людей, стану мережі та періоду до або після відключення	Телефони, студентські ноутбуки	Перед відключеннями і після них навантаження заряджання збільшується
Паяльні та лабораторні роботи	Кількість активних місць залежить від типу заняття, випадкового візиту та стану системи	Пайка, лабораторні БЖ, локальне освітлення, клейовий пістолет	Створює середнє та високе навантаження, яке може обмежуватися під час відключень

Продовження таблиці 4.8

1	2	3	4
Тестові примусові параметри	forced_professors, forced_students, forced_kettle, forced_high_power	Штучно задані сценарії для перевірки	Дозволяє перевірити модуль у контрольованих умовах без очікування випадкової події
Високотужні короткочасні події	обмежуються станом електромережі й SoC батареї	Чайник, фен, дріль	Створює короткі піки, важливі для перевірки режиму захисту

Після визначення активностей модуль переходить до розрахунку потужності. Спочатку додається базове критичне навантаження мережевої інфраструктури. Далі, залежно від кількості активних викладачів і студентів, додаються робочі станції, зарядні пристрої, студентські ноутбуки, паяльні місця, лабораторні блоки живлення, локальне освітлення, чайник та інші короткочасні пристрої. Для частини пристроїв використовується не постійна потужність, а поведінкова логіка. Наприклад, паяльна станція не весь час працює на максимальній потужності, а має чергування активного нагріву та підтримання температури.

Як видно з рисунка 4.10, модуль споживання спочатку отримує контекст від інших частин системи, після чого визначає активність людей і обладнання. Запит на споживання формується до рішення EMS, тобто він є вхідним параметром для подальшого розподілу енергії. Уже після цього EMS визначає, чи може система повністю покрити навантаження, чи необхідно перейти до захисного режиму.

Окремою частиною модуля є реакція поведінки споживачів на стан електромережі та заряд акумуляторної батареї. У моделі виділено кілька станів мережевого контексту: нормальна робота, очікуване відключення, активне відключення та відновлення після відключення. Якщо мережа працює нормально, активності дозволяються без додаткових обмежень. Якщо очікується відключення,

модель збільшує імовірність заряджання телефонів і ноутбуків, але зменшує частину енергоємних робіт. Це імітує підготовку людей до можливої втрати живлення.

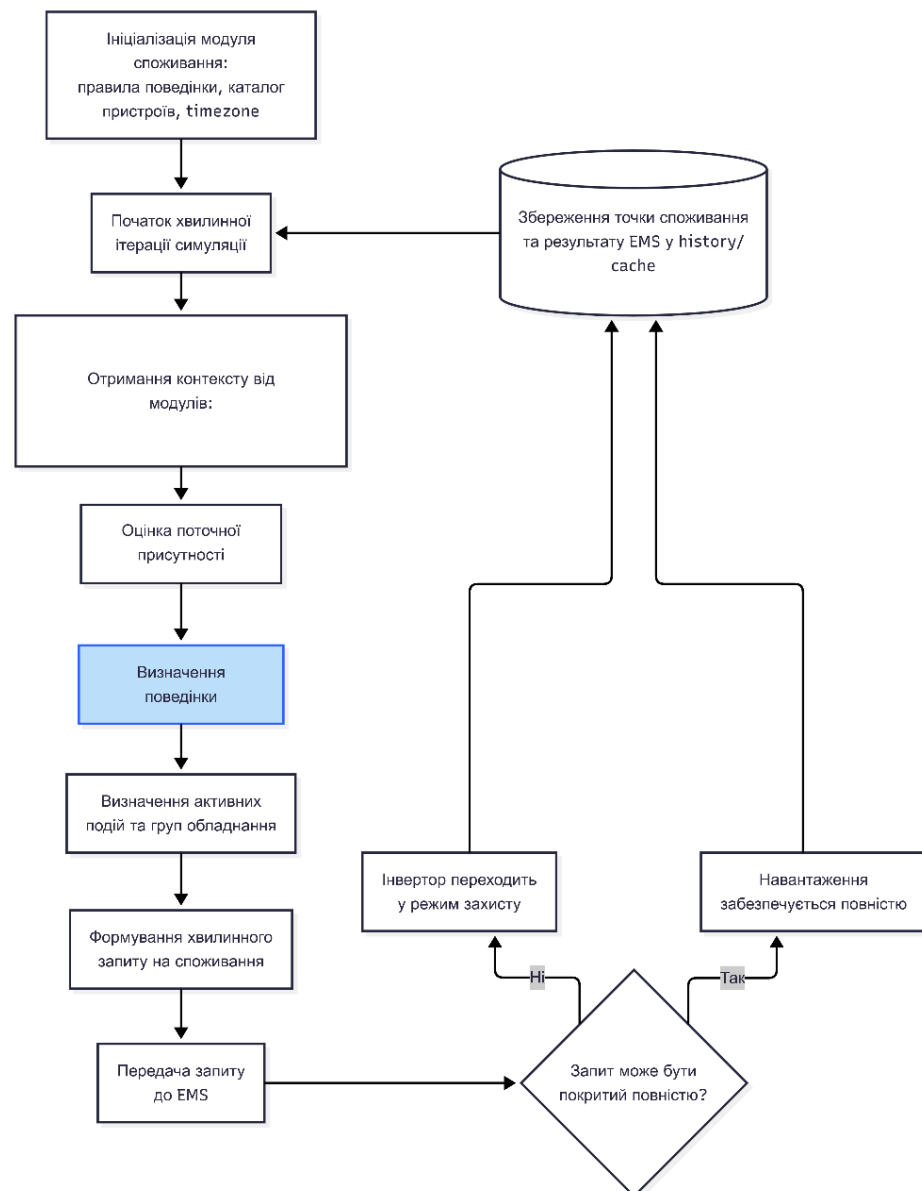


Рисунок 4.10 – Блок-схема загального алгоритму формування споживання електроенергії

Під час активного відключення поведінка залежить від рівня заряду батареї. Для цього SoC поділяється на п'ять діапазонів: comfortable, cautious, restricted, critical та emergency. Якщо заряд високий, частина активностей ще може виконуватися. Якщо заряд зменшується, пайка, лабораторне обладнання,

заряджання ноутбуків і високопотужні пристрої поступово обмежуються. У критичному та аварійному режимах модель залишає лише найнеобхідніше або мінімальне навантаження.

Рисунок 4.11 показує, що зміна поведінки відбувається не після рішення EMS, а ще на етапі формування запиту споживання. Це важливо, оскільки реальні користувачі також змінюють поведінку залежно від обставин: заряджають пристрої перед відключенням, обмежують роботу потужного обладнання під час відключення та повертаються до нормальної активності після відновлення живлення.

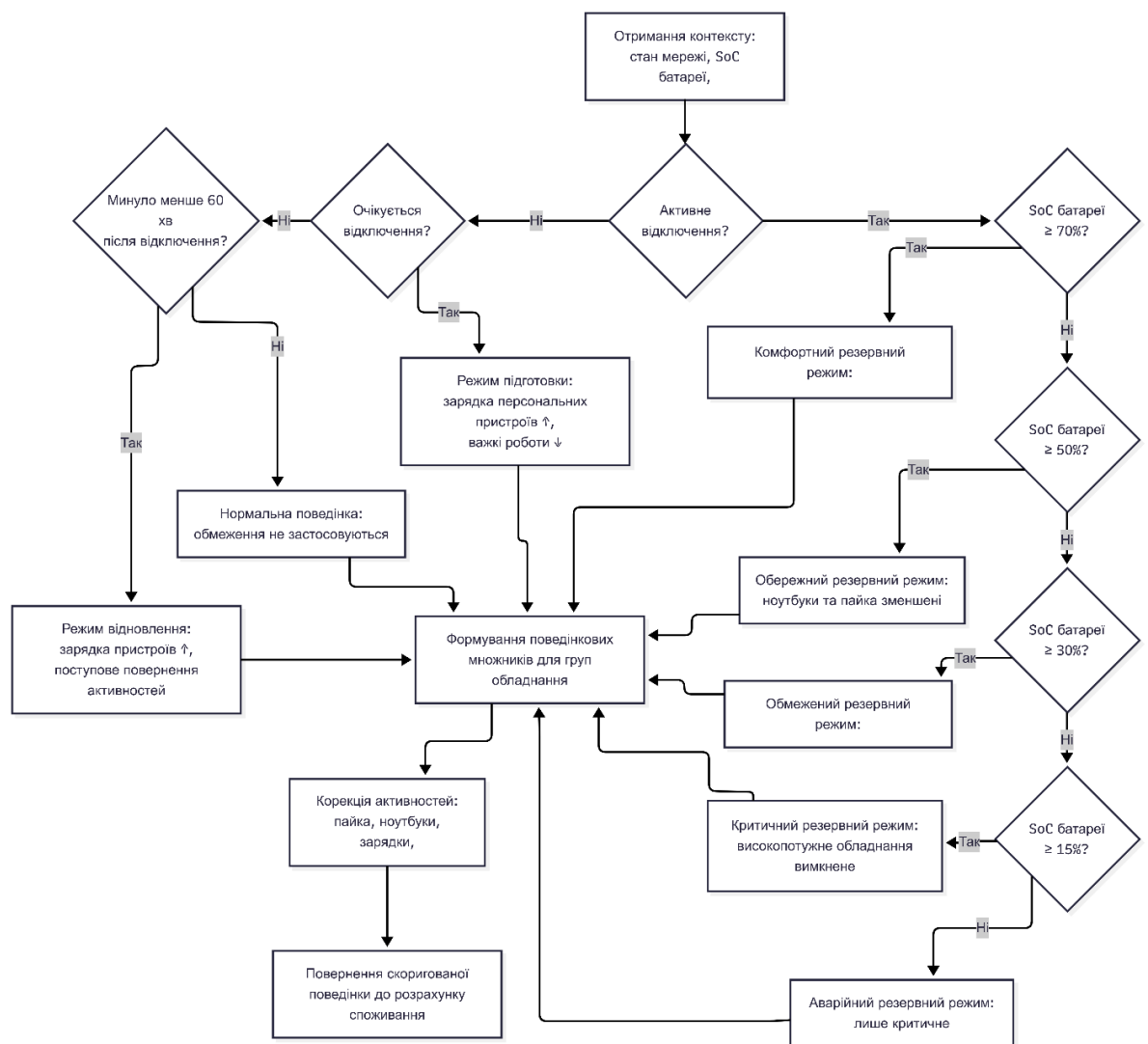


Рисунок 4.11 – Блок-схема алгоритму корекції поведінки споживачів залежно від стану електромережі та заряду акумуляторної батареї

Таблиця 4.9 – Коефіцієнти корекції споживання залежно від стану мережі та SoC батареї

Умова роботи системи	Діапазон SoC	Множник пайки	Множник заряджання телефонів	Множник ноутбуків	Високопотужні пристрої
1	2	3	4	5	6
Нормальна робота мережі	Будь-який	1,00	1,00	1,00	Дозволені за сценарієм активності
Очікуване відключення	Будь-який	0,60	1,30	1,20	Обмежуються; важке обладнання небажане
Відновлення після відключення	Будь-який	0,75	1,40	1,30	Дозволені за сценарієм активності
Активне відключення	comfortable $\geq 70\%$	0,60	0,70	0,70	Можливі чайник, фен, дріль і клейовий пістолет
Активне відключення	cautious $\geq 50\%$	0,30	0,50	0,50	Можливі лише менш критичні короткі події: дріль або клейовий пістолет
Активне відключення	restricted $\geq 30\%$	0,10	0,25	0,25	Дозволяються лише мінімальні допоміжні події
Активне відключення	critical $\geq 15\%$	0,00	0,10	0,10	Високопотужні пристрої заборонені
Активне відключення	emergency $< 15\%$	0,00	0,00	0,00	залишаються тільки критичні споживачі

Таблиця 4.9 - узагальнює поведінкові обмеження, які застосовуються до різних груп навантаження. Для паяльних робіт використовується окремий

множник, який зменшує кількість активних паяльних місць під час підготовки до відключення та особливо під час роботи від батареї. Для телефонів і ноутбуків, навпаки, перед відключенням і після нього множник збільшується, оскільки користувачі намагаються зарядити персональні пристрої. Високопотужні пристрої мають найжорсткіші обмеження, оскільки вони можуть швидко перевантажити акумуляторну систему або інвертор.

Погодні умови та освітленість також впливають на споживання. Якщо людей у лабораторії немає, освітлення не додається. Якщо люди присутні, потреба в освітленні залежить від часу доби, погодного стану та розрахованого часу сходу і заходу Сонця. У темний час доби або за умов туману, дощу, снігу чи сильного хмарного покриву модель збільшує кількість активних ламп. Це дозволяє врахувати додаткове навантаження, яке виникає не через роботу обладнання, а через умови освітлення приміщення.

Для перевірки та подальшого використання в системі хвилинні точки можуть агрегуватися у 15-хвилинні значення. При цьому енергія за 15-хвилинне вікно визначається як сума хвилинних значень потужності, поділених на 60. Така агрегація дає змогу отримати як детальний хвилинний профіль для інтегрованої симуляції, так і більш компактні дані для збереження та аналізу.

4.5 Програмна реалізація модуля автоматизації режимів заряджання

Модуль EMS є центральною частиною програмної системи, оскільки саме він перетворює дані від інших симуляційних модулів на конкретне рішення про розподіл енергії. Модулі сонячної генерації, електромережі, акумуляторної батареї та споживання формують вхідні умови, але самі по собі не визначають, як саме система повинна діяти в поточний момент. Це рішення приймається модулем автоматизації режимів заряджання.

У межах роботи EMS реалізовано як rule-based систему. Такий підхід був обраний через прозорість логіки, простоту тестування та можливість пояснити кожне прийняте рішення. На відміну від оптимізаційних або машинно-навчальних підходів, rule-based EMS не потребує великого набору навчальних даних і дозволяє явно задати реакцію системи на типові стани: стабільну роботу мережі, підвищений ризик відключень, активне відключення, низький заряд батареї або перевищення можливостей інвертора.

Основною задачею EMS є вибір режиму роботи та формування енергетичних потоків між сонячною генерацією, централізованою мережею, акумуляторною батареєю та навантаженням. Для кожного кроку симуляції EMS визначає, яка частина навантаження покривається сонячною генерацією, чи потрібно використовувати батарею, чи дозволене заряджання від мережі, яку потужність заряду потрібно запросити та чи має бути ввімкнений інвертор.

Результатом роботи EMS є не лише назва режиму, а повний об'єкт рішення. Він містить вибраний режим, оцінку ризику, причину прийнятого рішення, стан інвертора, ознаку активного захисту, запитану потужність заряду, енергію розряду батареї за останню хвилину та розподіл потоків енергії між джерелами й навантаженням. Завдяки цьому модуль EMS виступає з'єднувальною ланкою між симуляційними даними та подальшим оновленням стану батареї.

4.5.1 Вхідні дані, конфігурація та режими EMS

На вхід EMS передається поточний стан системи. До нього входять час симуляції, факт доступності централізованої електромережі, потужність сонячної генерації, поточна потужність споживання, SoC та SoH акумуляторної батареї, поточний запас енергії, доступна корисна ємність, напруга батареї, статус батареї та максимальна потужність її заряду. Також передається попередній стан електромережі та коротка історія відключень, яка використовується для оцінки ризику.

Конфігурація EMS задає основні експлуатаційні межі. У ній визначено ліміт вихідної потужності інвертора, критичний поріг SoC, резервний поріг SoC, нормальну ціль заряду, резервну ціль заряду, часовий інтервал дешевого тарифу, дозвіл заряджання батареї від електромережі та тривалість періоду післяаварійного відновлення. Ці параметри дозволяють змінювати поведінку EMS без зміни програмного коду.

У поточній реалізації EMS підтримує кілька конфігураційних режимів: `auto`, `grid_priority`, `solar_priority`, `self_consumption`, `battery_priority`, `backup_reserve` та `force_charge`. Окремо існують службові режими й прапори, які не обираються користувачем напряду, але виникають під час роботи системи. До них належать `outage_mode`, `battery_protection`, `inverter_protection_shutdown` та `post_outage_recovery`.

Таблиця 4.10 показує, що режими EMS відрізняються не лише назвою, а й цільовою поведінкою. У режимі `grid_priority` система переважно використовує централізовану мережу та не розряджає батарею. У режимі `solar_priority` сонячна генерація першою покриває навантаження, а надлишок може бути спрямований у батарею. У режимі `self_consumption` батарея використовується для зменшення споживання з мережі, але тільки до резервного порогу. У режимі `backup_reserve` батарея зберігається для можливих відключень, а в `force_charge` пріоритетом стає швидке доведення батареї до резервного рівня.

Таблиця 4.10 – Основні режими роботи EMS

Режим EMS	Тип	Умова або призначення	Основна дія та обмеження
<code>auto</code>	конфігураційний	Автоматичний вибір режиму за оцінкою ризику.	Обирає режим за <code>auto_risk_score</code> .
<code>grid_priority</code>	конфігураційний	Пріоритет мережі або дешевий тариф.	Мережа живить навантаження; батарея не розряджається.

Продовження таблиці 4.10

grid_priority	конфігураційний	Пріоритет мережі або дешевий тариф.	Мережа живить навантаження; батарея не розряджається.
solar_priority	конфігураційний	Пріоритет сонячної генерації.	Сонце живить навантаження; мережа покриває залишок.
self_consumption	конфігураційний	Низький ризик, зменшення споживання з мережі.	Сонце й батарея покривають навантаження вище резерву.
battery_priority	конфігураційний	Ручне активніше використання батареї.	Батарея живить навантаження до критичного SoC.
backup_reserve	конфігураційний / автоматичний	Підвищений ризик відключень.	Розряд батареї блокується; резерв зберігається.
force_charge	конфігураційний / автоматичний	Високий ризик або потреба швидкого заряду.	Заряд батареї пріоритетний; розряд вимкнений.
outage_mode	службовий	Активне відключення мережі.	Сонце живить першим; батарея покриває залишок.
battery_protection	службовий прапор / захист	Критичний або недоступний стан батареї.	Розряд блокується; можливий перехід у захист.
Inverter protection shutdown	службовий захист	Перевищення ліміту інвертора або батареї.	Інвертор вимикається; навантаження не подається.
post_outage_recovery	службовий прапор	Період після нещодавнього відключення.	Додає прапор обережнішого відновлення резерву.

Наявність службових режимів важлива для коректної роботи системи в аварійних умовах. Outage_mode використовується під час активного відключення електромережі. Battery_protection блокує розряд батареї у випадках, коли її стан не

дозволяє безпечно віддавати енергію. `Inverter_protection_shutdown` вимикає подачу навантаження, якщо його неможливо покрити в межах можливостей інвертора або батареї. `Post_outage_recovery` позначає період після нещодавнього відновлення мережі, коли система повинна обережніше ставитися до заряду батареї.

4.5.2 Розрахунок ризику та вибір автоматичного режиму

У режимі `auto` EMS самостійно вибирає стратегію роботи. Для цього використовується показник `auto_risk_score`, який приймає значення від 0 до 100. Він не є фізичною величиною, а виконує роль узагальненої оцінки ризику нестабільної роботи системи. Чим вищий цей показник, тим обережнішою стає поведінка EMS і тим більший пріоритет надається збереженню або відновленню заряду батареї.

Оцінка ризику зростає, якщо централізована мережа зараз недоступна, якщо за останні години або доби були відключення, якщо кількість відключень за останню добу або три доби є великою, якщо батарея не встигла відновитися після попереднього відключення або якщо мінімальний `SoC` за останню добу падав нижче резервного чи критичного порогу. Також ризик збільшується, якщо поточний `SoC` батареї знаходиться нижче резервного або критичного значення.

Оцінка ризику зменшується, якщо мережа стабільна, відключень давно не було, а батарея вже заряджена до нормальної або резервної цілі. Це дозволяє EMS переходити від консервативної поведінки до економнішого використання батареї, коли зовнішні умови не створюють високого ризику.

Як видно з рисунка 4.12, спочатку EMS отримує поточні вхідні дані та історичний контекст. Далі обчислюється оцінка ризику. Якщо в конфігурації задано ручний режим, EMS використовує його без автоматичного вибору. Якщо активний режим `auto`, система вибирає режим залежно від значення ризику.

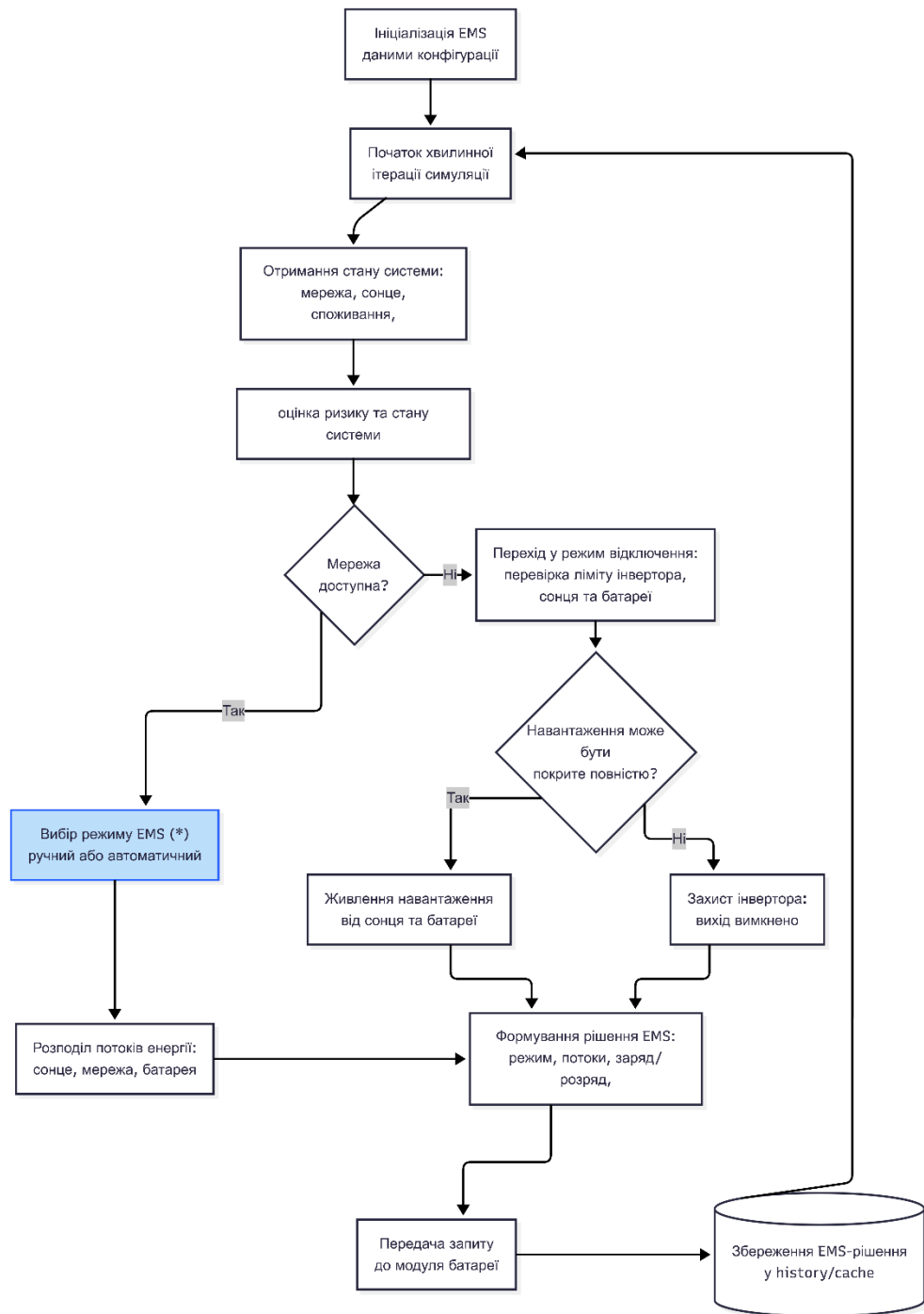


Рисунок 4.12 – Блок-схема алгоритму вибору режиму EMS

При низькому ризику EMS переходить до *self_consumption*, тобто дозволяє використовувати сонячну генерацію та батарею для зменшення споживання з мережі. Якщо при цьому активний дешевий тариф і заряджання від мережі дозволене, EMS може вибрати *grid_priority*, щоб підготувати батарею дешевшою електроенергією. При середньому ризику використовується *solar_priority*. При підвищеному ризику EMS переходить у *backup_reserve*, зберігаючи батарею для

майбутніх відключень. При високому ризику вибирається `force_charge`, якщо батарея ще не заряджена до резервної цілі та заряджання від мережі дозволене.

Такий алгоритм дозволяє уникнути жорсткої прив'язки EMS до одного сценарію. У стабільних умовах система може поводитися економніше, а за наявності частих або нещодавніх відключень переходить до більш обережної стратегії. Це відповідає практичній меті модуля: не просто заряджати батарею, а адаптувати режим заряджання до стану енергетичної системи.

4.5.3 Формування рішення про розподіл енергії

Після вибору режиму EMS формує рішення про розподіл потоків енергії. Якщо централізована електромережа доступна, система має можливість повністю покрити поточне навантаження. У цьому випадку різниця між режимами полягає не в самому факті живлення навантаження, а в тому, яке джерело використовується першим і чи дозволяється розряд батареї.

У режимах `backup_reserve` та `force_charge` батарея не використовується для покриття навантаження, поки мережа доступна. Навантаження в цьому випадку покривається мережею, а сонячна генерація може бути спрямована на заряд батареї. Така поведінка потрібна тоді, коли система повинна зберегти запас енергії на випадок майбутнього відключення.

У режимі `solar_priority` сонячна генерація першою покриває поточне навантаження. Якщо її недостатньо, залишок покривається централізованою мережею. Якщо сонячної генерації більше, ніж потрібно навантаженню, надлишок може бути спрямований на заряд батареї. Це базовий режим використання локальної генерації без активного розряду батареї.

У режимі `self_consumption` EMS намагається зменшити споживання з мережі. Спочатку навантаження покривається сонячною генерацією, а якщо її недостатньо, частина залишку може бути покрита батареєю. При цьому розряд батареї обмежується резервним порогом `SoC`. Отже, система використовує батарею для

економії, але не дозволяє розряджати її нижче рівня, потрібного для резервного живлення.

У режимі `battery_priority` батарея використовується активніше. Вона може покривати залишок навантаження до критичного порогу SoC. Такий режим є менш консервативним і може бути корисним для демонстрації роботи системи або для сценаріїв, у яких користувач свідомо дозволяє глибше використання накопиченої енергії.

Після визначення потоків на навантаження EMS розраховує можливе заряджання батареї. Спочатку визначається цільовий SoC для поточного режиму. Для звичайних режимів ціллю є нормальний рівень заряду, для `battery_priority` – резервний рівень, а для `backup_reserve` та `force_charge` – резервна ціль. Далі EMS визначає, скільки енергії не вистачає до цілі, і перетворює цей запас у допустиму потужність заряду на поточному кроці симуляції.

Заряджання батареї обмежується кількома факторами: максимальною потужністю заряду батареї, доступною вільною ємністю, цільовим SoC, надлишком сонячної генерації та дозволом на заряджання від мережі. Якщо доступний сонячний надлишок, він використовується першим. Якщо цього недостатньо і режим дозволяє заряджання від мережі, решта запиту може бути покрита централізованою мережею.

У результаті EMS формує значення потоків: `solar_to_load`, `solar_to_battery`, `grid_to_load`, `grid_to_battery` та `battery_to_load`. Саме ці значення потім зберігаються в історії симуляції та використовуються для аналізу роботи системи. Така деталізація дає змогу не лише бачити вибраний режим, а й перевіряти, як саме енергія розподілялася в кожен момент часу.

4.5.4 Робота EMS під час відключення та захисні режими

Якщо централізована електромережа недоступна, EMS переходить у режим `outage_mode`. У цьому режимі заряджання від мережі неможливе, тому система

може використовувати лише сонячну генерацію та акумуляторну батарею. Логіка роботи під час відключення відрізняється від логіки при доступній мережі, оскільки помилка в оцінці доступної потужності може призвести до перевантаження інвертора або надмірного розряду батареї.

Під час відключення сонячна генерація першою спрямовується на покриття навантаження. Якщо її достатньо, батарея не розряджається. Якщо сонячної генерації недостатньо, EMS перевіряє, чи може батарея покрити залишок навантаження. Для цього враховується поточна енергія батареї, її SoC, критичний поріг і ліміт вихідної потужності інвертора.

Алгоритм роботи EMS під час відключення електромережі наведено на рисунку 4.13.

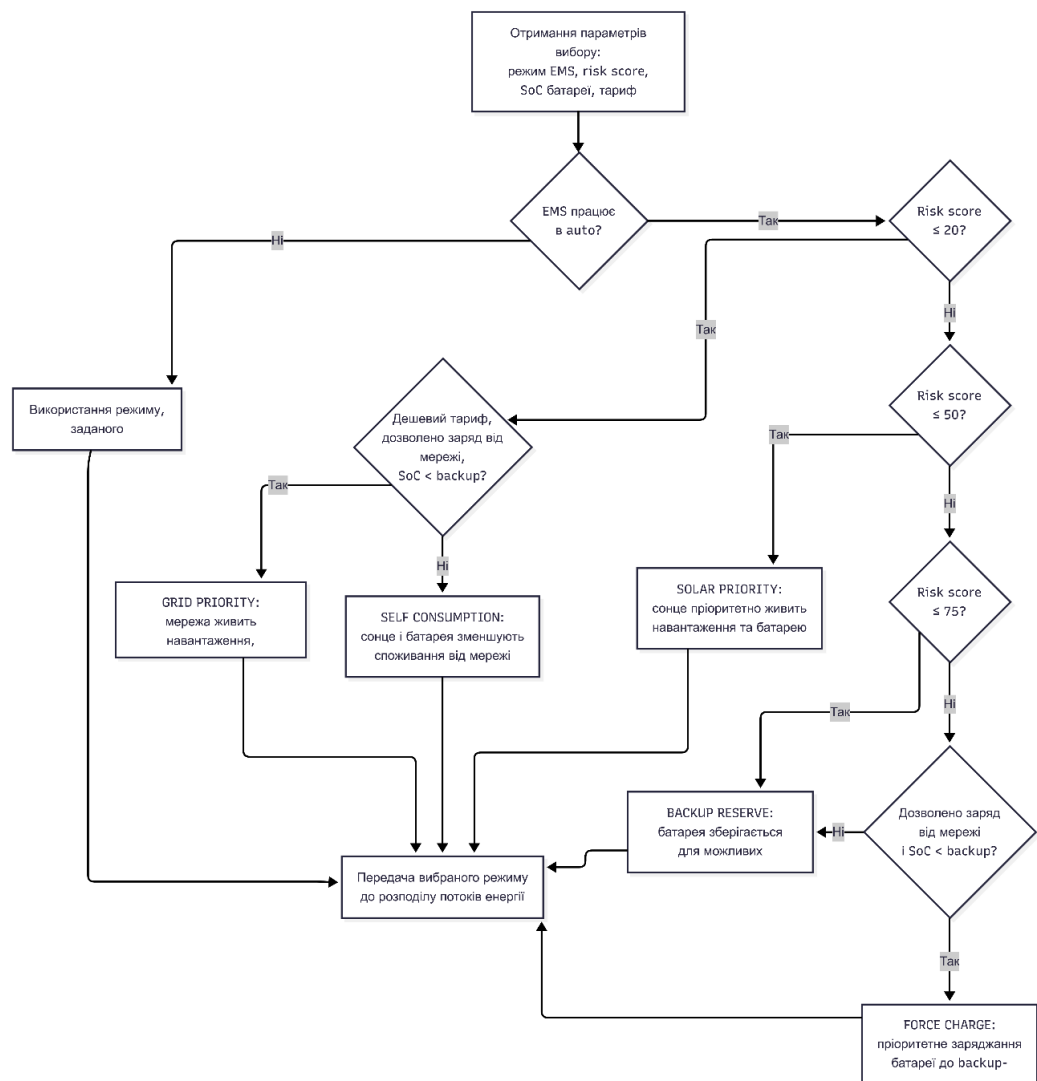


Рисунок 4.13 – Блок-схема роботи EMS під час відключення електромережі

Як видно з рисунка 4.13, першим обмеженням є ліміт інвертора. Якщо поточне навантаження перевищує допустиму вихідну потужність інвертора, EMS переходить у захисне вимкнення. У цьому випадку інвертор вимикається, ефективно подане навантаження стає нульовим, а рішення позначається як `inverter_protection_shutdown`.

Якщо навантаження не перевищує ліміт інвертора, EMS перевіряє можливість використання батареї. Розряд блокується, якщо статус батареї недоступний, батарея порожня, її корисна ємність дорівнює нулю, енергії немає або SoC знаходиться на рівні критичного порогу чи нижче нього. У таких випадках активується `battery_protection`, оскільки подальший розряд може бути небезпечним для накопичувача або не може бути виконаний фізично.

Якщо батарея може віддавати енергію, EMS порівнює залишкове навантаження з доступною потужністю батареї. Якщо батарея та інвертор можуть покрити залишок, система продовжує жити навантаження від сонячної генерації та батареї. Якщо доступної потужності недостатньо, EMS також переходить у захисне вимкнення. Таким чином, під час відключення система не допускає часткового неявного перевантаження, а явно фіксує неможливість покриття навантаження.

Окремо важливо, що навіть у режимі захисного вимкнення надлишок сонячної генерації може бути використаний для заряджання батареї. Тобто відключення навантаження не означає повне припинення роботи енергетичної частини системи. Якщо сонячна генерація доступна, EMS може спрямувати її на відновлення заряду батареї, щоб система могла повернутися до нормальної роботи після зменшення навантаження або покращення умов.

4.5.5 Передача рішення EMS до батареї та збереження результатів

EMS працює в межах інтегрованого циклу симуляції. На кожному хвилинному кроці система спочатку отримує поточну сонячну генерацію, стан

електромережі та попередній стан батареї. Далі модуль споживання формує поточний запит навантаження. Після цього створюється вхідний об'єкт EMS, який містить усі ці дані, а також коротку історію відключень і конфігурацію режимів заряджання.

Після формування рішення EMS воно передається до модуля батареї. Для цього використовується запит на крок батареї, у якому вказується енергія, яку потрібно віддати за останню хвилину, прапорець використання батареї для покриття навантаження та запитана потужність заряду. Таким чином, EMS не змінює стан батареї напряму, а формує керуючий запит, який потім перевіряється фізичними обмеженнями модуля батареї.

Після виконання кроку симуляції результати зберігаються в історії або кеші. Для EMS зберігається вибраний режим, оцінка ризику, активність захисту, стан інвертора, цільовий SoC, запитана потужність заряду, потік від мережі до навантаження, потік від мережі до батареї, потік від сонячної генерації до навантаження, потік від сонячної генерації до батареї та потік від батареї до навантаження.

Завдяки такій структурі EMS завершує логічний ланцюг інтегрованої симуляції. Модулі генерації, електромережі, батареї та споживання створюють стан системи, а EMS визначає реакцію на цей стан. Отримане рішення потім змінює стан батареї та формує дані для подальшого аналізу.

Висновки до розділу 4

У четвертому розділі було описано програмну реалізацію основних симуляційних модулів та модуля автоматизації режимів заряджання акумуляторної батареї лабораторного стенду SmartEnergy Lab. Розглянуті модулі забезпечують формування даних про сонячну генерацію, стан централізованої електромережі, акумуляторну батарею, споживання електроенергії та роботу EMS.

Було розроблено модуль симуляції сонячної генерації, який формує історичні та поточні значення потужності з урахуванням часу доби, сезону, параметрів сонячних панелей і погодних умов. Модуль електромережі дозволяє моделювати не лише факт наявності або відсутності живлення, а й нестабільність енергосистеми через пошкодження, відновлення, дефіцит і графіки відключень. Це дає змогу перевіряти роботу системи не в ідеальних умовах, а в сценаріях, наближених до реальних умов нестабільного електропостачання.

Окремо було реалізовано модуль акумуляторної батареї, який відстежує SoC, SoH, доступну енергію, напругу, статус батареї та обмеження заряду й розряду. Така модель не є надмірно деталізованою фізико-хімічною моделлю, але достатньо точно відображає ті параметри, які потрібні EMS для прийняття рішень. Модуль споживання електроенергії формує стохастичний профіль навантаження лабораторії з урахуванням присутності людей, занять, випадкових візитів, роботи обладнання, освітленості, стану мережі та заряду батареї.

Ключовим результатом розділу є реалізація EMS — rule-based модуля автоматизації режимів заряджання. Він отримує дані від усіх симуляційних модулів, оцінює ризик нестабільної роботи електромережі, вибирає режим роботи та формує рішення про розподіл енергії між сонячною генерацією, мережею, батареєю і навантаженням. У модулі передбачено як звичайні режими роботи, так і захисні сценарії для випадків активного відключення, критичного заряду батареї або перевищення можливостей інвертора.

Таким чином, у межах четвертого розділу було сформовано функціональну програмну основу системи. Розроблені модулі працюють не як ізольовані частини, а як єдина інтегрована симуляція, у якій вихідні дані одного модуля впливають на роботу інших. Це дозволяє демонструвати логіку автоматизації режимів заряджання акумуляторної батареї в умовах змінної сонячної генерації, нестабільної електромережі, змінного споживання та обмеженого ресурсу накопичувача.

5 ОПИС ГРАФІЧНОГО ІНТЕРФЕЙСУ ПРОГРАМНОЇ СИСТЕМИ

Графічний інтерфейс програмної системи призначений для демонстрації роботи розроблених симуляційних модулів та модуля EMS. Він не є окремою ціллю роботи, а використовується як засіб взаємодії з системою, перегляду поточного стану лабораторного стенду та перевірки результатів автоматизації режимів заряджання.

Інтерфейс об'єднує в одному середовищі дані про сонячну генерацію, стан електромережі, акумуляторну батарею, споживання електроенергії та рішення EMS. Завдяки цьому користувач може не тільки бачити окремі параметри системи, а й перевіряти взаємозв'язок між ними: як зміна сонячної генерації, навантаження або доступності електромережі впливає на режим роботи EMS і стан акумуляторної батареї.

5.1 Призначення графічного інтерфейсу

Основним користувачем інтерфейсу є оператор, який перевіряє роботу програмної системи. Він може переглядати загальний стан стенду, відкривати окремі модулі, аналізувати історичні графіки, контролювати поточне рішення EMS та змінювати режим її роботи.

На діаграмі прецедентів показано одного актора – користувача системи. Він взаємодіє з основними функціями інтерфейсу: переглядом загального стану системи, переглядом сонячної генерації, стану електромережі, акумуляторної батареї, споживання електроенергії, рішення EMS та історичних графіків. Окремо передбачено можливість зміни режиму роботи EMS та запуску або оновлення симуляції.

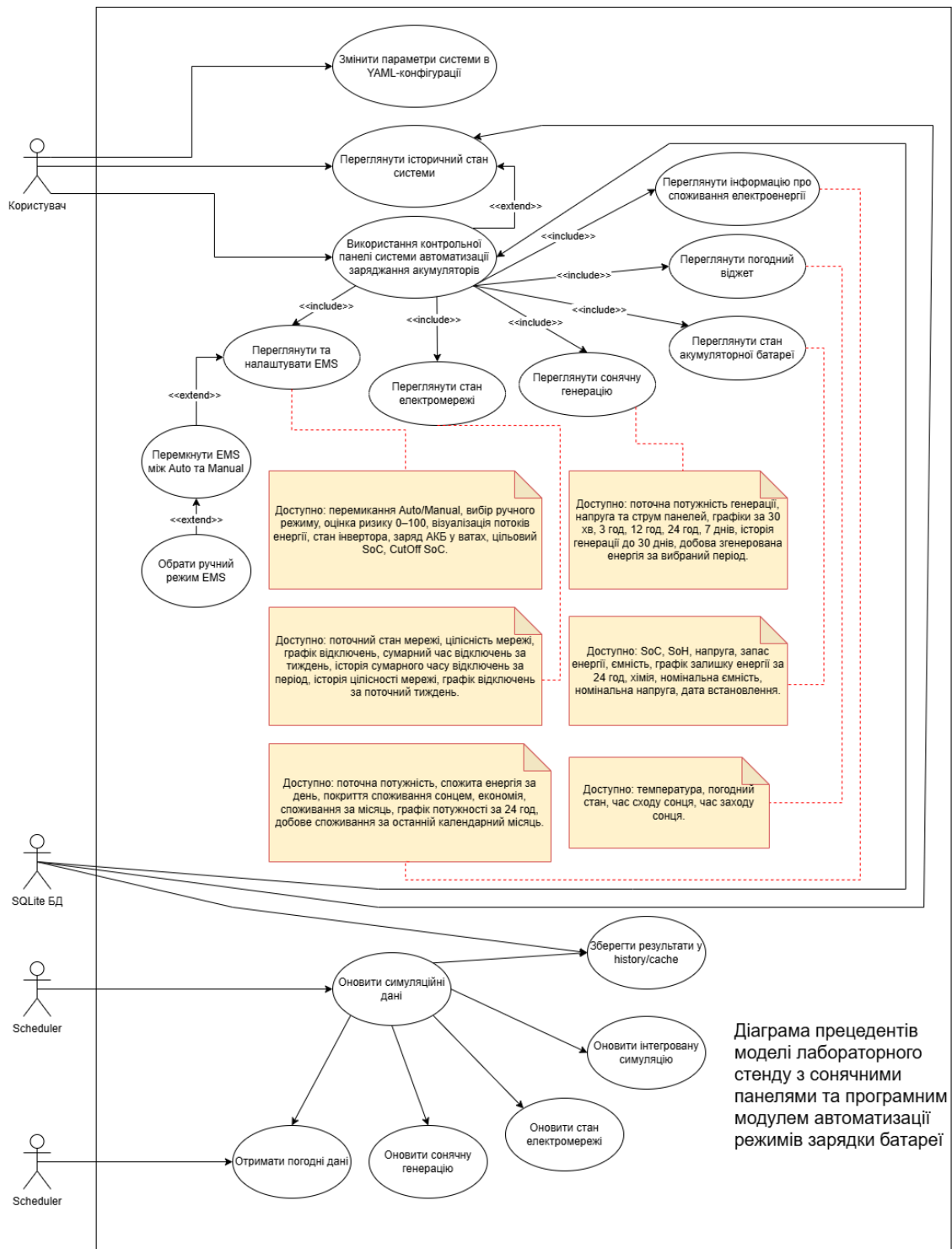


Рисунок 5.1 – Діаграма прецедентів графічного інтерфейсу програмної системи

Така структура відповідає призначенню інтерфейсу як демонстраційного та контрольного середовища. Користувач не керує фізичними пристроями напряму, а взаємодіє з програмною моделлю стенду, переглядає її стан і перевіряє результати роботи алгоритмів автоматизації.

5.2 Загальний вигляд інтерфейсу

На основному екрані відображаються ключові частини системи: сонячна генерація, електромережа, акумуляторна батарея, навантаження та EMS. Для кожного модуля показуються поточні значення, які використовуються в інтегрованій симуляції. Це дозволяє швидко оцінити стан стенду без переходу до окремих технічних даних (рис 5.2).

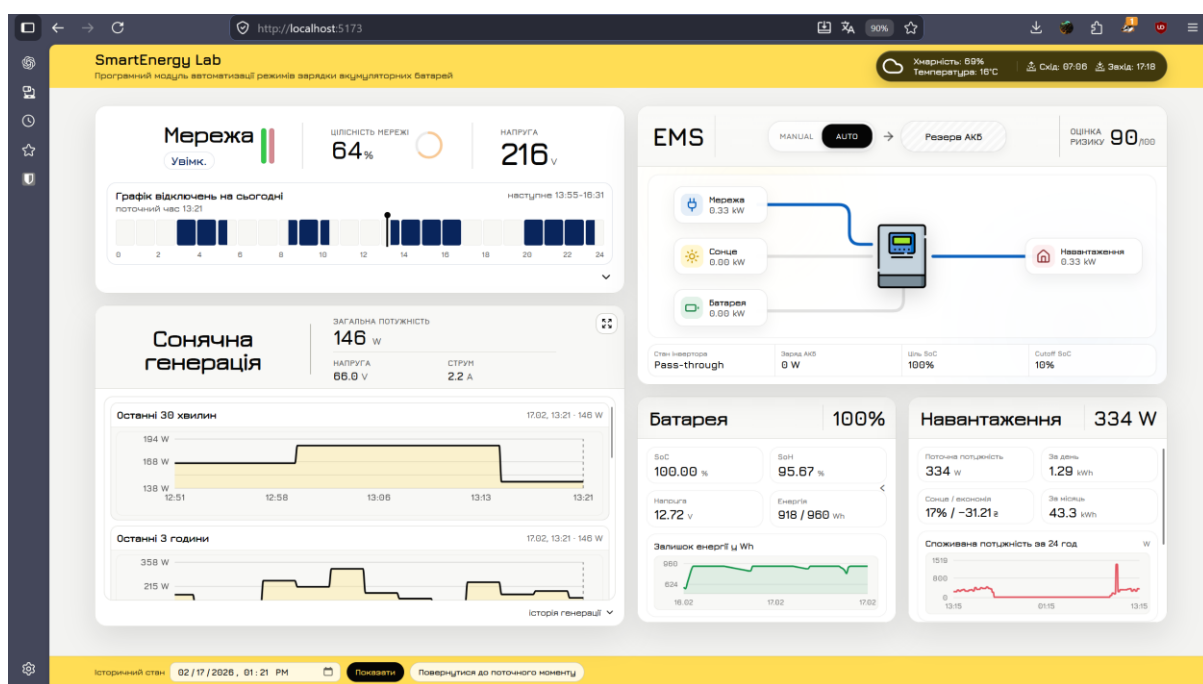


Рисунок 5.2 – Загальний вигляд графічного інтерфейсу програмної системи

Інтерфейс також відображає історичні графіки, завдяки яким можна перевірити зміну параметрів у часі. Такі графіки потрібні для демонстрації того, що система працює не з окремим статичним значенням, а з послідовністю симуляційних точок, які зберігаються та використовуються для аналізу.

Окреме значення має відображення EMS. Воно показує не тільки поточний режим роботи, а й результат прийнятого рішення: які джерела енергії використовуються, чи активний захист, чи подається навантаження та чи виконується заряджання акумуляторної батареї. Таким чином, інтерфейс робить результат роботи алгоритму EMS доступним для перевірки.

5.3 Перегляд додаткових модулів системи

Крім загального вигляду, інтерфейс передбачає можливість перегляду розгорнутих модулів. Це потрібно для детальнішого аналізу окремих частин симуляції без перевантаження основного екрана. Приклад такого перегляду наведено на рисунку 5.3.

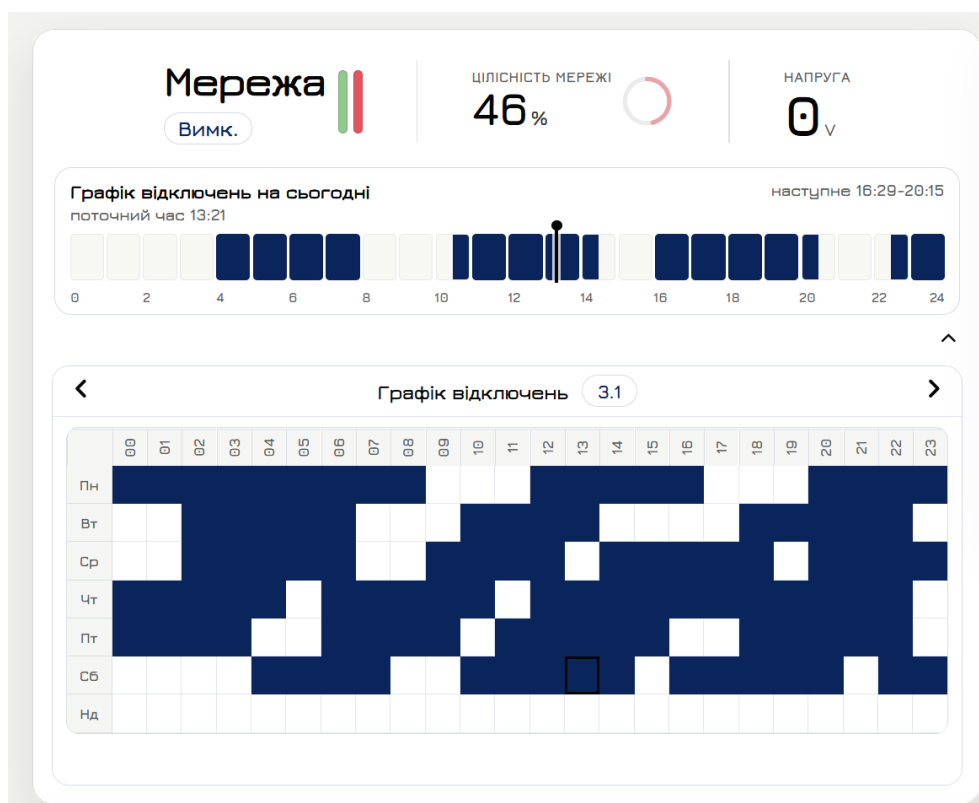


Рисунок 5.3 – Приклад розгорнутого перегляду модулів програмної системи

У розгорнутому вигляді користувач може детальніше переглянути стан окремого модуля. Для сонячної генерації це можуть бути графіки вироблення енергії та погодні умови. Для електромережі – поточна доступність, рівень цілісності, напруга та історія відключень. Для акумуляторної батареї – SoC, SoH, напруга, поточна енергія та статус. Для модуля споживання – поточне навантаження й ознаки активних сценаріїв. Для EMS – вибраний режим, потоки енергії та захисні стани.

Такий підхід дозволяє зберегти основний екран компактним, але водночас залишити доступ до детальнішої інформації. У межах дипломної роботи це

важливо, оскільки інтерфейс використовується не лише для візуального представлення даних, а й для перевірки того, що всі розроблені модулі працюють як єдина система.

Висновки до розділу 5

У п'ятому розділі було розглянуто графічний інтерфейс програмної системи. Було показано, що інтерфейс забезпечує перегляд загального стану стенду, доступ до окремих симуляційних модулів, перегляд історичних графіків, контроль рішень EMS та зміну режиму її роботи.

Графічний інтерфейс виконує демонстраційну та контрольну функцію. Він підтверджує, що модулі сонячної генерації, електромережі, акумуляторної батареї, споживання та EMS не існують окремо, а інтегровані в єдину програмну систему. Завдяки цьому користувач може оцінювати як поточний стан системи, так і результат роботи алгоритмів автоматизації режимів заряджання та напряду керувати режимами зарядки.

ВИСНОВКИ

У межах дипломної роботи було розроблено програмну систему для моделювання роботи лабораторного стенду SmartEnergy Lab та автоматизації режимів заряджання акумуляторних батарей у системі з сонячною генерацією, акумуляторним накопичувачем, навантаженням і централізованою електромережею. Поставлену мету роботи досягнуто: створено програмне забезпечення, яке дозволяє симулювати основні процеси в енергетичній системі стенду, зберігати отримані дані, відображати їх у веб-інтерфейсі та формувати рішення щодо розподілу доступної енергії.

У першій частині роботи було обґрунтовано актуальність теми. Показано, що задача автоматизації режимів заряджання є важливою як для українських умов, де нестабільність електропостачання пов'язана з пошкодженнями енергетичної інфраструктури, так і для довгострокового розвитку систем локальної генерації та накопичення енергії. Було визначено, що через обмеження реального апаратного стенду доцільним є створення програмної моделі, яка дозволяє відтворювати поведінку системи та перевіряти алгоритми автоматизації без прямого керування фізичними пристроями.

У ході аналізу предметної області було розглянуто підходи до моделювання сонячної генерації, акумуляторних батарей, систем EMS/BMS, споживання електроенергії та нестабільної роботи електромережі. Проведений огляд показав, що готові рішення часто орієнтовані на окремі задачі або потребують великої кількості точних вхідних параметрів. Тому для лабораторного стенду було обрано підхід із власними спрощеними симуляційними модулями, які забезпечують достатній рівень деталізації без надмірного ускладнення.

Для реалізації системи було обрано стек програмних засобів на основі Python, FastAPI, SQLite, YAML-конфігурації, бібліотек для числової обробки та React-інтерфейсу. Такий набір інструментів дозволив розділити серверну логіку,

симуляційні модулі, збереження даних та графічний інтерфейс, а також забезпечити відтворюваний запуск системи через Docker.

У практичній частині роботи було реалізовано модуль симуляції сонячної генерації, який формує історичні та поточні значення потужності з урахуванням географічного положення, часу доби, сезону, параметрів панелей і погодних умов. Було створено модуль симуляції електромережі, який відтворює сценарії стабільної роботи, пошкоджень, дефіциту, відключень та поступового відновлення. Також було реалізовано модель акумуляторної батареї з урахуванням SoC, SoH, доступної ємності, напруги, ефективності заряду й розряду та обмежень за потужністю. Окремо було створено стохастичний модуль споживання, який формує змінний профіль навантаження лабораторії з урахуванням людей, занять, випадкових подій, освітлення, стану мережі та рівня заряду батареї.

Ключовим результатом роботи стала реалізація rule-based EMS-модуля. Він отримує дані від усіх симуляційних підсистем, оцінює ризик нестабільної роботи електромережі, вибирає режим роботи та формує рішення щодо розподілу енергії між сонячною генерацією, мережею, батареєю і навантаженням. У модулі передбачено режими пріоритету мережі, сонячної генерації, самоспоживання, збереження резерву, примусового заряджання, а також захисні сценарії під час відключення або перевищення можливостей батареї та інвертора.

Було також реалізовано графічний веб-інтерфейс, який дозволяє переглядати загальний стан системи, окремі симуляційні модулі, історичні графіки та рішення EMS. Інтерфейс виконує демонстраційну і контрольну функцію та підтверджує, що розроблені модулі працюють не ізольовано, а як єдина програмна система.

Отже, результатом дипломної роботи є програмна система, яка демонструє принципи автоматизованого керування режимами заряджання акумуляторної батареї в умовах змінної сонячної генерації, нестабільної електромережі, змінного споживання та обмеженого ресурсу накопичувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. System Advisor Model (SAM) photovoltaic model technical reference. 2016 update. Golden, CO : National Renewable Energy Laboratory, 2016. URL: <https://www.nrel.gov/docs/fy18osti/67399.pdf> (дата звернення: 13.05.2026).
2. pvlib python: 2023 project update. Journal of Open Source Software. 2023. URL: <https://doi.org/10.21105/joss.05994> (дата звернення: 13.05.2026).
3. Battery energy storage systems: a review of energy management systems and health metrics. Energies. 2024. Vol. 17, no. 5. P. 1250. URL: <https://doi.org/10.3390/en17051250> (дата звернення: 15.05.2026).
4. Review on battery state estimation and management solutions for next-generation connected vehicles. Energies. 2024. Vol. 17, no. 1. P. 202. URL: <https://doi.org/10.3390/en17010202> (дата звернення: 15.05.2026).
5. Python battery mathematical modelling (PyBaMM). Journal of Open Research Software. 2021. Vol. 9, no. 1. URL: <https://doi.org/10.5334/jors.309> (дата звернення: 17.05.2026).
6. A review on maximum power point tracking for photovoltaic systems with and without shading conditions. Renewable and Sustainable Energy Reviews. 2016. URL: <https://doi.org/10.1016/j.rser.2016.09.013> (дата звернення: 17.05.2026).
7. Energy management of small-scale PV-battery systems: a systematic review considering practical implementation, computational requirements, quality of input data and battery degradation. Renewable and Sustainable Energy Reviews. 2019. URL: <https://doi.org/10.1016/j.rser.2019.06.007> (дата звернення: 19.05.2026).
8. A rule-based energy management system for hybrid renewable energy sources with battery bank optimized by genetic algorithm optimization. Scientific Reports. 2024. Vol. 14. URL: <https://doi.org/10.1038/s41598-024-54333-0> (дата звернення: 19.05.2026).

9. End-use load profiles for the U.S. building stock. National Renewable Energy Laboratory. URL: <https://www.nrel.gov/buildings/end-use-load-profiles.html> (дата звернення: 22.05.2026).
10. Richardson I., Thomson M., Infield D., Clifford C. Domestic electricity use: a high-resolution energy demand model. Energy and Buildings. 2010. Vol. 42, no. 10. P. 1878–1887. URL: <https://doi.org/10.1016/j.enbuild.2010.05.023> (дата звернення: 22.05.2026).
11. Designing resilient decentralized energy systems: the importance of modeling extreme events and long-duration power outages. iScience. 2021. URL: <https://doi.org/10.1016/j.isci.2021.103630> (дата звернення: 24.05.2026).
12. Ukraine's energy security and the coming winter. International Energy Agency. 2024. URL: <https://www.iea.org/reports/ukraines-energy-security-and-the-coming-winter> (дата звернення: 24.05.2026).
13. Empowering Ukraine through a decentralised electricity system. International Energy Agency. URL: <https://www.iea.org/commentaries/empowering-ukraine-through-a-decentralised-electricity-system> (дата звернення: 26.05.2026).
14. Modelling the electricity deficit in Ukraine and potential policy responses for the period June 2024 – May 2025. URL: <https://doi.org/10.5281/zenodo.12532697> (дата звернення: 26.05.2026).
15. Python documentation. Python Software Foundation. URL: <https://docs.python.org/3/> (дата звернення: 28.05.2026).
16. FastAPI documentation. URL: <https://fastapi.tiangolo.com/> (дата звернення: 28.05.2026).
17. Uvicorn documentation. URL: <https://www.uvicorn.org/> (дата звернення: 28.05.2026).
18. React documentation. Meta Open Source. URL: <https://react.dev/> (дата звернення: 30.05.2026).
19. Vite documentation. URL: <https://vite.dev/> (дата звернення: 30.05.2026).

20.Docker Compose documentation. Docker Inc. URL: <https://docs.docker.com/compose/> (дата звернення: 30.05.2026).

21.Over 1m Ukrainians without power after major Russian assault on energy system. The Guardian. 2024. URL: <https://www.theguardian.com/world/2024/mar/22/over-1m-ukrainians-without-power-after-russias-largest-assault-on-energy-sector> (дата звернення: 02.06.2026).

22.Russia launches fierce missile and drone attacks at Ukraine's infrastructure. Politico. 2024. URL: <https://www.politico.com/news/2024/11/17/russia-missile-drone-attacks-ukraine-infrastructure-00190030> (дата звернення: 02.06.2026).

23.Ukraine says it is running short of air defence missiles. Reuters. 2026. URL: <https://www.reuters.com/business/aerospace-defense/ukraine-says-it-is-running-short-air-defence-missiles-2026-05-08/> (дата звернення: 03.06.2026).

24.What is the Patriot missile system and how is it helping Ukraine? Reuters. 2025. URL: <https://www.reuters.com/business/aerospace-defense/what-is-patriot-missile-system-how-is-it-helping-ukraine-2025-07-14/> (дата звернення: 04.06.2026).

25.Russian missile upgrade outpaces Ukraine's Patriot defences. Financial Times. 2025. URL: <https://www.ft.com/content/078b8e70-a58c-47cc-b573-598850dd5685> (дата звернення: 05.06.2026).

ДОДАТОК А Лістинг розробленого модуля

`ems.py`

```
from __future__ import annotations

from dataclasses import dataclass, fields
from datetime import datetime, time
from enum import Enum
from math import isfinite
from typing import Mapping

class EmsMode(str, Enum):
    AUTO = "auto"
    GRID_PRIORITY = "grid_priority"
    SOLAR_PRIORITY = "solar_priority"
    SELF_CONSUMPTION = "self_consumption"
    BATTERY_PRIORITY = "battery_priority"
    BACKUP_RESERVE = "backup_reserve"
    FORCE_CHARGE = "force_charge"
    OUTAGE_MODE = "outage_mode"
    BATTERY_PROTECTION = "battery_protection"
    INVERTER_PROTECTION_SHUTDOWN = "inverter_protection_shutdown"
    POST_OUTAGE_RECOVERY = "post_outage_recovery"

CONFIG_MODES = {
    EmsMode.AUTO,
    EmsMode.GRID_PRIORITY,
    EmsMode.SOLAR_PRIORITY,
    EmsMode.SELF_CONSUMPTION,
    EmsMode.BATTERY_PRIORITY,
    EmsMode.BACKUP_RESERVE,
    EmsMode.FORCE_CHARGE,
}

@dataclass(frozen=True, slots=True)
class EmsConfig:
```

```

mode: EmsMode | str = EmsMode.AUTO
inverter_output_limit_w: float = 2000.0
critical_soc_percent: float = 10.0
reserve_soc_percent: float = 30.0
normal_target_soc_percent: float = 80.0
backup_target_soc_percent: float = 100.0
cheap_tariff_start: str = "23:00"
cheap_tariff_end: str = "07:00"
cheap_tariff_price_factor: float = 0.5
allow_grid_charging: bool = True
recent_outage_recovery_minutes: int = 60

def __post_init__(self) -> None:
    try:
        mode = EmsMode(self.mode)
    except ValueError as exc:
        allowed = ", ".join(sorted(mode.value for mode in CONFIG_MODES))
        raise ValueError(f"EMS mode must be one of: {allowed}") from exc
    if mode not in CONFIG_MODES:
        allowed = ", ".join(sorted(item.value for item in CONFIG_MODES))
        raise ValueError(f"EMS mode must be one of: {allowed}")
    object.__setattr__(self, "mode", mode)

    _validate_positive_finite(
        "inverter_output_limit_w",
        self.inverter_output_limit_w,
    )
    _validate_percent("critical_soc_percent", self.critical_soc_percent)
    _validate_percent("reserve_soc_percent", self.reserve_soc_percent)
    _validate_percent("normal_target_soc_percent",
self.normal_target_soc_percent)
    _validate_percent("backup_target_soc_percent",
self.backup_target_soc_percent)
    if not (
        self.critical_soc_percent
        < self.reserve_soc_percent
        < self.normal_target_soc_percent
        <= self.backup_target_soc_percent
    ):
        raise ValueError(

```

```

        "EMS SoC thresholds must satisfy critical < reserve < normal <=
backup"

    )
    object.__setattr__(
        self,
        "cheap_tariff_start",
        _normalize_clock_time(self.cheap_tariff_start),
    )
    object.__setattr__(
        self,
        "cheap_tariff_end",
        _normalize_clock_time(self.cheap_tariff_end),
    )
    _validate_positive_finite(
        "cheap_tariff_price_factor",
        self.cheap_tariff_price_factor,
    )
    if int(self.recent_outage_recovery_minutes) !=
self.recent_outage_recovery_minutes:
        raise ValueError("recent_outage_recovery_minutes must be an integer")
    if self.recent_outage_recovery_minutes < 0:
        raise ValueError("recent_outage_recovery_minutes must be non-negative")
    object.__setattr__(
        self,
        "recent_outage_recovery_minutes",
        int(self.recent_outage_recovery_minutes),
    )
    object.__setattr__(self, "allow_grid_charging",
bool(self.allow_grid_charging))

    @classmethod
    def from_station_config(cls, station_or_app_config: object) -> "EmsConfig":
        station = getattr(station_or_app_config, "station", station_or_app_config)
        raw_config = getattr(station, "ems", station_or_app_config)
        return cls.from_raw(raw_config)

    @classmethod
    def from_raw(cls, raw_config: object) -> "EmsConfig":
        if isinstance(raw_config, cls):
            return raw_config

```

```

if hasattr(raw_config, "model_dump"):
    return cls(**raw_config.model_dump())
if hasattr(raw_config, "dict"):
    return cls(**raw_config.dict())
if isinstance(raw_config, Mapping):
    return cls(**dict(raw_config))

values: dict[str, object] = {}
for field in fields(cls):
    if hasattr(raw_config, field.name):
        values[field.name] = getattr(raw_config, field.name)
if values:
    return cls(**values)
raise ValueError("station config does not contain EMS settings")

```

```

@dataclass(frozen=True, slots=True)
class EmsHistorySummary:
    outage_minutes_last_6h: float = 0.0
    outage_minutes_last_24h: float = 0.0
    outage_count_last_24h: int = 0
    outage_count_last_72h: int = 0
    hours_since_last_outage: float | None = None
    min_soc_last_24h: float | None = None
    battery_recovered_to_full_after_last_outage: bool = True

    def __post_init__(self) -> None:
        _validate_non_negative_finite(
            "outage_minutes_last_6h",
            self.outage_minutes_last_6h,
        )
        _validate_non_negative_finite(
            "outage_minutes_last_24h",
            self.outage_minutes_last_24h,
        )
        if self.outage_count_last_24h < 0 or self.outage_count_last_72h < 0:
            raise ValueError("outage counts must be non-negative")
        if self.hours_since_last_outage is not None:
            _validate_non_negative_finite(
                "hours_since_last_outage",

```

```

        self.hours_since_last_outage,
    )
    if self.min_soc_last_24h is not None:
        _validate_percent("min_soc_last_24h", self.min_soc_last_24h)
    object.__setattr__(
        self,
        "outage_count_last_24h",
        int(self.outage_count_last_24h),
    )
    object.__setattr__(
        self,
        "outage_count_last_72h",
        int(self.outage_count_last_72h),
    )
    object.__setattr__(
        self,
        "battery_recovered_to_full_after_last_outage",
        bool(self.battery_recovered_to_full_after_last_outage),
    )

```

```
@dataclass(frozen=True, slots=True)
```

```
class EmsInput:
```

```

    timestamp: datetime
    grid_available: bool
    solar_available_power_w: float
    load_power_w: float
    battery_soc_percent: float
    battery_soh_percent: float
    battery_energy_wh: float
    battery_current_usable_capacity_wh: float
    battery_voltage_v: float
    battery_status: str
    battery_max_charge_power_w: float
    grid_was_available_previous_minute: bool | None = None
    history_summary: EmsHistorySummary | None = None
    config: EmsConfig | None = None

```

```
def __post_init__(self) -> None:
```

```
    if self.timestamp.tzinfo is None or self.timestamp.utcoffset() is None:
```

```

        raise ValueError("EMS timestamp must be timezone-aware")
    _validate_non_negative_finite(
        "solar_available_power_w",
        self.solar_available_power_w,
    )
    _validate_non_negative_finite("load_power_w", self.load_power_w)
    _validate_percent("battery_soc_percent", self.battery_soc_percent)
    _validate_percent("battery_soh_percent", self.battery_soh_percent)
    _validate_non_negative_finite("battery_energy_wh", self.battery_energy_wh)
    _validate_non_negative_finite(
        "battery_current_usable_capacity_wh",
        self.battery_current_usable_capacity_wh,
    )
    _validate_non_negative_finite("battery_voltage_v", self.battery_voltage_v)
    _validate_non_negative_finite(
        "battery_max_charge_power_w",
        self.battery_max_charge_power_w,
    )
    object.__setattr__(self, "grid_available", bool(self.grid_available))
    object.__setattr__(
        self,
        "battery_status",
        str(self.battery_status).strip().lower(),
    )
    if self.history_summary is None:
        object.__setattr__(self, "history_summary", EmsHistorySummary())
    if self.config is not None and not isinstance(self.config, EmsConfig):
        object.__setattr__(self, "config", EmsConfig.from_raw(self.config))

```

```
@dataclass(frozen=True, slots=True)
```

```
class EmsDecision:
```

```

    selected_mode: EmsMode
    auto_risk_score: int
    reason: str
    effective_served_load_w: float
    inverter_output_enabled: bool
    battery_provides_energy: bool
    requested_battery_discharge_energy_wh_last_minute: float
    requested_charge_power_w: float

```

```

solar_to_load_w: float
solar_to_battery_w: float
grid_to_load_w: float
grid_to_battery_w: float
battery_to_load_w: float
protection_active: bool
cheap_tariff_active: bool
flags: tuple[str, ...] = ()

```

```

class EmsDecisionEngine:
    def __init__(self, config: EmsConfig | object | None = None) -> None:
        self.config = EmsConfig() if config is None else EmsConfig.from_raw(config)

    def decide(self, ems_input: EmsInput) -> EmsDecision:
        config = ems_input.config or self.config
        cheap_tariff_active = is_cheap_tariff_active(ems_input.timestamp, config)
        auto_risk_score = calculate_auto_risk_score(ems_input, config)

        if not ems_input.grid_available:
            return _decide_outage(
                ems_input,
                config,
                auto_risk_score=auto_risk_score,
                cheap_tariff_active=cheap_tariff_active,
            )

        selected_mode = _selected_mode(config, ems_input, auto_risk_score,
cheap_tariff_active)
        return _decide_grid_available(
            ems_input,
            config,
            selected_mode=selected_mode,
            auto_risk_score=auto_risk_score,
            cheap_tariff_active=cheap_tariff_active,
        )

    def decide_ems(ems_input: EmsInput, config: EmsConfig | object | None = None) ->
EmsDecision:

```

```

return EmsDecisionEngine(config).decide(ems_input)

def is_cheap_tariff_active(timestamp: datetime, config: EmsConfig) -> bool:
    current = timestamp.timetz().replace(tzinfo=None)
    start = _parse_clock_time(config.cheap_tariff_start)
    end = _parse_clock_time(config.cheap_tariff_end)
    if start < end:
        return start <= current < end
    return current >= start or current < end

def calculate_auto_risk_score(ems_input: EmsInput, config: EmsConfig) -> int:
    history = ems_input.history_summary or EmsHistorySummary()
    score = 0.0

    if not ems_input.grid_available:
        score += 40.0
    if history.outage_minutes_last_6h > 0.0 or _hours_since_at_most(history, 6.0):
        score += 35.0
    if history.outage_minutes_last_24h > 240.0:
        score += 25.0
    if history.outage_count_last_24h >= 2:
        score += 20.0
    elif history.outage_count_last_24h == 1:
        score += 8.0
    if history.outage_count_last_72h >= 5:
        score += 25.0
    elif history.outage_count_last_72h >= 3:
        score += 15.0

    critical_soc_seen = (
        history.min_soc_last_24h is not None
        and history.min_soc_last_24h < config.critical_soc_percent
    )
    reserve_soc_seen = (
        history.min_soc_last_24h is not None
        and history.min_soc_last_24h < config.reserve_soc_percent
    )
    if critical_soc_seen:

```

```

        score += 60.0
    elif reserve_soc_seen:
        score += 20.0

    recent_outage_seen = (
        history.outage_minutes_last_6h > 0.0
        or history.outage_minutes_last_24h > 0.0
        or history.outage_count_last_24h > 0
        or history.outage_count_last_72h > 0
        or _hours_since_at_most(history, 72.0)
    )
    if recent_outage_seen and not
history.battery_recovered_to_full_after_last_outage:
        score += 20.0

    if ems_input.battery_soc_percent <= config.critical_soc_percent:
        score += 35.0
    elif ems_input.battery_soc_percent < config.reserve_soc_percent:
        score += 15.0

    if ems_input.grid_available and not critical_soc_seen:
        if history.hours_since_last_outage is None and not recent_outage_seen:
            score -= 35.0
        elif history.hours_since_last_outage is not None:
            if history.hours_since_last_outage >= 12.0:
                score -= 20.0
            if history.hours_since_last_outage >= 24.0:
                score -= 15.0
            if history.hours_since_last_outage >= 72.0:
                score -= 15.0
        if (
            history.outage_minutes_last_24h == 0.0
            and history.outage_count_last_24h == 0
            and history.outage_count_last_72h == 0
        ):
            score -= 10.0

    if ems_input.battery_soc_percent >= config.backup_target_soc_percent - 0.1:
        score -= 15.0
    elif ems_input.battery_soc_percent >= config.normal_target_soc_percent:

```

```

        score -= 10.0

    if critical_soc_seen:
        score = max(score, 80.0)

    return int(round(_clamp(score, 0.0, 100.0)))

def _selected_mode(
    config: EmsConfig,
    ems_input: EmsInput,
    auto_risk_score: int,
    cheap_tariff_active: bool,
) -> EmsMode:
    if config.mode != EmsMode.AUTO:
        return config.mode
    if auto_risk_score <= 20:
        if (
            cheap_tariff_active
            and config.allow_grid_charging
            and ems_input.battery_soc_percent < config.backup_target_soc_percent
        ):
            return EmsMode.GRID_PRIORITY
        return EmsMode.SELF_CONSUMPTION
    if auto_risk_score <= 50:
        return EmsMode.SOLAR_PRIORITY
    if auto_risk_score <= 75:
        return EmsMode.BACKUP_RESERVE
    if (
        config.allow_grid_charging
        and ems_input.battery_soc_percent < config.backup_target_soc_percent
    ):
        return EmsMode.FORCE_CHARGE
    return EmsMode.BACKUP_RESERVE

def _decide_grid_available(
    ems_input: EmsInput,
    config: EmsConfig,
    *,

```

```

    selected_mode: EmsMode,
    auto_risk_score: int,
    cheap_tariff_active: bool,
) -> EmsDecision:
    load_power_w = ems_input.load_power_w
    solar_power_w = ems_input.solar_available_power_w
    flags: list[str] = []
    battery_to_load_w = 0.0

    if _recent_post_outage(ems_input, config):
        flags.append(EmsMode.POST_OUTAGE_RECOVERY.value)

    if selected_mode in {EmsMode.BACKUP_RESERVE, EmsMode.FORCE_CHARGE}:
        solar_to_load_w = 0.0
        grid_to_load_w = load_power_w
        solar_surplus_w = solar_power_w
    else:
        solar_to_load_w = min(solar_power_w, load_power_w)
        remaining_load_w = load_power_w - solar_to_load_w
        if selected_mode == EmsMode.SELF_CONSUMPTION:
            battery_to_load_w = _battery_to_load_power(
                ems_input,
                config,
                remaining_load_w,
                config.reserve_soc_percent,
            )
        elif selected_mode == EmsMode.BATTERY_PRIORITY:
            battery_to_load_w = _battery_to_load_power(
                ems_input,
                config,
                remaining_load_w,
                config.critical_soc_percent,
            )
        elif selected_mode not in {EmsMode.GRID_PRIORITY, EmsMode.SOLAR_PRIORITY}:
            selected_mode = EmsMode.GRID_PRIORITY
            grid_to_load_w = max(0.0, remaining_load_w - battery_to_load_w)
            solar_surplus_w = max(0.0, solar_power_w - solar_to_load_w)

    if (
        battery_to_load_w == 0.0

```

```

        and selected_mode in {EmsMode.SELF_CONSUMPTION, EmsMode.BATTERY_PRIORITY}
        and _battery_discharge_block_reason(ems_input, config) is not None
    ):
        flags.append(EmsMode.BATTERY_PROTECTION.value)

target_soc = _target_soc_for_mode(selected_mode, config)
if (
    config.mode == EmsMode.AUTO
    and auto_risk_score <= 20
    and cheap_tariff_active
    and config.allow_grid_charging
):
    target_soc = config.backup_target_soc_percent
grid_charging_allowed = _grid_charging_allowed(
    selected_mode,
    ems_input,
    config,
    cheap_tariff_active,
)
solar_to_battery_w, grid_to_battery_w = _charge_allocation(
    ems_input,
    config,
    target_soc_percent=target_soc,
    solar_surplus_w=solar_surplus_w,
    grid_charging_allowed=grid_charging_allowed,
)

requested_charge_power_w = solar_to_battery_w + grid_to_battery_w
reason = _grid_reason(
    selected_mode,
    requested_charge_power_w=requested_charge_power_w,
    grid_charging_allowed=grid_charging_allowed,
    cheap_tariff_active=cheap_tariff_active,
)
return _decision(
    selected_mode=selected_mode,
    auto_risk_score=auto_risk_score,
    reason=reason,
    effective_served_load_w=load_power_w,
    inverter_output_enabled=True,

```

```

        battery_to_load_w=battery_to_load_w,
        requested_charge_power_w=requested_charge_power_w,
        solar_to_load_w=solar_to_load_w,
        solar_to_battery_w=solar_to_battery_w,
        grid_to_load_w=grid_to_load_w,
        grid_to_battery_w=grid_to_battery_w,
        protection_active=False,
        cheap_tariff_active=cheap_tariff_active,
        flags=tuple(flags),
    )

def _decide_outage(
    ems_input: EmsInput,
    config: EmsConfig,
    *,
    auto_risk_score: int,
    cheap_tariff_active: bool,
) -> EmsDecision:
    load_power_w = ems_input.load_power_w
    if load_power_w <= 0.0:
        solar_to_battery_w, _grid_to_battery_w = _charge_allocation(
            ems_input,
            config,
            target_soc_percent=config.backup_target_soc_percent,
            solar_surplus_w=ems_input.solar_available_power_w,
            grid_charging_allowed=False,
        )
    return _decision(
        selected_mode=EmsMode.OUTAGE_MODE,
        auto_risk_score=auto_risk_score,
        reason="outage mode: no current load; solar may charge the battery",
        effective_served_load_w=0.0,
        inverter_output_enabled=True,
        battery_to_load_w=0.0,
        requested_charge_power_w=solar_to_battery_w,
        solar_to_load_w=0.0,
        solar_to_battery_w=solar_to_battery_w,
        grid_to_load_w=0.0,
        grid_to_battery_w=0.0,
    )

```

```

        protection_active=False,
        cheap_tariff_active=cheap_tariff_active,
        flags=(EmsMode.OUTAGE_MODE.value,),
    )

if load_power_w > config.inverter_output_limit_w:
    return _shutdown_decision(
        ems_input,
        config,
        auto_risk_score=auto_risk_score,
        cheap_tariff_active=cheap_tariff_active,
        reason=(
            "inverter protection shutdown: outage load exceeds inverter "
            "output limit"
        ),
        flags=("outage_load_above_inverter_limit",),
    )

solar_to_load_w = min(ems_input.solar_available_power_w, load_power_w)
remaining_load_w = load_power_w - solar_to_load_w
battery_to_load_w = 0.0
if remaining_load_w > 0.0:
    block_reason = _battery_discharge_block_reason(ems_input, config)
    if block_reason is not None:
        return _shutdown_decision(
            ems_input,
            config,
            auto_risk_score=auto_risk_score,
            cheap_tariff_active=cheap_tariff_active,
            reason=f"battery protection shutdown: {block_reason}",
            flags=(EmsMode.BATTERY_PROTECTION.value,),
        )
    available_battery_power_w = _available_battery_output_power_w(
        ems_input,
        config,
        config.critical_soc_percent,
    )
    if remaining_load_w > available_battery_power_w + 1e-9:
        return _shutdown_decision(
            ems_input,

```

```

        config,
        auto_risk_score=auto_risk_score,
        cheap_tariff_active=cheap_tariff_active,
        reason=(
            "inverter protection shutdown: battery/inverter capability "
            "is insufficient for outage load"
        ),
        flags=("battery_capability_insufficient",),
    )
    battery_to_load_w = remaining_load_w

solar_surplus_w = max(0.0, ems_input.solar_available_power_w - solar_to_load_w)
solar_to_battery_w, _grid_to_battery_w = _charge_allocation(
    ems_input,
    config,
    target_soc_percent=config.backup_target_soc_percent,
    solar_surplus_w=solar_surplus_w,
    grid_charging_allowed=False,
)
requested_charge_power_w = solar_to_battery_w
return _decision(
    selected_mode=EmsMode.OUTAGE_MODE,
    auto_risk_score=auto_risk_score,
    reason=_outage_reason(battery_to_load_w),
    effective_served_load_w=load_power_w,
    inverter_output_enabled=True,
    battery_to_load_w=battery_to_load_w,
    requested_charge_power_w=requested_charge_power_w,
    solar_to_load_w=solar_to_load_w,
    solar_to_battery_w=solar_to_battery_w,
    grid_to_load_w=0.0,
    grid_to_battery_w=0.0,
    protection_active=False,
    cheap_tariff_active=cheap_tariff_active,
    flags=(EmsMode.OUTAGE_MODE.value,),
)

```

```

def _outage_reason(battery_to_load_w: float) -> str:
    if battery_to_load_w > 0.0:

```

```

        return "outage mode: solar serves load first and battery covers the rest"
    return "outage mode: solar fully serves load; battery discharge is disabled"

```

```

def _shutdown_decision(
    ems_input: EmsInput,
    config: EmsConfig,
    *,
    auto_risk_score: int,
    cheap_tariff_active: bool,
    reason: str,
    flags: tuple[str, ...],
) -> EmsDecision:
    solar_to_battery_w, _grid_to_battery_w = _charge_allocation(
        ems_input,
        config,
        target_soc_percent=config.backup_target_soc_percent,
        solar_surplus_w=ems_input.solar_available_power_w,
        grid_charging_allowed=False,
    )
    return _decision(
        selected_mode=EmsMode.INVERTER_PROTECTION_SHUTDOWN,
        auto_risk_score=auto_risk_score,
        reason=reason,
        effective_served_load_w=0.0,
        inverter_output_enabled=False,
        battery_to_load_w=0.0,
        requested_charge_power_w=solar_to_battery_w,
        solar_to_load_w=0.0,
        solar_to_battery_w=solar_to_battery_w,
        grid_to_load_w=0.0,
        grid_to_battery_w=0.0,
        protection_active=True,
        cheap_tariff_active=cheap_tariff_active,
        flags=(EmsMode.INVERTER_PROTECTION_SHUTDOWN.value, *flags),
    )

```

```

def _decision(
    *,

```

```

selected_mode: EmsMode,
auto_risk_score: int,
reason: str,
effective_served_load_w: float,
inverter_output_enabled: bool,
battery_to_load_w: float,
requested_charge_power_w: float,
solar_to_load_w: float,
solar_to_battery_w: float,
grid_to_load_w: float,
grid_to_battery_w: float,
protection_active: bool,
cheap_tariff_active: bool,
flags: tuple[str, ...],
) -> EmsDecision:
    battery_to_load_w = _round_power(battery_to_load_w)
    requested_discharge_wh = battery_to_load_w / 60.0 if battery_to_load_w > 0.0 else
0.0

    return EmsDecision(
        selected_mode=selected_mode,
        auto_risk_score=auto_risk_score,
        reason=reason,
        effective_served_load_w=_round_power(effective_served_load_w),
        inverter_output_enabled=inverter_output_enabled,
        battery_provides_energy=battery_to_load_w > 0.0,
        requested_battery_discharge_energy_wh_last_minute=round(
            requested_discharge_wh,
            6,
        ),
        requested_charge_power_w=_round_power(requested_charge_power_w),
        solar_to_load_w=_round_power(solar_to_load_w),
        solar_to_battery_w=_round_power(solar_to_battery_w),
        grid_to_load_w=_round_power(grid_to_load_w),
        grid_to_battery_w=_round_power(grid_to_battery_w),
        battery_to_load_w=battery_to_load_w,
        protection_active=protection_active,
        cheap_tariff_active=cheap_tariff_active,
        flags=flags,
    )

```

```

def _grid_reason(
    selected_mode: EmsMode,
    *,
    requested_charge_power_w: float,
    grid_charging_allowed: bool,
    cheap_tariff_active: bool,
) -> str:
    reason_by_mode = {
        EmsMode.GRID_PRIORITY: "grid priority: grid covers load while battery
discharge is disabled",
        EmsMode.SOLAR_PRIORITY: "solar priority: solar serves load first and grid
covers the rest",
        EmsMode.SELF_CONSUMPTION: "self-consumption: solar and battery reduce grid
use while reserve is protected",
        EmsMode.BATTERY_PRIORITY: "battery priority: battery is used aggressively
above critical reserve",
        EmsMode.BACKUP_RESERVE: "backup reserve: battery discharge is disabled while
grid exists",
        EmsMode.FORCE_CHARGE: "force charge: battery discharge is disabled and
charging is prioritized",
    }
    reason = reason_by_mode.get(selected_mode, "grid available: conservative
routing")
    if requested_charge_power_w > 0.0:
        source = "grid/solar" if grid_charging_allowed else "solar"
        if cheap_tariff_active and grid_charging_allowed:
            source = "cheap-tariff grid/solar"
        reason = f"{reason}; requesting {source} battery charging"
    return reason


def _target_soc_for_mode(selected_mode: EmsMode, config: EmsConfig) -> float:
    if selected_mode in {EmsMode.BACKUP_RESERVE, EmsMode.FORCE_CHARGE}:
        return config.backup_target_soc_percent
    if selected_mode == EmsMode.BATTERY_PRIORITY:
        return config.reserve_soc_percent
    return config.normal_target_soc_percent

```

```

def _grid_charging_allowed(
    selected_mode: EmsMode,
    ems_input: EmsInput,
    config: EmsConfig,
    cheap_tariff_active: bool,
) -> bool:
    if not config.allow_grid_charging:
        return False
    if selected_mode in {
        EmsMode.GRID_PRIORITY,
        EmsMode.BACKUP_RESERVE,
        EmsMode.FORCE_CHARGE,
    }:
        return True
    if cheap_tariff_active and selected_mode == EmsMode.SOLAR_PRIORITY:
        return True
    if ems_input.battery_soc_percent < config.reserve_soc_percent:
        return True
    return False

def _charge_allocation(
    ems_input: EmsInput,
    config: EmsConfig,
    *,
    target_soc_percent: float,
    solar_surplus_w: float,
    grid_charging_allowed: bool,
) -> tuple[float, float]:
    desired_charge_power_w = _charge_power_to_target(
        ems_input,
        target_soc_percent,
    )
    solar_to_battery_w = min(
        max(0.0, solar_surplus_w),
        desired_charge_power_w,
    )
    remaining_charge_w = max(0.0, desired_charge_power_w - solar_to_battery_w)
    grid_to_battery_w = remaining_charge_w if grid_charging_allowed else 0.0
    if not config.allow_grid_charging:

```

```

        grid_to_battery_w = 0.0
    return solar_to_battery_w, grid_to_battery_w

def _charge_power_to_target(
    ems_input: EmsInput,
    target_soc_percent: float,
) -> float:
    if not ems_input.battery_status:
        return 0.0
    if ems_input.battery_current_usable_capacity_wh <= 0.0:
        return 0.0
    if ems_input.battery_max_charge_power_w <= 0.0:
        return 0.0
    target_energy_wh = (
        ems_input.battery_current_usable_capacity_wh
        * _clamp(target_soc_percent, 0.0, 100.0)
        / 100.0
    )
    headroom_wh = max(0.0, target_energy_wh - ems_input.battery_energy_wh)
    if headroom_wh <= 0.0:
        return 0.0
    return min(ems_input.battery_max_charge_power_w, headroom_wh * 60.0)

def _battery_to_load_power(
    ems_input: EmsInput,
    config: EmsConfig,
    remaining_load_w: float,
    floor_soc_percent: float,
) -> float:
    if remaining_load_w <= 0.0:
        return 0.0
    if _battery_discharge_block_reason(ems_input, config) is not None:
        return 0.0
    available_power_w = _available_battery_output_power_w(
        ems_input,
        config,
        floor_soc_percent,
    )

```

```

    return min(remaining_load_w, available_power_w)

def _available_battery_output_power_w(
    ems_input: EmsInput,
    config: EmsConfig,
    floor_soc_percent: float,
) -> float:
    if _battery_discharge_block_reason(ems_input, config) is not None:
        return 0.0
    floor_energy_wh = 0.0
    if ems_input.battery_current_usable_capacity_wh > 0.0:
        floor_energy_wh = (
            ems_input.battery_current_usable_capacity_wh
            * _clamp(floor_soc_percent, 0.0, 100.0)
            / 100.0
        )
    available_energy_wh = max(0.0, ems_input.battery_energy_wh - floor_energy_wh)
    return min(config.inverter_output_limit_w, available_energy_wh * 60.0)

def _battery_discharge_block_reason(
    ems_input: EmsInput,
    config: EmsConfig,
) -> str | None:
    if not ems_input.battery_status:
        return "battery status is unavailable"
    if ems_input.battery_status == "empty":
        return "battery status is empty"
    if ems_input.battery_current_usable_capacity_wh <= 0.0:
        return "battery usable capacity is unavailable"
    if ems_input.battery_energy_wh <= 0.0:
        return "battery has no usable energy"
    if ems_input.battery_soc_percent <= config.critical_soc_percent:
        return "battery SoC is at or below critical threshold"
    return None

def _recent_post_outage(ems_input: EmsInput, config: EmsConfig) -> bool:
    if ems_input.grid_was_available_previous_minute is False:

```

```

        return True
    history = ems_input.history_summary or EmsHistorySummary()
    if history.hours_since_last_outage is None:
        return False
    recovery_hours = config.recent_outage_recovery_minutes / 60.0
    return history.hours_since_last_outage <= recovery_hours

def _hours_since_at_most(history: EmsHistorySummary, hours: float) -> bool:
    return (
        history.hours_since_last_outage is not None
        and history.hours_since_last_outage <= hours
    )

def _normalize_clock_time(value: object) -> str:
    parsed = _parse_clock_time(str(value))
    return f"{parsed.hour:02d}:{parsed.minute:02d}"

def _parse_clock_time(value: str) -> time:
    parts = value.split(":")
    if len(parts) != 2:
        raise ValueError("EMS tariff time must use HH:MM")
    try:
        hour = int(parts[0])
        minute = int(parts[1])
    except ValueError as exc:
        raise ValueError("EMS tariff time must use HH:MM") from exc
    if not (0 <= hour <= 23 and 0 <= minute <= 59):
        raise ValueError("EMS tariff time must use HH:MM")
    return time(hour, minute)

def _validate_positive_finite(name: str, value: float) -> None:
    _validate_non_negative_finite(name, value)
    if float(value) <= 0.0:
        raise ValueError(f"{name} must be positive")

```

```

def _validate_non_negative_finite(name: str, value: float) -> None:
    try:
        numeric = float(value)
    except (TypeError, ValueError) as exc:
        raise ValueError(f"{name} must be a finite number") from exc
    if not isfinite(numeric) or numeric < 0.0:
        raise ValueError(f"{name} must be a non-negative finite number")

def _validate_percent(name: str, value: float) -> None:
    try:
        numeric = float(value)
    except (TypeError, ValueError) as exc:
        raise ValueError(f"{name} must be a finite percent") from exc
    if not isfinite(numeric) or not (0.0 <= numeric <= 100.0):
        raise ValueError(f"{name} must be between 0 and 100")

def _clamp(value: float, minimum: float, maximum: float) -> float:
    return max(minimum, min(maximum, value))

def _round_power(value: float) -> float:
    return round(max(0.0, value), 6)

```

ДОДАТОК Б Презентація



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

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ПРОГРАМНИЙ МОДУЛЬ АВТОМАТИЗАЦІЇ РЕЖИМІВ ЗАРЯДКИ АКУМУЛЯТОРНИХ БАТАРЕЙ ЛАБОРАТОРНОГО СТЕНДУ З СОНЯЧНОЮ ГЕНЕРАЦІЄЮ ЕНЕРГІЇ

виконав: студент групи ТВ-22 Полупан Михайло Валерійович

керівник: доктор технічних наук, професор кафедри ІПЗЕ Федорова Н. В.

2026

Актуальність теми

Актуальність роботи зумовлена потребою у стійких локальних енергетичних системах в умовах регулярних пошкоджень енергетичної інфраструктури та нестабільного електропостачання. Для систем із сонячною генерацією та акумуляторними батареями важливо не лише виробляти й накопичувати енергію, а й автоматично обирати оптимальний режим її розподілу. Це дозволяє підвищити ефективність використання обладнання, зменшити втрати та забезпечити більший резерв енергії без постійного втручання користувача.

	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00
Понеділок	✗			✗	✗	✗	✗						✗	✗	✗	✗				
Вівторок	✗						✗	✗	✗	✗						✗	✗	✗	✗	
Середа	✗	✗	✗	✗						✗	✗	✗	✗						✗	✗
Четвер				✗	✗	✗	✗						✗	✗	✗	✗				
П'ятниця	✗						✗	✗	✗	✗						✗	✗	✗	✗	
Субота	✗	✗	✗	✗						✗	✗	✗	✗						✗	✗
Неділя				✗	✗	✗	✗						✗	✗	✗	✗				



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

2

Мета та постановка задачі

Мета: розробка програмного забезпечення для моделювання роботи лабораторного стенду SmartEnergy Lab та автоматизації режимів зарядки акумуляторів.

Завдання:

- провести аналіз існуючих підходів до моделювання сонячної генерації, акумуляторів та EMS;
- Розробити модулі симуляції сонячної генерації, електромережі, навантаження та акумулятора;
- Розробити алгоритм автоматичного вибору режимів зарядки акумуляторів;
- розробити API, базу даних і вебінтерфейс для візуалізації та керування системою;
- провести тестування програмного модуля.



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

3

Аналіз існуючих рішень

Напрямок	Існуючі підходи	Обмеження	Обраний підхід
Сонячна генерація	SAM, pvlb, PV-моделі	Надмірна деталізація для лабораторного стенду	Спрощена модель з урахуванням положення Сонця, погоди та параметрів панелей
Акумулятор	SoC/SoH, BMS, фізико-хімічні моделі	Складні моделі потребують великої кількості параметрів	Модель заряду, розряду та безпечних меж роботи
Керування енергією	EMS, rule-based, прогнозні алгоритми	Прогнозні підходи залежать від якості вхідних даних	Rule-based EMS-логіка для вибору режиму зарядки
Електромережа	Моделі стійкості та дефіциту енергосистеми	Працюють на рівні великих систем, не стенду	Сценарна модель: нормальна робота, обмеження, відключення, відновлення
Споживання енергії	Фіксовані профілі, end-use моделі, стохастичні підходи	Фіксовані профілі не відображають випадкові піки навантаження	Стохастична модель лабораторного навантаження з урахуванням занять, людей і стану системи



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

4

Програмне забезпечення та інструменти розробки

Backend:

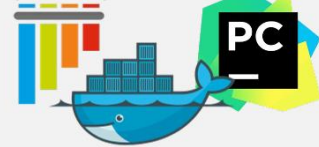


- Python
- FastAPI
- SQLite
- React
- Vite
- Recharts
- Docker
- Pylab
- Open-Meteo
- pytest
- Uvicorn
- Pydantic
- Pandas
- NumPy
- Matplotlib
- Lucide
- React
- YAML

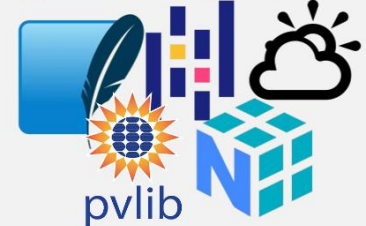
Frontend:



Тестування, розробка та розгортання:



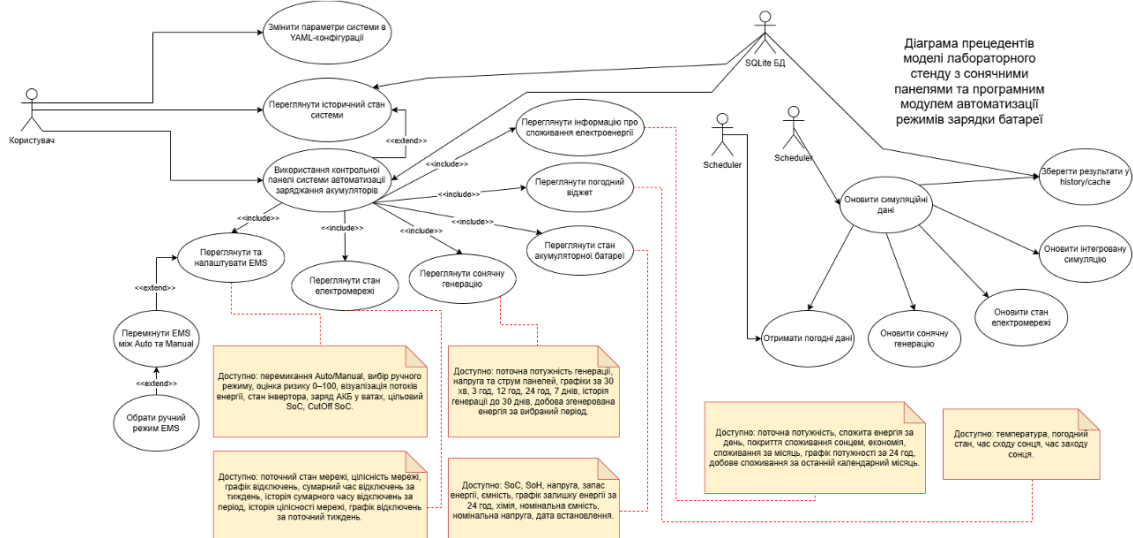
Дані та симуляція:



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

5

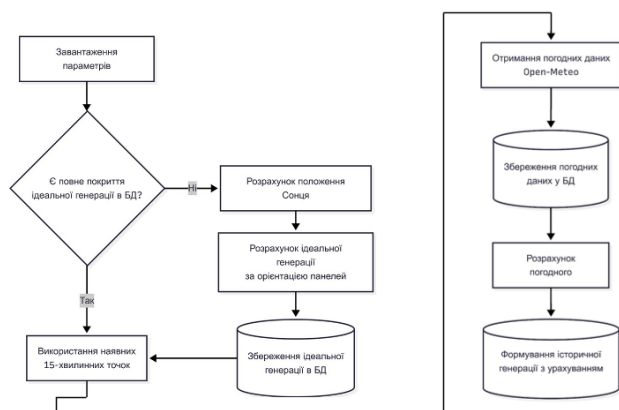
Реалізація програмного забезпечення.



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

6

Алгоритм роботи програмного модуля сонячної генерації



Модуль сонячної генерації формує історичні та прогнозовані значення потужності сонячних панелей. Для розрахунку враховуються параметри станції, географічне положення, час доби, сезонність, положення Сонця та погодний коефіцієнт.

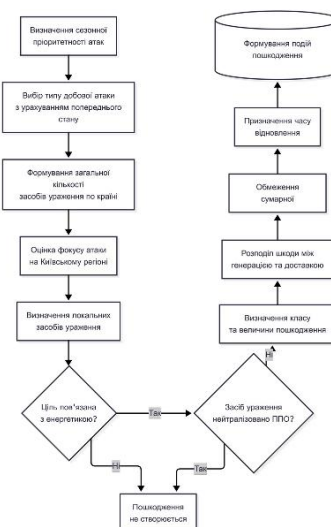
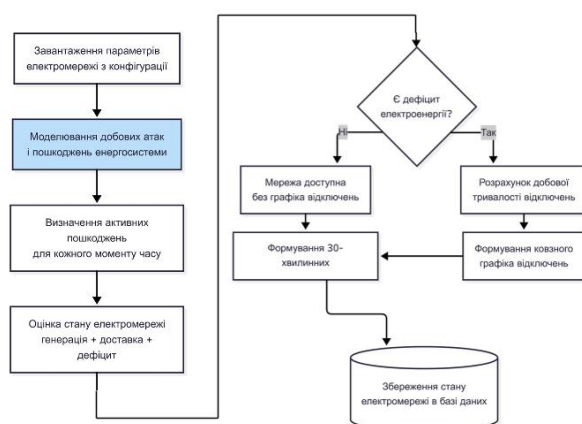
Спочатку модуль визначає ідеальну генерацію для заданої конфігурації панелей, а потім коригує її за погодними умовами. Отримані значення використовуються іншими модулями системи та EMS для розрахунку доступної енергії й вибору режиму заряджання.



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

7

Алгоритм роботи програмного модуля електромережі



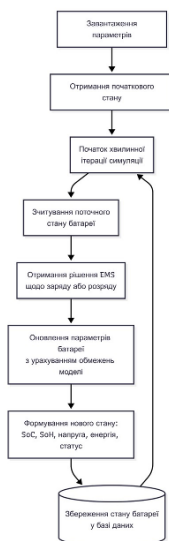
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

8

Алгоритм роботи програмного модуля батареї

Модуль акумуляторної батареї моделює стан накопичувача енергії та його основні обмеження: тип батареї, напругу, ємність, допустиму глибину розряду, ефективність і максимальну потужність заряджання.

На кожному кроці симуляції модуль отримує рішення EMS та оновлює SoC, SoH, доступну енергію, напругу й статус батареї. Це дозволяє EMS враховувати реальні межі заряду, розряду та захисту накопичувача.



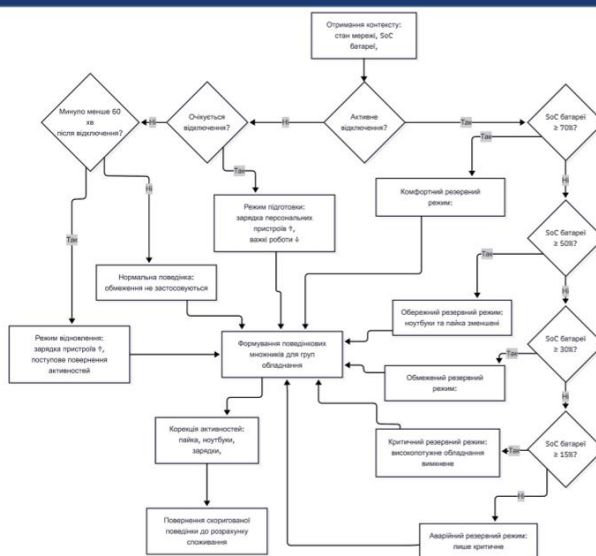
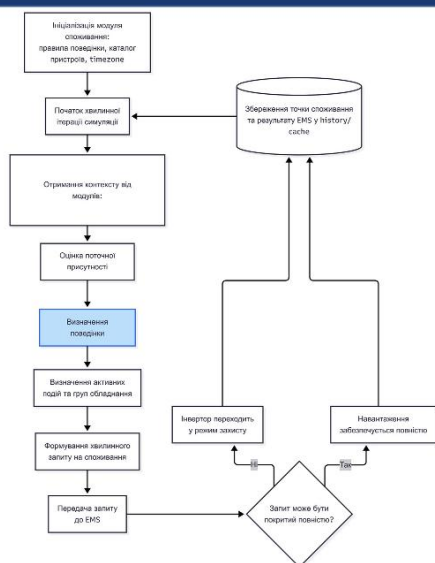
Параметр	Значення	Пояснення
Тип батареї	lead_acid	Тип акумулятора, заданий у конфігурації стенду
Номинальна напруга	12 В	Базова напруга акумуляторної батареї
Ємність	200 А-год	Конфігураційна ємність акумулятора
Номинальна енергія	2400 Вт-год	Розраховується як 12 В × 200 А-год
Корисна частка DoD	0,5	Для свинцево-кислотної батареї використовується не вся номінальна ємність
Робочий коефіцієнт DoD	0,8	Додаткове обмеження експлуатаційної глибини розряду
Ефективна корисна ємність	960 Вт-год	2400 Вт-год × 0,50 × 0,80
Початковий стан заряду	50%	Початковий стан батареї за відсутності попередніх даних
Максимальний C-rate заряду	0,20C	Обмеження зарядного струму для профілю lead_acid
Максимальна потужність заряду	480 Вт	12 В × 200 А-год × 0,20
Ефективність заряду	0,85	Частина енергії заряду, що зберігається в батареї
Ефективність розряду	0,85	Частина енергії батареї, доступна навантаженню
Ресурс до 80 % SoH	500 циклів	Опорне значення для розрахунку деградації



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

9

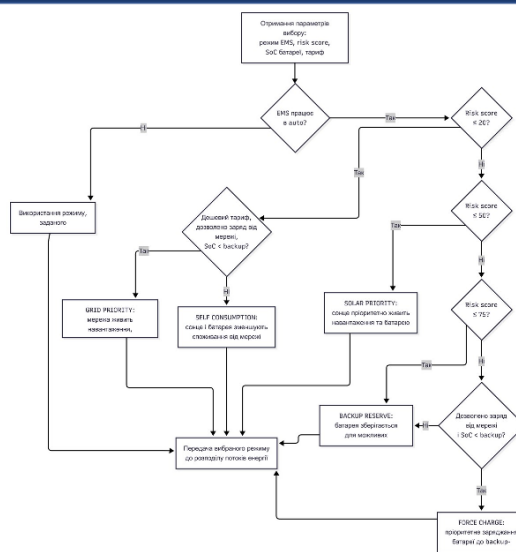
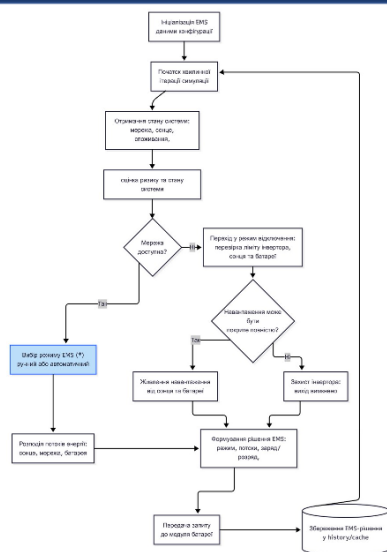
Алгоритм роботи програмного модуля навантаження



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

10

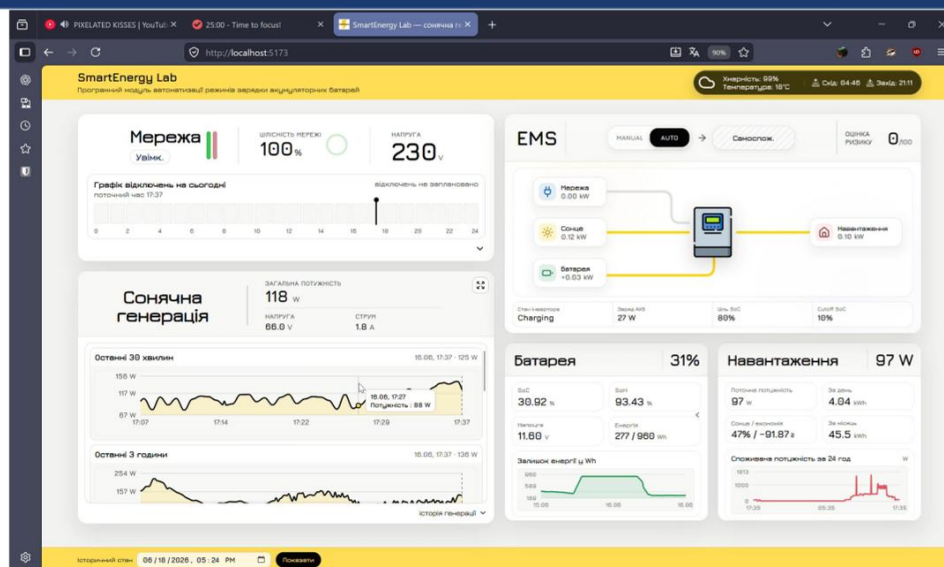
Алгоритм роботи програмного модуля EMS



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

11

Веб-інтерфейс програмної системи



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

12

Висновки

У процесі розробки було виконано наступні завдання:
проведено аналіз існуючих підходів до моделювання сонячної генерації, акумуляторів, EMS та електромереж.

Розроблено програмну основу цифрового двійника лабораторного стенду SmartEnergy Lab;
реалізовано модулі симуляції сонячної генерації, стану електромережі, акумулятора, навантаження та EMS.

Створено backend API, SQLite-сховище та React-вебінтерфейс для відображення стану системи.

Проведено тестування та підготовлено систему до демонстрації.

У результаті було створено програмний модуль, який дозволяє моделювати роботу стенду, аналізувати енергетичний баланс і автоматично визначати доцільний режим роботи системи.



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

13



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

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