МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Комп'ютерна система моніторингу та оптимізації роботи сонячних електростанцій**

Виконав: студент 2 курсу, групи 8.1212-іпз-2
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
В.А. Рижков
(ініціали та прізвище)
Керівник доцент, к.т.н., О.М.Міхайлуца
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ Алтер Віжн Груп
В. С. Тряпічко
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра Електроніки, інформаційних систем та програмного забезпечення

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)

Освітня програма ___Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т.В. Критська
"_01_" __вересня_____2023_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Рижкову Валерію Анатолійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи __Комп'ютерна система моніторингу та оптимізації роботи сонячних електростанцій

керівник роботи ___Міхайлуца Олена Миколаївна, доцент, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від 02.06. 2023 р. №597-с

2. Строк подання студентом кваліфікаційної роботи 1 грудня 2023 р.

3. Вихідні дані магістерської роботи

- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження особливостей протоколів комунікації інверторів;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

17 слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області. Огляд існуючих рішень у сфері моніторингу СЕС	02.09-07.09.23	виконано
2	Формулювання основної задачі та визначення цілей дослідження	08.09-10.09.23	виконано
3	Дослідження протоколу Modbus та інших засобів комунікації з інверторами	11.09-15.09.23	виконано
4	Розробка загальної архітектури комп'ютерної системи моніторингу	16.09-19.09.23	виконано
5	Реалізація основних функцій системи моніторингу та опитування інверторів	20.09-02.10.23	виконано
6	Тестування розроблених функцій системи під час взаємодії з різними типами інверторів	03.10-08.10.23	виконано
7	Реалізація API для взаємодії з базою даних системи	09.10-19.10.23	виконано
8	Реалізація клієнтського веб-застосунку та інтеграція з API	20.10-01.11.23	виконано
9	Тестування загальної роботи системи	02.11-09.11.23	виконано
10	Демонстрація результатів реалізації системи науковому керівнику та узгодження етапів дослідження	10.11-14.11.23	виконано
11	Проведення дослідження роботи різновидів протоколу Modbus при опитуванні інверторів	10.11-15.11.23	виконано
12	Опис результатів дослідження	16.11-19.11.23	виконано
13	Оформлення звіту	20.11-27.11.23	виконано

Студент _____ В.А. Рижков
(підпис) (прізвище та ініціали)

Керівник роботи _____ О.М. Міхайлуца
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціал)

АНОТАЦІЯ

Сторінок: 80

Рисунків: 25

Таблиць: 3

Джерел: 31

Рижков В.А. Комп'ютерна система моніторингу та оптимізації роботи сонячних електростанцій : кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник О.М. Міхайлуца. Запоріжжя : ЗНУ, 2023. 80 с.

Метою кваліфікаційної роботи є дослідження та аналіз комунікаційних протоколів Modbus RTU over TCP та Modbus TCP для надсилання та отримання інформації з інверторів, а також розробка комп'ютерної системи моніторингу роботи сонячних електростанцій з можливістю опитування різних типів інверторів.

В процесі дослідження було розглянуто особливості різновидів протоколу Modbus та проаналізовано їхнє використання для комунікації з інверторами, а також сформовано список їхніх особливостей та недоліків.

Розроблена система моніторингу сонячних електростанцій, що складається з сервісу опитування інверторів, API та клієнтського веб-застосунку, підтримує можливість опитування різних підтримуваних типів інверторів.

Ключові слова: система моніторингу, клієнт-серверний застосунок, сонячна електростанція, інвертор, протокол комунікації, Modbus, Blazor, перетворювач інтерфейсів.

SUMMARY

Pages: 80

Figures: 25

Tables: 3

Sources: 31

Ryzhkov V.A. Computer system for monitoring and optimising the operation of solar power plants : qualification work of the master of specialty 121 "Software Engineering" / Science head O.M. Mikhailutsa. Zaporizhzhia : ZNU, 2023. 80 p.

The aim of the research is to study and analyse the Modbus RTU over TCP and Modbus TCP communication protocols for sending and receiving information from inverters, as well as to develop a computer system for monitoring the operation of solar power plants with the ability to interrogate different types of inverters.

In the course of the research, the features of the Modbus protocol types were considered and their use for communication with inverters was analyzed. In addition, analogues of SPP monitoring systems were considered, and a list of their features and disadvantages was compiled. As a result, a solar power plant monitoring system was developed, consisting of an inverter polling service, an API, and a client web application. It includes the main functionalities of similar systems and supports the ability to poll various supported inverter types.

Keywords: monitoring system, client-server application, solar power plant, inverter, communication protocol, Modbus, Blazor, interface converter.

ЗМІСТ

ВСТУП

Актуальність теми

У сучасному світі все більшу популярність набирає тенденція розвитку та використання альтернативної енергетики. Концепція використання нафти, газу та вугілля в якості енергоресурсів все сильніше відходить на другий план через їхнє поступове вичерпання та не екологічність їхнього використання. На заміну таким традиційним джерелам енергії приходять відновлювані або «зелені» джерела енергії [22].

Одним із основних напрямків альтернативної енергетики є перетворення сонячної енергії в електричну. Для виконання цієї задачі використовуються спеціальні інженерні споруди — СЕС (сонячні електростанції), які набули неабиякого поширення в Україні за останні роки [20]. Вони активно використовуються в підприємствах, промисловості, приватних домогосподарствах та різних закладах в якості автономних систем для отримання електроенергії з метою забезпечення власних потреб або подальшого продажу електрики державі за «зеленим тарифом» [21].

Тож, доцільним є питання отримання інформації з СЕС з метою спостереження за процесом її роботи. Для цього використовується спеціальний пристрій, інвертор, який є невід'ємною частиною системи електростанції. Однією з функціональних можливостей такого апарату є надання різноманітної інформації щодо вироблення електроенергії на окремій СЕС при підключенні до нього через мережу [17]. Завдяки цьому є можливим не тільки спостереження за процесом роботи електростанції, але й отримання статистики вироблення електроенергії та збір аналітичних даних.

Наразі існують різні системи моніторингу СЕС від компаній-виробників інверторів та постачальників послуг встановлення електростанцій, взаємодія клієнта з якими відбувається за допомогою веб-застосунку. Але слід зауважити, що кожна така система не може взаємодіяти з інверторами від сторонніх

компаній, а за користування нею, в більшості випадків, необхідно вносити щомісячні платежі. Виходячи з цього, власнику декількох СЕС з інверторами від різних компаній необхідно користуватися різними додатками та сплачувати вдвічі більшу вартість за доступ до можливостей моніторингу.

Таким чином, актуальною проблемою є створення клієнт-серверної системи моніторингу СЕС з гнучким налаштуванням та урахуванням особливостей інверторів від популярних виробників, що буде надавати можливості зі спостереження за процесом роботи електростанції, з отримання статистики вироблення електроенергії та зі збору аналітичних даних.

Мета і завдання дослідження

Мета і завдання дослідження полягають у вивченні та аналізі комунікаційних протоколів Modbus RTU over TCP та Modbus TCP для надсилання та отримання інформації з інверторів, а також у розробці комп'ютерної системи моніторингу роботи сонячних електростанцій з можливістю опитування різних типів інверторів.

Об’єкт дослідження

Об’єктом дослідження є інвертори, які використовуються для конвертації постійного струму, що виробляється сонячними панелями, в змінний струм, а також відповідають за контроль та регулювання параметрів роботи сонячних електростанцій.

Предмет дослідження

Предметом дослідження є механізм отримання інформації з інвертора та протоколи комунікації інверторів Modbus RTU over TCP та Modbus TCP.

Методи дослідження

Для вирішення поставленої задачі використовуються наступні методи дослідження:

1. Вивчення літератури з роботи сонячних електростанцій та протоколів комунікації інверторів.
2. Аналіз протоколів комунікації інверторів.
3. Визначення параметрів налаштування інверторів та показників роботи сонячних електростанцій.
4. Проектування алгоритмів збору даних роботи сонячних електростанцій.
5. Розробка системи моніторингу сонячних електростанцій.
6. Використання програм-аналізаторів трафіку комп’ютерних мереж для виявлення особливостей роботи протоколів комунікації інверторів.

Наукова новизна одержаних результатів

Наукова новизна одержаних результатів дослідження полягає у виявлені переваг та недоліків використання різних видів протоколів комунікації інверторів Modbus RTU over TCP та Modbus TCP під час збору аналітичних даних роботи сонячних електростанцій.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у тому, що була створена комп’ютерна система моніторингу роботи сонячних електростанцій на базі протоколів Modbus RTU over TCP та Modbus TCP. Ця система забезпечує збір, обробку та відображення даних з інверторів різних моделей в режимі реального часу, що дає можливість відстежувати роботу сонячних електростанцій. Оскільки така система використовує базові протоколи комунікації інверторів та не потребує підключення додаткового обладнання для збору даних, вона є ефективною з точки зору апаратних витрат, тому її можна використовувати як дешеве та ефективне рішення для створення подібних комп'ютерних систем.

Апробація одержаних результатів

Результати роботи, викладені у кваліфікаційній роботі магістра, були опубліковані на XVI університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених — «Молода наука-2023» [25], а також на III Всеукраїнській науково-практичній конференції за участю молодих науковців — «Актуальні питання сталого науково-технічного та соціально-економічного розвитку регіонів України» [26].

Глосарій

Система моніторингу — це комплексний набір технічних засобів та програмного забезпечення, які використовуються для збору, аналізу та відображення даних щодо роботи певного технічного об'єкту або процесу.

Сонячна електростанція — це комплекс інженерних споруд та обладнання, яке призначене для перетворення сонячної енергії в електричну. Зазвичай, сонячні електростанції складаються з сонячних панелей, інверторів, трансформаторів, електричних комутаційних пристроїв, акумуляторних батарей, контрольно-вимірювальних пристроїв та систем управління.

Фотовольтаїка — це галузь енергетики, що вивчає використання сонячної енергії для генерації електричної енергії за допомогою фотоелектричного ефекту.

Інвертор — це електронний пристрій, який призначений для перетворення постійного струму на змінний струм. У контексті сонячних електростанцій, інвертор виконує функцію перетворення постійного струму, що генерується сонячними панелями, на змінний струм, який може бути використаний для живлення електроприладів та надсилається до електромережі.

Клієнт-серверний застосунок — це програмне забезпечення, яке включає дві основні складові: клієнтську та серверну. Клієнтська складова — це програма або компонент, який виконується на більш розповсюджених платформах, таких як ПК, мобільні телефони або планшети, і забезпечує взаємодію

з користувачем. Серверна складова — це програма або компонент, який виконується на сервері та забезпечує централізоване зберігання та обробку даних, а також керування доступом до цих даних.

Протокол комунікації — це набір правил, форматів і процедур, що визначають, як повинні взаємодіяти дві або більше сторін під час передачі даних між ними. Він визначає формати даних, що передаються, порядок їх взаємодії, перевірку правильності передачі даних та інші аспекти комунікації.

Modbus — це промисловий протокол передачі даних, який використовується для збору даних із сенсорів та пристроїв у промислових мережах і системах моніторингу. Цей протокол може використовуватися для передачі даних через різні фізичні канали зв'язку, такі як RS-232, RS-485 або TCP / IP. Modbus має просту структуру та легкий для реалізації код, що робить його популярним протоколом в промислових застосуваннях.

.NET — це платформа з відкритим вихідним кодом, розроблена компанією Microsoft, для створення кросплатформових десктопних, мобільних та веб-додатків.

C# — це об'єктно-орієнтована мова програмування високого рівня, яка широко використовується для розробки десктопних та веб-додатків.

Blazor — це фреймворк з відкритим вихідним кодом для розробки інтерактивних веб-додатків, який дозволяє створювати застосунки за допомогою мови програмування C# та HTML.

Blazor.Bootstrap — це бібліотека готових компонентів Blazor, створених базуючись на CSS фреймворку Bootstrap.

Фізичний інтерфейс — це обладнання для перетворення сигналів та взаємодії пристроїв, що визначає набір електричних зв’язків та характеристик сигналів на фізичному рівні.

RS-485 — це стандарт фізичного інтерфейсу для обміну даними між декількома пристроями за допомогою однієї двопровідної лінії зв’язку (витій парі) у напівдуплексному режимі.

Ethernet — це технологія пакетної передачі даних між пристроями для різних типів комп’ютерних мереж, що використовує модель TCP/IP для передачі інформації.

Дискретний вхід — це роз’єм пристрою, крізь який надходять бінарні сигнали 0 та 1, що представляють стан вимикачів, сенсорів або інших елементів. Вони використовуються для вимірювання стану різних пристроїв чи обладнання.

Дискретний вихід — це роз’єм пристрою, крізь який надсилаються бінарні сигнал, які мають два стани 0 та 1. Вони використовуються для керування різними пристроями чи обладнанням. Наприклад, вони можуть вказувати на стан реле, ввімкнення або вимкнення пристрою чи інші дії

Таблиця регістрів вводу — це таблиця даних, що визначає блоки регістрів пристрою, які містять дані, доступні для читання, але не можуть бути змінені віддалено. Ці регістри представляють собою 16-бітні слова і можуть містити інформацію, таку як вимірювальні дані, статуси, або інші значення.

Таблиця регістрів зберігання — це таблиця даних, що визначає блоки регістрів пристрою, які можуть використовуватися як для читання, так і для запису. Ці регістри призначені для передачі і зміни даних між пристроями, що використовують протокол Modbus.

Контрольна сума — це числовий параметр, який обчислюється на основі набору даних за допомогою певного математичного алгоритму, для перевірки цілісності даних при їх зберіганні або передачі.

CRC (Cyclic redundancy check) — алгоритм обчислення контрольної суми, що заснований на додаванні до оригінальних даних контрольних бітів, які обчислюються на основі математичних операцій.

Десеріалізація — це процес перетворення потоку байтів чи іншого формату даних в об'єкт або структуру даних. Цей процес використовується для відновлення збережених або переданих даних в їхню оригінальну форму.

Перетворювач інтерфейсів — це пристрій або обладнання, яке призначене для конвертації сигналів між різними типами інтерфейсів для забезпечення сумісності між пристроями чи мережами, які використовують різні стандарти або фізичні характеристики передачі даних.

ADU (Application data unit) — це одиниця даних, що передається між пристроями через мережу. Зазвичай, вона представляє собою структуроване повідомлення, що містить дані для виконання окремої команди або функції.

PDU (Protocol data unit) — це блок даних, що представляє собою базову одиницю обміну даними за певним протоколом та передається між об’єктами мережі.

MBAP (Modbus application protocol) — це частина структури повідомлення протоколу Modbus TCP, що використовується для забезпечення вкладеності Modbus протоколу в TCP-пакеті. Вона представляє собою заголовок, що знаходиться перед PDU в фреймі Modbus TCP.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ ОТРИМАННЯ ДАНИХ З СОНЯЧНИХ ЕЛЕКТРОСТАНЦІЙ

1.1 Огляд задачі з отримання даних з сонячної електростанції

При використанні сонячних електростанцій нагальною потребою є отримання даних з них для формування інформації про поточне функціонування системи, прогнозування вироблення електроенергії та фінансових прибутків, отримання статистичної інформації та відслідковування можливих проблем в їхній роботі [30].

Головний пристрій, який дозволяє проводити збір такої інформації – це інвертор. Він виступає частиною станції до якої під’єднуються дроти, проводячи постійний струм від сонячних панелей. Крім цього, він конвертує цей струм в постійний та пропускає його до загальної електромережі. Також інвертор для СЕС містить пропрієтарну операційну систему, що керує отриманням інформації щодо роботи загальної системи електростанції, до якої він під’єднаний, а також дозволяє взаємодіяти зі сторонніми пристроями за допомогою різних протоколів [19]. Зокрема, найпопулярнішим протоколом, який використовується для виконання такої задачі є Modbus та його різновиди (Modbus TCP, Modbus RTU, Modbus ASCII). Окрім цього, деякі інвертори можуть використовувати API та комунікувати з іншими пристроями за допомогою протоколу HTTP.

Основними фізичними інтерфейсами інверторів, через які відбувається передача даних, виступають RS-485, RS-232 та Ethernet [28]. Але при організації загальної мережі для опитування сонячних електростанцій доцільним є використання Ethernet, що значно спрощує її створення та підтримку. Для цього використовуються спеціальні перетворювачі інтерфейсів RS-485, RS-232 на Ethernet. Окрім цього, такий підхід дещо змінює тип зв’язку з інверторами, замінюючи використання Modbus RTU на Modbus RTU over TCP.

Отже, підсумовуючи усе вищесказане, інвертор отримує дані про загальне функціонування системи, може взаємодіяти з іншими пристроями та виступає проміжною ланкою між мережею електростанції та загальною зовнішньою електричною мережею. Саме за допомогою опитування інверторів системи моніторингу СЕС отримують дані щодо роботи сонячних електростанцій.

1.2 Проблематика отримання даних з різних типів інверторів

Основною проблемою в опитуванні інверторів від різних виробників є використання різних протоколів комунікації та розбіжності в методах отримання такої інформації.

Як було зазначено раніше, в більшості випадків, інвертори використовують протокол комунікації Modbus та його різновиди. Кожен з таких протоколів має свої особливості щодо формування повідомлень та їх отримання [15]. Також, враховуючи використання HTTP, який підтримується деякими інверторами, уніфікація засобів комунікації з такими пристроями вибудовується у більш складну проблему.

Окрім цього, згідно до специфіки Modbus, дані клієнтських пристроїв зберігаються в спеціальних комірках пам’яті. В залежності від моделей та виробників інверторів, адреси комірок з необхідними даними можуть відрізнятись, що ускладнює можливість опитування таких пристроїв [29].

Тож, наведені особливості значно ускладнюють уніфікацію методів отримання даних з різних типів інверторів та призводять до актуалізації цієї проблеми. Але слід зауважити, що існуючі системи моніторингу підтримують роботу з інверторами від окремих виробників або набором певних моделей, що деякою мірою вирішує наведену задачу [27].

1.3 Аналіз існуючих систем моніторингу СЕС

1.3.1 Рішення від компанії Fronius

Fronius — це австрійська компанія, яка спеціалізується у галузях зварювальних технологій, фотовольтаїки та систем для заряджання акумуляторних батарей. Компанія надає свої послуги та рішення з енергозбереження, автономних систем живлення та СЕС домашнім господарствам та комерційним і промисловим підприємствам.

Для моніторингу СЕС Fronius пропонує платформу FRONIUS SOLAR.WEB для мобільних пристроїв та комп'ютерів. Користувач системи може взаємодіяти з нею за допомогою веб-застосунку або програм для різних платформ.

Застосунок FRONIUS SOLAR.WEB є безкоштовним. Він надає можливість перегляду списку підключених фотовольтаїчних систем та їхніх статусів, статистики вироблення електроенергії та обчислення прибутку з продажу виробленої енергії (див. Рис. 1.1). Окрім цього, є можливим отримання детальної інформації щодо процесу роботи окремого інвертора та огляд детального журналу помилок.

Наведена платформа отримує дані з серверів компанії, на які передається інформація з інверторів. Для налаштування роботи системи необхідно забезпечити під'єднання інверторів до мережі Інтернет та виконати конфігурацію цих пристроїв за допомогою мобільного застосунку, після чого вони будуть приєднані до загального списку електростанцій клієнта. Слід зазначити, що надходження інформації до серверів відбувається автоматично через вбудований в інвертор пристрій для передавання даних — Fronius Datamanager, який не входить до базової комплектації усіх інверторів компанії Fronius [24].

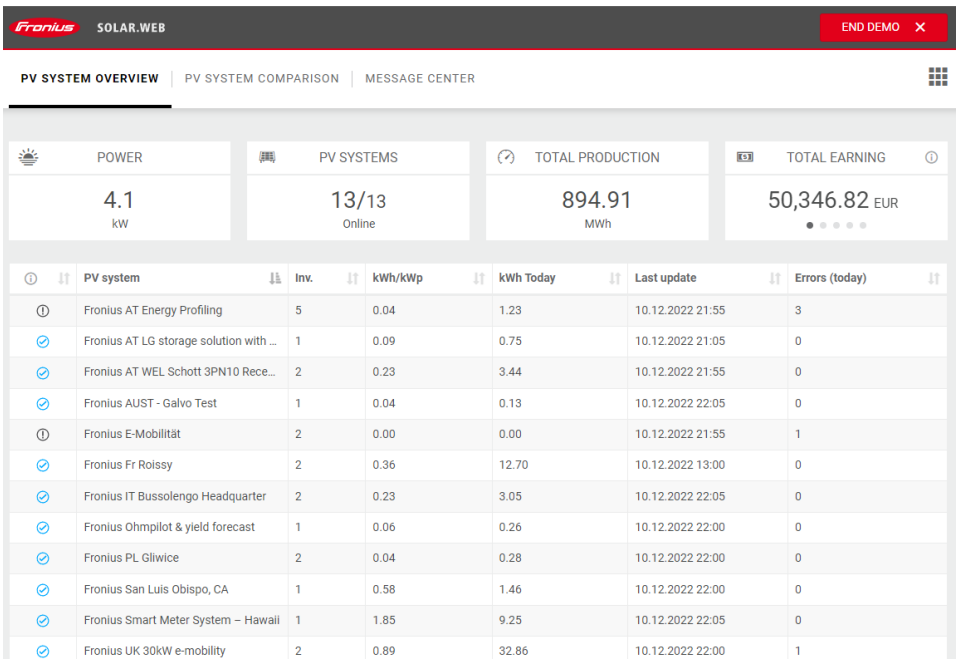

Рисунок 1.1 — Інтерфейс FRONIUS SOLAR.WEB

До переваг FRONIUS SOLAR.WEB можна віднести:

- Можливість конфігурації інвертора за допомогою мобільного застосунку.
- Надання журналу помилок інверторів.

До недоліків FRONIUS SOLAR.WEB можна віднести:

- Функціонування системи лише з інверторами від компанії Fronius.
- Необхідність у встановленні додаткового обладнання для функціонування системи.
- Складний процес налаштування інверторів для додавання СЕС у загальну систему моніторингу.

1.3.2 Рішення від компанії APsystems

APsystems — це американська компанія, що займається виробництвом мікроінверторів, накопичувачів енергії та пристроїв для різних потреб фотоелектричної індустрії. Окрім цього, вона надає передові технології з функціонування СЕС для житлових і комерційних систем [12].

Програмним рішенням для моніторингу від APsysetms виступає мобільний застосунок EMA Manager App (див. Рис. 1.2), який є повністю безкоштовним.

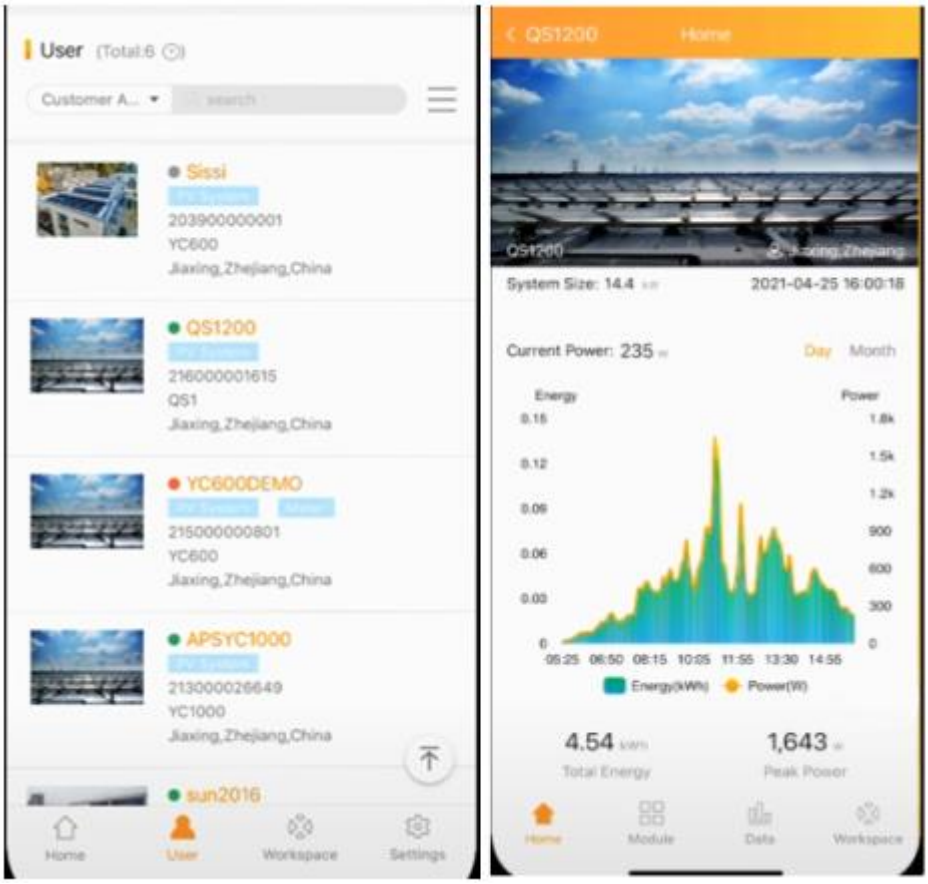

Рисунок 1.2 — Інтерфейс EMA Manager App

За допомогою цієї програми є можливим здійснювати моніторинг інверторів, під’єднаних до локальної мережі клієнта або до глобальної мережі Інтернет. EMA Manager App дозволяє переглядати список фотовольтаїчних систем, наявних у користувача в якому наведені їхні головні атрибути: статус, назва, серійний номер пристрою, адреса, невеличке зображення для більш зручної ідентифікації та інші. Застосунок дозволяє налаштовувати відображення окремих полів даних інверторів в загальному переліку. Тобто, користувач за допомогою налаштувань системи може відключати або вмикати демонстрацію певних атрибутів наявних систем в списку. Окрім цього, є можливим перегляд даних щодо роботи окремо обраного інвертора та під’єднаної до нього електростанції, які відображаються у вигляді зручних графіків. Також застосунок

надає статистичні дані для клієнтів, які включають загальну кількість встановлених інверторів і показники об’ємів виробленої енергії.

Наведений застосунок отримує інформацію про роботу СЕС клієнта з серверів APsystems, які отримують інформацію з інверторів. Налаштування такої системи моніторингу є доволі простим. Клієнту достатньо зчитати QR-код на інверторі та виконати його базове просте налаштування для того, щоб додати цей пристрій до загального списку СЕС. Однак, слід зазначити, що система працює лише з мікроінверторами від компанії APsystems.

До переваг EMA Manager App можна віднести:

- Можливість зручної конфігурації та підключення інвертора до загальної системи за допомогою мобільного застосунку.
- Можливість гнучкого налаштування відображення окремих полів даних інверторів у загальному списку.

До недоліків EMA Manager App можна віднести:

- Система працює лише з мікроінверторами від компанії APsysetms.

1.3.3 Рішення від компанії SunPower

SunPower — це американська компанія, що постачає фотогальванічні системи генерації електроенергії та продуктів для зберігання енергії в батареях, насамперед для побутових споживачів. Вона відома виробництвом високоефективних сонячних панелей, які значно перевершують конкурентні продукти за багатьма показниками [31].

Одним із продуктів, які надає SunPower є онлайн система моніторингу СЕС SunPower Monitoring System. Вона складається з двох компонентів:

1. EnergyLink Hardware — спеціальний пристрій, який збирає інформацію з інвертора та надсилає звіт назад на сервер SunPower через Інтернет.
2. EnergyLink Software — додаток порталу для клієнтів SunPower (див. Рис. 1.3).

Рисунок 1.3 — Інтерфейс EnergyLink Software

В ньому можливо відстежувати продуктивність системи, виробництво енергії, енергоспоживання, приблизну економію рахунків і показники позитивного впливу на навколишнє середовище. демонструвати економію на навколишньому середовищі, отримувати відповіді на запитання щодо підтримки тощо. Застосунок використовує графіки, які відображають поточний стан СЕС та потужність, яку ваша вона генерує за певний період часу. Якщо у системі встановлено додатковий лічильник споживання, за допомогою EnergyLink Software також можна відстежувати, скільки енергії використовується, і порівнювати ці показники з об’ємами виробленої електроенергії.

До переваг SunPower Monitoring System можна віднести:

- Простий та інтуїтивно зрозумілий інтерфейс.
- Відображення показників позитивного впливу СЕС на навколишнє середовище.

До недоліків SunPower Monitoring System можна віднести:

- Відсутність можливості перегляду роботи кожного окремого інвертора.
- Можливість функціонування системи лише при встановленні додаткового обладнання від компанії SunPower.

- Відсутність можливості самостійного додавання нових інверторів до системи моніторингу.

1.4 Висновки на основі аналізу аналогів

Виходячи з проведеного дослідження аналогів програмних продуктів моніторингу СЕС було висунуто наступні рекомендації до системи, яка б могла бути конкурентоспроможною на ринку:

1. Система повинна працювати без підключення додаткових апаратних систем опитування СЕС. Тобто, достатньою вимогою є підключення інвертора до однієї мережі з сервером, на якому буде розгорнуто програмну систему.
2. Система повинна мати опції для налаштування параметрів підключення інверторів.
3. Система повинна коректно працювати з різними типами інверторів.
4. Система повинна надавати архівну інформацію з вироблення електроенергії за тривалий час.
5. Система повинна мати журнал помилок в роботі інверторів.
6. Система повинна надавати функціонал для обчислення очікуваного прибутку з продажу виробленої енергії.
7. Система повинна відображати статистичні дані з загального вироблення електроенергії.
8. Система повинна надавати інформацію щодо роботи окремого інвертора.
9. Система повинна використовувати графіки для відображення аналітичних даних.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Протокол Modbus та бібліотеки для його використання

Для передачі інформації на підключені пристрої, інвертори використовують протокол Modbus. Modbus — це відкритий комунікаційний протокол, який базується на взаємодії «ведучий-підлеглий» (англ. «master-slave»). Він широко використовується у сфері організації зв'язку та передачі даних між електронними пристроями через інтерфейси RS-485, RS-422, RS-232 та мережі Ethernet.

За специфікацією протоколу, підлеглі пристрої мають чотири таблиці даних: дискретні входи (Discrete Input Contacts), регістри прапорів (Discrete Output Coils), регістри вводу (Analog Input Registers) та регістри зберігання (Analog Output Holding Registers) (див. Рис. 2.1) [13]. Дві з них доступні для тільки читання, а дві для читання та запису. Доступ до кожного з елементів цієї таблиці відбувається за допомогою відповідної адреси, що складається з 16 бітів.

Coil/Register Numbers	Data Addresses	Type	Table Name
1-9999	0000 to 270E	Read-Write	Discrete Output Coils
10001-19999	0000 to 270E	Read-Only	Discrete Input Contacts
30001-39999	0000 to 270E	Read-Only	Analog Input Registers
40001-49999	0000 to 270E	Read-Write	Analog Output Holding Registers

Рисунок 2.1 — Організація таблиць даних клієнтського пристрою за протоколом Modbus

Ведучі пристрої надсилають команди до підлеглих з використанням певних команд (читання або запис) в залежності від бажаного результату. Але, коди цих функцій відрізняються в залежності від комірок пам’яті до яких вони застосовуються (див. Рис. 2.2) [16]. Таким чином, сервер може надсилати наступні команди:

- 01 — Читання групи регістрів прапорів.
- 02 — Читання групи регістрів дискретних входів.
- 03 — Читання групи регістрів зберігання.
- 04 — Читання групи регістрів вводу.
- 05 — Запис одного регістру до регістрів прапорів.
- 06 — Запис одного регістру до регістрів зберігання.
- 15 — Запис групи регістрів до регістрів прапорів.
- 16 — Запис групи регістрів до регістрів зберігання.

Function Code	Action	Table Name
01 (01 hex)	Read	Discrete Output Coils
05 (05 hex)	Write single	Discrete Output Coil
15 (0F hex)	Write multiple	Discrete Output Coils
02 (02 hex)	Read	Discrete Input Contacts
04 (04 hex)	Read	Analog Input Registers
03 (03 hex)	Read	Analog Output Holding Registers
06 (06 hex)	Write single	Analog Output Holding Register
16 (10 hex)	Write multiple	Analog Output Holding Registers

Рисунок 2.2 — Функції, що використовуються серверними пристроями за протоколом Modbus

Повідомлення Modbus, що надсилається підлеглому пристрою від ведучого, містить наступні складові [11] (див. Рис. 2.3):

- Адреса підлеглого пристрою. Вона може приймати значення від 0 до 247.
- Код функції.
- Спеціальні дані, які залежать від коду функції та контрольної суми. Наприклад, при виклику команди читання групи регістрів зберігання (команда 03), це поле містить адресу комірки даних клієнта та кількість бітів для зчитування необхідного значення.
- Контрольна сума (за алгоритмом CRC).

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low | CRC Hi

Рисунок 2.3 — Схема повідомлення Modbus

Інвертори використовують два основних формати наведеного протоколу: Modbus RTU та його модифіковану версію — Modbus TCP.

Наведені протоколи використовуються для забезпечення комунікації між системами моніторингу та компонентами сонячних електростанцій. Згідно дослідження «Застосування комунікаційного протоколу Modbus для обміну даними між елементами систем автоматизації» [23], ці протоколи є стандартами промислової автоматизації та широко використовуються у сфері фотовольтаїки.

Modbus TCP — це протокол передачі даних через TCP/IP мережі. Він дозволяє здійснювати комунікацію з декількома пристроями одночасно та передавати дані у реальному часі. Цей протокол забезпечує швидку і надійну передачу даних між системою моніторингу та інверторами сонячних електростанцій. Як зазначають автори у статті «Modbus protocol performance analysis in a

variable configuration of the physical Fieldbus architecture» [4], Modbus TCP є дуже популярним протоколом у сфері сонячної енергетики завдяки своїм перевагам у сфері швидкості та надійності передачі даних.

Modbus RTU — це протокол, який використовує послідовну лінію передачі даних (RS-485 або RS-232) для комунікації між системою моніторингу та пристроями. Окрім цього, він використовує менше ресурсів мережі порівняно з Modbus TCP. Згідно дослідження «Development of a data acquisition and monitoring system based on Modbus RTU communication protocol» [6], Modbus RTU є широко застосовуваним протоколом у випадках, коли необхідно здійснювати комунікацію у мережі з обмеженими ресурсами.

Обидві версії протоколу не розрізняються за функціоналом. Основні ж відмінності між ними полягають у наступному:

1. Протокол Modbus RTU використовує лінії зв'язку RS-232 або RS-485, тоді як Modbus TCP працює у мережі Ethernet.
2. Блок даних для Modbus TCP також кодується у двійковий формат, але упаковується в пакет TCP [10] (див. Рис. 2.4).
3. Modbus TCP використовує окремий механізм перевірки цілісності пакетів, передбачений протоколом TCP.

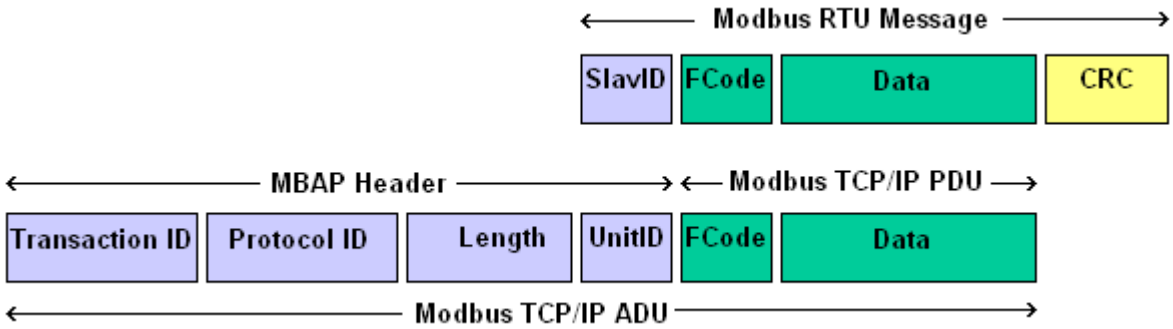

Рисунок 2.4 — Порівняння схем повідомлення Modbus RTU та Modbus TCP

Окрім цього, досить часто на практиці застосовується варіант протоколу Modbus RTU over TCP через використання перетворювачів інтерфейсів при

підключенні до інверторів. Ця модифікація протоколу дозволяє використовувати Modbus RTU в мережах TCP/IP, поєднуючи переваги обох протоколів та має наступні особливості:

- Використання TCP/IP для передачі даних, що дозволяє працювати в стандартних мережевих інфраструктурах.
- Представлення даних у вигляді TCP пакетів, в яких зберігається стандартна структура повідомлення Modbus RTU з адресою підлеглого пристрою, кодом функції, даними та контрольною сумою.

З метою реалізації роботи з протоколом Modbus, для різних мов програмування існує низка бібліотек:

- NModbus — бібліотека для C#, що включає реалізацію всіх режимів роботи з протоколом Modbus. Модель реалізації класів цієї бібліотеки дозволяє працювати з будь-яким Modbus-пристроєм, але тільки з одним, оскільки класи бібліотеки інкапсулюють порт, не дозволяючи реалізувати синхронізацію між декількома Modbus-об'єктами. Даний протокол є досить популярним у сфері розробки периферії для розумного будинку, а також інтернету речей.
- Java Modbus Library (jamod) — бібліотека для Java, яку можна використовувати для реалізації клієнтських і серверних пристроїв Modbus різних типів (ASCII, RTU (Master only), BIN, TCP, UDP). Дизайн цієї бібліотеки повністю об’єктно-орієнтований, заснований на абстракціях, які повинні підтримувати легке розуміння, можливість повторного використання та розширення застосунку.
- PyModbus — бібліотека для Python, яка містить повну реалізацію протоколу Modbus з використанням синхронного або асинхронного ядра. Вона підтримує різні режими зв'язку Modbus (TCP, RTU-OVER-TCP, UDP, Serial, TLS) та використовується без будь-яких додаткових залежностей.

2.2 Платформа .NET в якості основного інструменту для розробки серверної частини застосунку

Серверною частиною застосунку повинна виступати консольна програма. Таке рішення має низку переваг:

1. Відсутність графічного інтерфейсу значно спрощує розробку застосунку.
2. Відсутність графічного інтерфейсу дещо пришвидшує роботу серверного застосунку через використання меншого обсягу системних ресурсів, що впливає на продуктивність системи та здатність обробки запитів.
3. Відсутність залежності від графічного інтерфейсу робить застосунок більш гнучким та пристосованим для розгортання у різних системах.

Для вирішення цієї задачі доцільним є використання платформи для розробки програмних застосунків .NET. Це модульний фреймворк з відкритим вихідним кодом від компанії Microsoft. Він виступає одним із найпопулярніших інструментів для розробки мультиплатформових застосунків. Загалом фреймворк .NET представляє потужну платформу створення додатків та має наступні основні риси:

1. Підтримка кількох мов програмування. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому. NET підтримує велику кількість мов, зокрема: C#, Visual Basic, C++, F#, ClojureCLR, Eiffel, IronPython, PowerBuilder.
2. Кросплатформовість. Остання версія платформи на даний момент – .NET 7 підтримується на більшості сучасних операційних системах Windows, MacOS та Linux архітектурних версій (Arm64, x64 та x86).
3. Різноманітність технологій. Загальномовне середовище виконання CLR та базова бібліотека класів є основою цілого стеку технологій, які розробники можуть задіяти при побудові тих чи інших додатків.

4. Висока продуктивність. Згідно з рядом тестів веб-програми на .NET 6 у ряді категорій сильно випереджають веб-програми, побудовані за допомогою інших технологій [18].

Виходячи з того, що функціональною необхідністю сервера системи моніторингу СЕС буде запис та зчитування даних з БД, плюсом використання .NET є технологія Entity Framework Core. Вона ефективно вирішує завдання об'єктно-реляційного відображення та усунення розбіжності парадигм об'єктно-орієнтованих мов програмування і мов запитів баз даних [8]. Окрім цього, Entity Framework Core може використовуватись на різних платформах стеку .NET, а саме: Windows Forms, консольні програми, WPF, UWP та ASP.NET Core. Також слід зазначити, що EF Core — це кросплатформова технологія, що дозволяє задіяти її не тільки на Windows, але і на Linux і Mac OS X.

2.3 Дослідження систем управління базами даних

Однією з важливих частин загальної системи моніторингу СЕС є база даних. Без неї буде неможливим накопичування показників роботи системи, відображення аналітичної інформації, збереження журналу помилок, облік наявних інверторів та їхнього налаштування. Саме тому, потрібно обрати максимально зручну систему управління базами даних, яка буде підходити під загальні потреби застосунку. Для проведення аналізу було обрано три популярні СУБД, які можна використати для реалізації системи моніторингу:

1. Microsoft SQL Server — одна за найбільш поширених реляційних систем управління базами даних у світі, яка підходить для великих проектів та невеличких програмних застосунків.

До переваг Microsoft SQL Server відносяться:

- Високий рівень безпеки. Сервер MS SQL вважається одним із найбезпечніших серверів баз даних зі складними алгоритмами шифрування, що робить практично неможливим зламати рівні безпеки, встановлені користувачем [7].

- Висока продуктивність. Сервер MS SQL містить чудові можливості стиснення та шифрування, які покращують функції зберігання та пошуку даних.

- Підтримка інтеграції з більшістю ORM фреймворків (Entity Framework, Hibernate ORM, Sequelize ORM, Django та іншими) [3].

До недоліків Microsoft SQL Server відносяться:

- Залежність від апаратних можливостей. Нові версії СУБД вимагають більш просунутих системних можливостей для підтримки ресурсів, необхідних для роботи бази даних, що ускладнює або унеможливлює її встановлення на відносно застарілому програмному обладнанні.

- Необхідність обов’язкового встановлення фреймворку .NET для функціонування SQL серверу.

- Велика вартість корпоративної версії в порівнянні з аналогічними системами.

2. PostgreSQL — це потужна об’єктно-реляційна СУБД з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які дозволяють безпечно зберігати і масштабувати найскладніші структури даних [5].

До переваг PostgreSQL відносяться:

- Чудова масштабованість. Вертикальна масштабованість є відмінною рисою PostgreSQL. Враховуючи, що майже будь-яке програмне рішення має тенденцію до розвитку, що призводить до розширення бази даних, така властивість СУБД є безумовним плюсом для бізнес-рішень.

- Підтримка спеціальних типів даних. PostgreSQL за умовчанням підтримує велику кількість типів даних, таких як JSON, XML, H-Store та інші. Крім того, вона дозволяє користувачам визначати власні типи даних, що забезпечує покращену гнучкість таблиці.

- Просте обслуговування та адміністрування при різних сценаріях використання СУБД.

- Підтримка складних запитів. PostgreSQL ефективно працює зі складними, комплексними запитами. Система може обробляти трудомісткі операції, які можуть одночасно складатися з читання, запису і валідації даних [1].

До недоліків PostgreSQL відносяться:

- Низька продуктивність при великій кількості запитів. Для кожного нового підключення клієнта PostgreSQL створює новий процес з виділенням для нього близько 10 МБ пам'яті. Відповідно, при великій кількості запитів на читання даних, PostgreSQL зазвичай менш продуктивний, ніж інші РСУБД.

3. MySQL — це швидка, надійна та гнучка система управління реляційними базами даних, яка підходить для проектів різних розмірів, добре працює з веб-серверами Apache і численними серверними операційними системами [14].

До переваг MySQL відносяться:

- Сумісність з хмарними сервісами. MySQL першочергово розроблювалася як бізнес-орієнтоване веб-рішення. Вона найкраще підходить для хмарних і Big Data платформ, а також підтримується такими провідними службами, як Oracle Cloud, AWS та Azure.

- Реплікація. СУБД підтримує кілька різних типів реплікації – обміну інформацією між двома або більше хостами, для підвищення надійності, доступності і відмовостійкості системи. Це корисно для налаштування рішення з метою резервного копіювання бази даних або її горизонтального масштабування.

- Універсальність. MySQL працює на ОС сімейств Windows, Linux, Unix, Solaris та інших. Також СУБД має API для більшості популярних мов програмування: C та C++, PHP, Python, Ruby, Java та інших.

До недоліків MySQL відносяться:

- Обмежена відповідність стандартам SQL. Оскільки MySQL розроблено для швидкості та простоти використання, а не для повної сумісності з SQL, вона має певні функціональні обмеження. Наприклад, СУБД не підтримує запити FULL JOIN.

Виходячи з цього, PostgreSQL — це вдалий варіант для використання клієнт-серверним застосунком. Ця СУБД зможе забезпечувати швидке виконання запитів до БД, працювати з великими об’ємами даних, нівелювати більшість проблем, пов’язаних з можливим масштабуванням системи, та є абсолютно безкоштовним рішенням.

2.4 Аналіз можливості реалізації клієнтського застосунку для десктоп

Однією з можливих реалізацій клієнтської частини системи моніторингу СЕС є створення десктоп застосунку.

До переваг наведеного рішення можна віднести:

- Залежність роботи застосунку лише від локального пристрою на якому він встановлений. У разі збою в роботі приладу або програми, системою можна користуватися на іншому ПК.

До недоліків наведеного рішення можна віднести:

- Залежність від апаратного забезпечення комп’ютера.
- Необхідність у встановленні додатка на пристрій кожного з користувачів, які будуть мати доступ до системи. Для компанії, в якій багато робочих місць, це може зайняти чимало часу.
- Необхідність оновлення застосунку щоразу, як виходить нова версія. Незважаючи на те, що найчастіше цей процес автоматизований — він все одно займає час користувачів і ресурси пристроїв. Окрім цього, додатково доведеться відстежувати версії на кожному пристрої.

2.5 Аналіз можливості реалізації клієнтського веб-застосунку

Іншим варіантом взаємодії клієнта з програмною системою виступає створення веб-застосунку.

До переваг наведеного рішення можна віднести:

- Відсутність необхідності завантаження застосунку. Програма розгортається лише на локальному або хмарному сервері.
- Простий процес підтримки продукту. Оновлення застосунку відбувається лише на сервері, на відміну від десктопних застосунків.
- Незалежність від операційної системи або пристрою клієнта. Веб-застосунком можна легко користуватися за допомогою ПК, планшета або смартфону.
- Доступ до застосунку з будь-якого місця в будь-який час при наявності Інтернету.

До недоліків наведеного рішення можна віднести:

- Залежність від роботи сервера, на якому розгорнутий веб-застосунок. У випадку апаратного або програмного збою серверного пристрою клієнт не зможе взаємодіяти з системою.
- Пряма залежність роботи застосунку від якості мережевого з'єднання.

2.6 Фреймворк Blazor в якості основного інструменту для розробки клієнтського застосунку

Враховуючи проаналізовані переваги та недоліки клієнтського дескотопного та веб-застосунку, а також потребу у створенні зручного інтерфейсу та відображенні аналітичної інформації у вигляді графічних даних, для реалізації системи моніторингу СЕС доцільним є створення саме веб-програми. З цією метою можна використати один з популярних веб-фреймворків – Blazor.

Blazor — це багатофункціональний інструмент на мові C# з використанням платформи .NET з відкритим вихідним кодом для створення інтерактивних користувацьких інтерфейсів [2]. Даний фреймворк може використовуватися при створенні серверних застосунків (BlazorServer), клієнтських веб-застосунків за допомогою технології WebAssembly (Blazor WebAssembly) та мобільних і десктопних застосунків (Blazor Hybrid).

Для створення клієнтського веб-застосунку системи моніторингу СЕС вдалим рішенням буде використання Blazor WebAssembly. Це платформа для створення клієнтських інтерактивних односторінкових веб-застосунків. Такі застосунки завантажуються в браузер користувача та автономно працюють в ньому за допомогою технології WebAssembly [9]. Таким чином, клієнтський веб-застосунок не залежить від сервера та значною мірою знижує навантаження на нього.

Серед особливостей Blazor можна виділити:

- Використання мови C#. Це усуває необхідність у використанні JavaScript і спрощує розробку та взаємодію з серверною частиною системи на .NET.
- Компонентність. За допомогою Blazor можливо створювати власні компоненти, які можна неодноразово повторювати в HTML-документі. Це дозволяє покращити структуру коду та зменшити його повторюваність.
- Реактивність. Blazor — це реактивний MVC-фреймворк в якому представлення змінюється в залежності зміни моделі.
- Доступність. Фреймоврк розповсюджується за ліцензією Apache License 2.0. Його можна вільно використовувати як в комерційних, так і в особистих цілях.
- Використання попередньої компіляції (Ahead-of-Time) коду. Наведена технологія значним чином зменшує час запуску та значним чином покращує продуктивність веб-застосунку, в порівнянні з аналогічними фреймворками.
- Вбудовані компоненти. Інструментарій Blazor включає багато вбудованих компонентів, таких як таблиці та форми. Окрім цього, він підтримує використання NuGet пакетів з інтерактивними компонентами (Blazor.Bootstrap, ChartJs.Blazor, Radzen Components for Blazor), наприклад, графіками та діаграмами.

Виходячи з розглянутих особливостей та можливостей фреймворку, Blazor WebAssembly є вдалим вибором у якості інструмента для розробки клієнтського веб-застосунку для системи моніторингу СЕС.

2.7 Висновки з розділу 2

1. Для реалізації взаємодії системи з інверторами за протоколами Modbus TCP та Modbus RTU доцільним є використання бібліотеки NModbus. Вона містить весь необхідний функціонал для комунікації з цільовими пристроями та доступна для використання при розробці застосунків на мові C#, яка також використовується для реалізації сервера.
2. В якості платформи для розробки серверної частини системи моніторингу СЕС було обрано .NET.
3. У якості СУБД для створення та взаємодія з базою даних для системи було обрано PostgreSQL.
4. Варіант використання клієнтського веб-застосунку є найбільш оптимальним для реалізації системи моніторингу. Фреймворк Blazor WebAssembly містить усі необхідні можливості для створення зазначеного застосунку.

РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ СОНЯЧНИХ ЕЛЕКТРОСТАНЦІЙ

3.1 Вимоги до системи моніторингу

3.1.1 Функціональні вимоги

До розроблюваної системи моніторингу СЕС було висунуто наступні функціональні вимоги:

- Налаштування підключення. Система повинна надавати можливість налаштування параметрів підключення інверторів, таких як IP-адреси, порти, протоколи зв'язку тощо.
- Сумісність з різними типами інверторів. Система повинна коректно працювати з різними типами інверторів, забезпечуючи збір та обробку даних від них.
- Збереження даних. Система повинна зберігати та надавати доступ до архівної інформації про вироблення електроенергії за тривалий час, що дозволить аналізувати та порівнювати дані за різні періоди.
- Журнал помилок. Система повинна мати журнал помилок в роботі інверторів, що дозволить виявляти та вирішувати проблеми в роботі електростанцій.
- Статистичні дані. Система повинна відображати статистичні дані щодо загального вироблення електроенергії, такі як сумарне вироблення за день, місяць, рік, графіки потужності тощо.
- Інформація про окремий інвертор. Система повинна надавати детальну інформацію щодо роботи окремого інвертора, таку як потужність, стан, напруга тощо.
- Використання графіків. Система повинна використовувати графіки для відображення аналітичних даних показників роботи сонячних електростанцій.

3.1.2 Нефункціональні вимоги

До розроблюваної системи моніторингу СЕС було висунуто наступні нефункціональні вимоги:

- Ефективність. Система повинна працювати швидко та ефективно, забезпечуючи мінімальну затримку у відповідях та оптимальне використання ресурсів.
- Надійність. Система повинна бути надійною, забезпечуючи стабільну роботу без відмов та відновлення в разі виникнення помилок.
- Масштабованість. Система повинна бути масштабованою, здатною працювати зі збільшенням обсягу даних та кількості підключених інверторів.
- Користувацький інтерфейс. Система повинна мати інтуїтивно зрозумілий користувацький інтерфейс для зручної взаємодії з системою та отримання необхідної інформації.
- Локалізація. Система повинна підтримувати можливість локалізації, що дозволить представити інтерфейс та інформацію в різних мовах та регіональних налаштуваннях.
- Супровід та підтримка. Система повинна мати можливість для її супроводу, оновлення та підтримки, забезпечуючи виправлення помилок та додавання нового функціоналу.

3.2 Архітектура системи моніторингу

Система моніторингу сонячних електростанцій (див. Рис. 3.1) має багатошарову архітектуру та складається з наступних компонентів:

- Веб-застосунок, який взаємодіє з API для отримання і відображення даних інверторів користувачеві.
- API, що надає інтерфейс для взаємодії між клієнтським веб-додатком та вмістом БД.
- База даних, яка призначена для зберігання даних, зібраних сервісом опитування інверторів.

- Сервіс моніторингу інверторів. Він відповідає за збирання даних шляхом опитування інверторів сонячної електростанції та запис отриманих даних до бази даних.

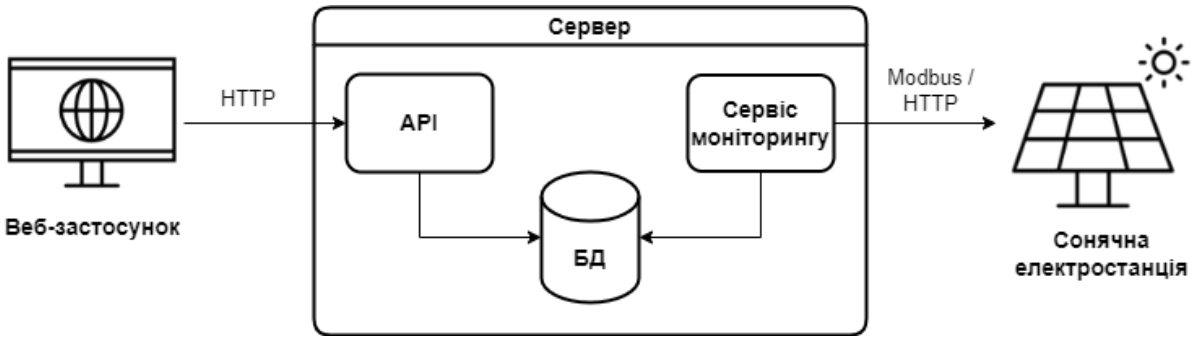

Рисунок 3.1 — Загальна архітектура системи моніторингу сонячних електростанцій

3.3 Проектування бази даних

Перед початком розробки застосунку було створено схему базу даних, яка відповідає вимогам системи моніторингу СЕС (Рис. 3.2).

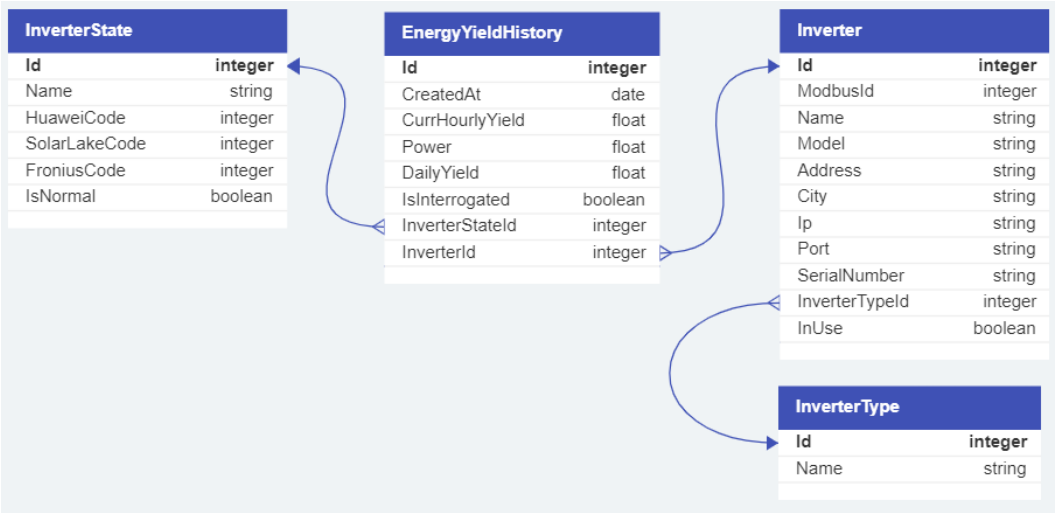

Рисунок 3.2 — Схема бази даних системи моніторингу сонячних електростанцій

Таблиця InverterType представляє собою тип інвертора та містить наступні поля:

- Id — ідентифікатор запису (первинний ключ).
- Name — назва типу інвертора.

Таблиця Inverter представляє собою інвертор та містить наступні поля:

- Id — ідентифікатор запису (первинний ключ).
- ModbusId — ідентифікатор пристрою для підключення за протоколом Modbus.
- Name — назва інвертора.
- Address — адреса розташування інвертора.
- City — місто в якому розташований інвертор.
- Ip — IP-адреса інвертора або контролера, до якого під’єднано інвертор.
- Port — порт інвертора або контролера, до якого під’єднано інвертор.
- SerialNumber — серійний номер інвертора.
- InverterTypeId — ідентифікатор типу інвертора (зовнішній ключ до таблиці InverterType).
- InUse — покажчик того чи використовується інвертор чи ні.

Таблиця InverterState представляє собою статус інвертора та містить наступні поля:

- Id — ідентифікатор запису (первинний ключ).
- Name — назва статусу інвертора.
- HuaweiCode — код інвертора від виробника Huawei.
- SolarLakeCode — код інвертора від виробника SolarLake.
- FroniusCode — код інвертора від виробника Fronius.
- IsNormal — прапорець, що визначає чи є статус нормальним або ідентифікує помлику.

Таблиця EnergyYieldHistory представляє собою статус інвертора та містить наступні поля:

- Id — ідентифікатор запису (первинний ключ).
- CreatedAt — дата та час створення запису.

- CurrHourlyYield — вироблення електроенергії за останню годину.
- Power — поточна потужність пристрою.
- DailyYield — кількість виробленої електроенергії за поточний день.
- IsInterrogated — покажчик вдалої спроби опитування інвертора.
- InverterStateId — ідентифікатор статусу інвертора (зовнішній ключ до таблиці InverterState).
- InverterId — ідентифікатор інвертора (зовнішній ключ до таблиці Inverter).

Для проектування БД було обрано Code First підхід через можливість швидкої зміни моделей даних, простоту синхронізації з БД, зручність при роботі з об'єктно-орієнтованим кодом та можливість використання ORM-фреймворків.

Виходячи з цього, було створено класи сутностей за вищенаведеною схемою (Рис. 3.2). Також було додано клас контексту SharedDBContext для опису вмісту БД, клас Configuration з налаштуваннями для міграції та клас InitialCreate з описом базової міграції зі створення бази даних. Після цього, за допомогою команди Update-Database було проведене оновлення БД зі створенням усіх необхідних таблиць.

Завдяки виконаним налаштуванням серверний застосунок може швидко наповнити базу даних необхідними таблицями при його першому запуску на цільовому пристрої.

3.4 Налаштування середовища для розробки системи

Для реалізації системи було використано інтегроване середовище розробки Visual Studio 2022, яке окрім стандартних функцій минулих версій містить низку нових можливостей, таких як автодоповнення коду, поліпшення інструментів розробки та підтримка останніх версій фреймворку .NET.

Фреймворком для розробки системи було обрано .NET 6.0, який в повній мірі забезпечує реалізацію необхідного функціоналу системи та його ефективну роботу завдяки своїй високій продуктивності і широким можливостям для роботи з мережевими протоколами та базами даних.

Окрім цього, за допомогою менеджеру пакетів NuGet для розробки системи також були завантажені наступні інструменти:

1. Entity Framework Core — для зберігання, оновлення та отримання даних з БД, розробки та керування моделями даних, а також використання LINQ-запитів.
2. NModbus — для забезпечення можливості взаємодії з різними типами протоколу Modbus, що використовується для зчитування даних з інверторів.
3. log4net — для записування лог-повідомлень та відслідковування подій у системі, що дозволяє відстежувати помилки та аналізувати роботу застосунку.

Також, було встановлено та розгорнуто об’єктно-реляційну систему управління базами даних PostgreSQL для створення БД з метою накопичення та зберігання інформації про роботу сонячних електростанцій.

3.5 Розробка та функціонал серверної частини комп’ютерної системи для моніторингу сонячних електростанцій

3.5.1 Проектування сервісу моніторингу інверторів та API

Перед початком розробки сервісу моніторингу інверторів було розроблено діаграму класів для сервісу моніторингу (див. Рис. 3.3) задля кращого розуміння організації класів та їхніх властивостей.

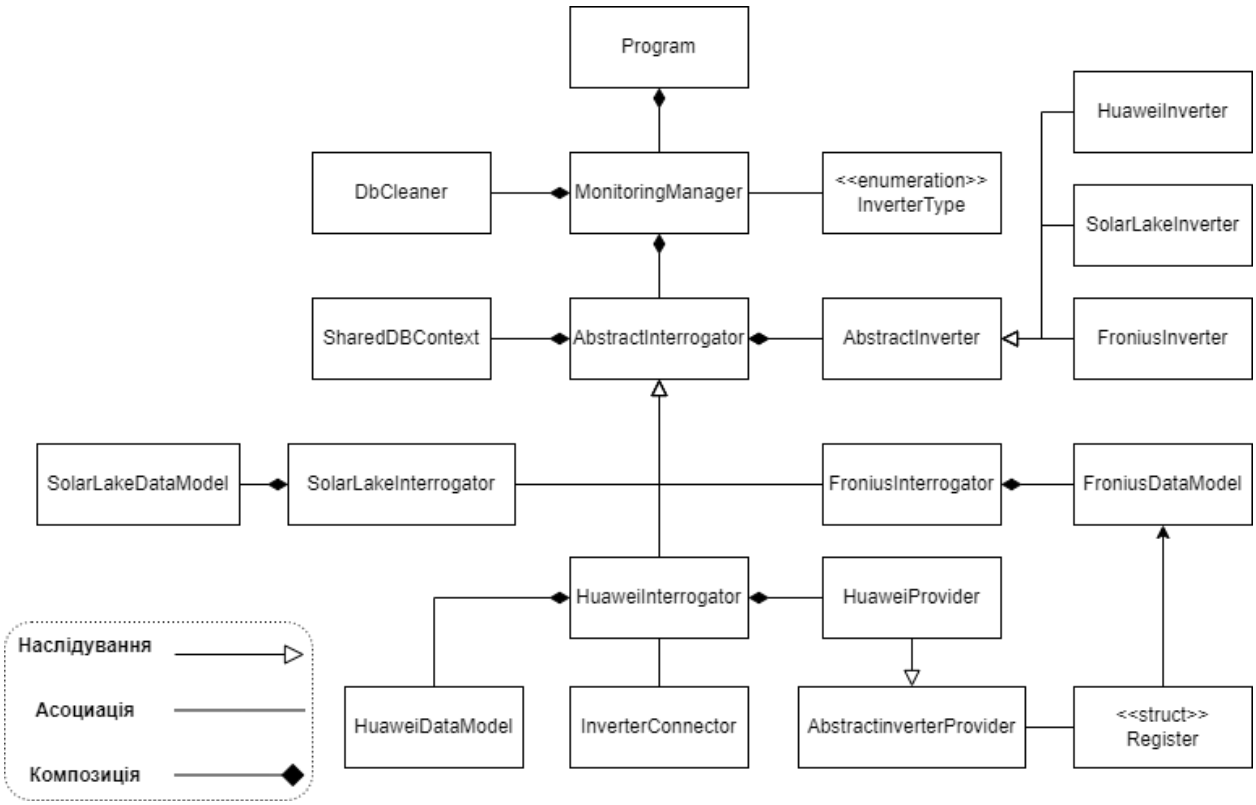

Рисунок 3.3 — Діаграма класів сервісу моніторингу інверторів

Окрім цього, було створено діаграму класів API (див. Рис. 3.4) системи моніторингу СЕС задля також візуалізації структури застосунку, його компонентів та взаємозв'язків між ними.

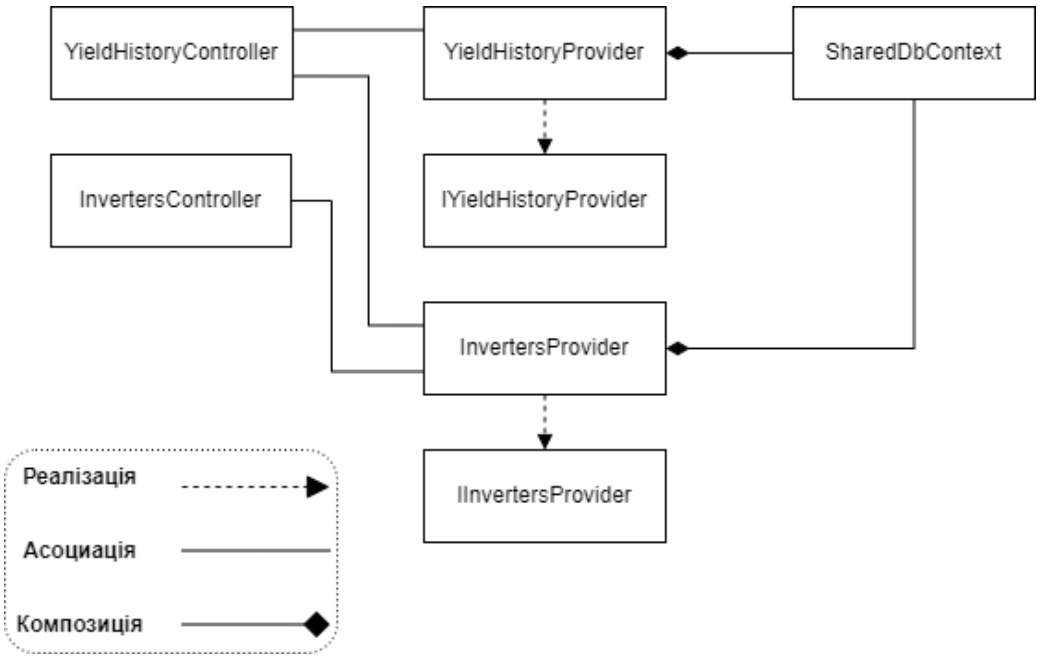

Рисунок 3.4 — Діаграма класів API (без урахування класів моделей)

Детальний огляд основних класів та опис реалізації головного функціоналу розглянуто в пунктах 3.5.2 та 3.5.3.

3.5.2 Реалізація основного функціоналу сервісу моніторингу інверторів

Сервіс моніторингу представляє собою застосунок для отримання інформації з інверторів за допомогою використання різновидів протоколу Modbus та HTTP. Він складається з наступних основних класів, що забезпечують його функціональність:

1. Клас `Program` — головний клас програми, в якому відбувається ініціалізація бази даних та налаштувань системи, а також запускається процес моніторингу інверторів.

2. Клас `MonitoringManager` необхідний для формування загального списку об’єктів `AbstractInterrogator` та запуску процесу опитування кожного з доступних інверторів.

3. Клас `DbCleaner` функціонує в якості очищувача бази даних від старих записів. Період, за який відбувається видалення записів з БД, вказується у файлі конфігурації системи.

4. Клас `DbSharedContext` представляє собою проміжний шар між застосунком та БД, що виконує запити та надає доступ до даних, що зберігаються в ній.

5. Абстрактний клас `AbstractInterrogator` містить в собі список об'єктів класу `AbstractInverter`, тобто моделей інверторів, які необхідно опитувати. Також він містить методи з додавання і видалення інверторів до внутрішнього списку та методи початку і завершення процесу опитування інверторів з нього (див. Лістинг 1).

Лістинг 1 — Вміст класу `AbstractInterrogator`

```
public abstract class AbstractInterrogator
```

```
{
    protected List<AbstractInverter> invertersList;
    protected SharedDBContext sharedDBContext;

    public void AddInverter(AbstractInverter inverter)
    {
        invertersList.Add(inverter);
    }

    public bool RemoveInverter(AbstractInverter inverter)
    {
        return  invertersList.Remove(invertersList.Find(inv
=>   (inv.Ip.Equals(inverter.Ip)   &&   inv.Port.Equals(in-
verter.Port))));
       }

    public abstract void StartInterrogation();
    public abstract void StopInterrogation();
}
```

6. Абстрактний клас `AbstractInverter` містить поля налаштувань інверторів, які характерні для усіх пристроїв та необхідні для функціонування системи (див. Лістинг 2). Даний клас використовується у процесі опитування у якості контейнера таких даних:

- IP-адреса інвертора.
- Порт, виділений для комунікації з інвертором.
- Кількість спроб підключень.
- Статус підключення до серверу.

Лістинг 2 — Вміст класу `AbstractInverter`

```
public abstract class AbstractInverter
{
      protected string ip;
      protected int port;
      protected int connectionTries;
      protected bool connectedToServer;
      public static int connectionTriesLimit = 3;

      public string Ip { get { return ip; } }
      public int Port { get { return port; } }
      public int ConnectionTries { get { return connec-
tionTries; } set { connectionTries = value; } }
```

```
        public bool ConnectedToServer { get { return connect-
edToServer; } set { connectedToServer = value; }
}
```

7. Структура Register представляє модель, що описує характеристики регістру з за протоколом Modbus (див. Лістинг 3). Вона містить наступні поля:

- Назва параметру, що зберігається за адресою регістра.
- Адреса регістру.
- Кількість бітів для читання, починаючи з адреси.

Лістинг 3 — Вміст структури Register

```
public struct Register
{
     public string SignalName { get; set; }
     public ushort Address { get; set; }
     public ushort Quantity { get; set; }

     public Register(string name, ushort address, ushort
quantity)
     {
          SignalName = name;
           Address = address;
           Quantity = quantity;
     }
}
```

8. Класи `FroniusInverter`, `HuaweiInverter` та `SolarLakeInverter` є нащадками класу `AbstractInverter` та містять додаткові поля налаштувань, притаманні інверторам від компаній Fronius, Huawei та SamilPower відповідно.

9. Класи `FroniusDataModel`, `HuaweiDataModel` та `SolarLakeDataModel` – моделі даних, що містять поля з інформацією, необхідною для підключення до відповідного інвертора.

10. Клас `FroniusInterrogator` є нащадком класу `AbstractInterrogator` та реалізує отримання даних з інвертора за допомогою API запитів.

Такий спосіб було обрано через специфіку опитуваної моделі інвертора, яка підтримує використання технології Fronuis Solar API. За цим методом сервер формує GET запит до необхідного пристрою за посиланням «`/solar_api/v1/GetInverterRealtimeData.cgi`». Доступні параметри для наведеного запиту зазначені в таблиці 1.

Таблиця 1 — Параметри GET запиту GetInverterRealtimeData.cgi для Fronuis Solar API

Параметр	**Тип**	**Можливі значення**	**Опис**
`scope`	String	«Device», «System»	Тип запиту (для певного пристрою або всієї системи)
`deviceid`	String	від 0 до 99	Ідентифікатор інвертору (доступний лише при типі запиту «Device»)
`datacollection`	String	«CumulationInverterData» «CommonInverterData» «3PInverterData» «MinMaxInverterData»	Визначає колекцію даних, яка повинна бути отримана з пристрою (доступний лише при типі запиту «Device»)

Таким чином, можна утворити необхідний запит для отримання повідомлення з необхідними даними конкретного інвертора. Наприклад, GET повідомлення на отримання загальних даних щодо роботи інвертора з ідентифікатором «1» та IP-адресою «192.168.57.11» буде виглядати наступним чином: «`http://192.168.57.11/solar_api/v1/GetInverterRealtimeData.cgi?scope=Device&datacollection=CommonInverterData&deviceid=1`».

В поточній реалізації системи GET запит надсилається на інвертор за допомогою класу `HttpWebRequest`, а отримана відповідь зберігається в об’єкті `HttpWebResponse`. Після отримання повідомлення у форматі JSON, його необхідно десеріалізувати, використовуючи метод `DeserializeObject` класу `JsonConvert` та отримати необхідні поля, в яких зберігаються параметри щодо

роботи певної сонячної електростанції. Після цього формується модель з отриманих даних для подальшого збереження в БД. У випадку помилки при надсиланні повідомлення або десеріалізації формується модель даних за замовчуванням, що ідентифікує про невдалу спробу виконання запиту.

Реалізація наведеного алгоритму представлена на лістингу 4.

Лістинг 4 — Функціонал отримання даних роботи інвертора Fronius

```
private void GetDataFromInverter(FroniusInverter inverter,
FroniusDataModel receivedModel)
{
     FroniusDataModel froniusDataModel = new Froni-
usDataModel();
     string connection = $"http://{inverter.Ip}";
     string apiRequest =    $"/solar_api/v1/Get-
InverterRealtimeData.cgi?scope=Device&datacollection=Com-
monInverterData&deviceid={inverter.ModbusId}";

     try {
          HttpWebRequest httpWebRequest =
               (HttpWebRequest)WebRequest.Create (connection
+ apiRequest);
          using (HttpWebResponse response = (HttpWebRe-
sponse)       httpWebRequest.GetResponse())
          using (Stream stream = re-
sponse.GetResponseStream())
          using (StreamReader reader = new
StreamReader(stream))       {
               string jsonString = reader.ReadToEnd();
               var jsonObject = JObject.Parse(jsonString);
               var data = jsonOb-
     ject["Body"]["Data"].ToString();
               var model = JsonConvert.DeserializeObject
          <CommonInverterDataModel>(data);
               if (model != null && model.DeviceStatus !=
     null) {
                    froniusDataModel.ActivePower = (model.PAC
          ==        null)    ? 0 : ((double)model.PAC.Value *
                         0.001);
                    froniusDataModel.DeviceStatus =
               model.DeviceStatus.StatusCode;
                    froniusDataModel.EnergyProducedToday =
                    model.DAY_ENERGY.Value * 0.001; //convert
          from Wh  to kWh
               }
```

```
            }
            receivedModel.Copy(froniusDataModel);
        }
        catch(Exception e) {
            Logger.WriteErrorLog("FroniusInterrogator : " +
                $"GetDataFromInverter({inverter.Ip}): " +
e.Message);
            receivedModel.Copy(froniusDataModel);
        }
}
```

11. Класи `HuaweiInterrogator` та `SolarLakeInterrogator` є нащадками класу `AbstractInterrogator` та реалізують отримання даних з інверторів за допомогою протоколів Modbus RTU over TCP та Modbus TCP.

Через використання певними інверторами (SUN2000-33KTL-A та SolarLake-10000TL) фізичного інтерфейсу RS-485 до них було під'єднано перетворювачі інтерфейсів типу Ethernet RS-485. Це необхідно для утворення загальної мережі з цими пристроями з метою опитування та моніторингу. В результаті наведених дій пряма комунікація з інверторами за допомогою Modbus RTU є неможливою. Саме для цього було вирішено використовувати модифікацію протоколу Modbus RTU over TCP, що дозволяє передавати дані мережею Ethernet.

Реалізація комунікації за наведеним протоколом відбувається за допомогою класу `ModbusFactory` бібліотеки NModbus. Для цього створюється TCP клієнт та передається в конструктор наведеного класу, після чого утворюється об'єкт `IModbusSerialMaster`, що підтримує надсилання запитів до підлеглого пристрою (див. Лістинг 5).

Лістинг 5 — Функціонал створення об'єкту `IModbusSerialMaster`

```
public static IModbusSerialMaster CreateRtuOverTcpMas-
ter(byte[] ip, int port, TcpClient client)
{
    IPEndPoint endPoint = new IPEndPoint(new IPAddress(ip),
port);
    client.Connect(endPoint);

    var factory = new ModbusFactory();
```

```
    return factory.CreateRtuMaster(new TcpClientAdapter(cli-
ent));
}
```

В поточній реалізації для надсилання запитів використовується метод `ReadInputsAsync`, який відповідає команді з кодом «04» (читання групи регістрів вводу). В нього передаються ідентифікатор пристрою, адреса комірки пам'яті та кількість бітів, що містять необхідне значення. Метод для надсилання команд відображено на лістингу 6.

Лістинг 6 — Метод для надсилання запиту на підлеглий пристрій

```
public async Task<bool[]> ReadInputsRegisterAsync(Register
register)
{
    return await SerialMaster.ReadInputsAsync(DeviceId, reg-
ister.Address, register.Quantity);
}
```

В результаті надсилання команди система отримує масив бітів, що містять необхідне значення поля. Після цього, наведений масив конвертується в 32-бітне значення, яке потім записується в загальну модель опитування інвертора для подальшого збереження в БД. Приклад реалізації отримання значення параметру добового вироблення енергії зазначено на лістингу 7.

Лістинг 7 — Отримання значення параметру добового вироблення енергії з інвертора за протоколом Modbus RTU over TCP

```
TcpClient client = new TcpClient();
HuaweiProvider provider = new HuaweiProvider(Convert.To-
Byte(inverter.ModbusId),
    InverterConnector.CreateRtuOverTcpMaster(IPAd-
dress.Parse(inverter.Ip).GetAddressBytes(),  inverter.Port,
client));

tokenSource = new CancellationTokenSource();
ushort[]? dailyEnergyYield = null;
Task dailyEnergyYieldTask = Task.Run(() => {
    dailyEnergyYield    =    provider.ReadHoldingRegisterA-
sync(HuaweiRegisterV2.DailyEnergyYield).Result;
```

```
}, tokenSource.Token);

if (dailyEnergyYield == null)
{
    client.GetStream().Close();
    client.Close();
    tokenSource.Cancel();
    return new HuaweiDataModel();
}

if (BitConverter.IsLittleEndian)
    Array.Reverse(dailyEnergyYield);

var dailyEnergyYieldConverted = dailyEnergyYield[0] | (dai-
lyEnergyYield[1] << 16);
```

Реалізація комунікації за протоколом Modbus TCP відбувається за допомогою класу `ModbusTCPMaster` бібліотеки NModbus. Для цього в конструктор методу передаються параметри підключення до інвертора, такі як IP адреса, номер порту та прапор типу зв'язку (асинхронний або синхронний) для визначення необхідності в очікуванні відповіді від клієнта під час надсилання команд на запис даних. Після цього, необхідно підписатися на подію `OnResponseData`, що повертає значення параметру, отримане з підлеглого пристрою. Реалізація наведених дій відображена на лістингу 8.

Лістинг 8 — Налаштування об'єкту класу `ModbusTCPMaster`

```
byte[] responseData;

ModbusTCPMaster mbMaster = new ModbusTCPMaster(inverter.Ip,
Convert.ToUInt16(inverter.Port), true);

mbMaster.OnResponseData += new ModbusTCPMaster.ModbusTCP-
Master.ResponseData((ushort id, byte unit, byte function,
byte[] values) => { responseData = values;});
```

Надсилання запитів до підлеглих пристроїв відбувається за допомогою методу `ReadHoldingRegister`, який відповідає команді читання групи регістрів вводу. В нього передаються унікальний ідентифікатор транзакції, необхід-

ний для її верифікації в асинхронному режимі відправки повідомлень, ідентифікатор підлеглого пристрою, адреса комірки пам’яті, з якої потрібно почати читання даних, та довжину цих даних в бітах. В якості відповіді системі надходить масив байтів, що представляє необхідне значення регістру. Для конвертації такого значення було використано вбудований метод `ToInt32` стандартного класу `BitConverter`. Після конвертації значення воно записується до спеціальної моделі даних для подальшого збереження в БД. Приклад отримання значення поточної потужності інвертора відображено на лістингу 9.

Лістинг 9 — Отримання значення параметру поточної потужності електростанції з інвертора за протоколом Modbus TCP

```
byte[] responseData;

ModbusTCPMaster mbMaster = new ModbusTCPMaster(inverter.Ip,
Convert.ToUInt16(inverter.Port), true);

mbMaster.OnResponseData += new ModbusTCPMaster.ModbusTCP-
Master.ResponseData((ushort id, byte unit, byte function,
byte[] values) => {
                responseData = values;
            });

mbMaster.ReadHoldingRegister(3, Convert.ToByte(in-
verter.ModbusId), HuaweiRegisterV3.ActivePower.Address,
                            HuaweiRegisterV3.Active-
Power.Quantity);

if (BitConverter.IsLittleEndian)
    Array.Reverse(responseData);

var activePower = BitConverter.ToInt32(responseData, 0);

huaweiDataModel.ActivePower = Convert.ToDouble(activePower)
* 0.001;
```

12. Клас MonitoringManager виступає головним класом, що регулює процес опитування інверторів за заданим інтервалом. Він створює об’єкти класів `HuaweiInterrogator`, `SolarLakeInterrogator` та . Після цього, він отримує

сутності існуючих інверторів з бази даних та заповнює ними списки цих класів. Після цього виконується опитування кожного з необхідних пристроїв. Наведений процес відображено на лістингу 10.

Лістинг 10 — Реалізація алгоритму запуску процесу опитування інверторів

```
private void InterrogateInverters()
{
    try
    {
        if (NetworkInterface.GetIsNetworkAvailable())
        {
            interrogationManager.AddInterrogator(new Solar-
LakeInterrogator(dbContextFactory, _options));
            interrogationManager.AddInterrogator(new Froni-
usInterrogator(dbContextFactory));
            interrogationManager.AddInterrogator(new
HuaweiInterrogator(dbContextFactory));
            using (SharedDBContext db = dbContextFac-
tory.CreateDbContext())
            {
                foreach (var inverter in db.Inverter.In-
clude(x => x.InverterType).Where(inverter => in-
verter.InUse).ToArray())
                {
                    switch (inverter.InverterType.Name)
                    {
                        case InverterType.HuaweiV2:
interrogationManager.InterrogatorsList[2].AddInverter(new
HuaweiInverter(inverter.Ip, Convert.ToInt32(inverter.Port),
inverter.ModbusId, 2, inverter.SerialNumber));
                            break;

                        case InverterType.HuaweiV3:
interrogationManager.InterrogatorsList[2].AddInverter(new
HuaweiInverter(inverter.Ip, Convert.ToInt32(inverter.Port),
inverter.ModbusId, 3, inverter.SerialNumber));
                            break;

                        case InverterType.SolarLake:
   interrogationManager.Interroga-
torsList[0].AddInverter(new SolarLakeInverter(inverter.Ip,
Convert.ToInt32(inverter.Port), inverter.SerialNumber));
                            break;
```

```
                        case InverterType.Fronius:
     interrogationManager.Interroga-
torsList[1].AddInverter(new FroniusInverter(inverter.Ip,
Convert.ToInt32(inverter.Port), inverter.ModbusId, in-
verter.SerialNumber));
                            break;
                    }
                }
            }

            interrogationManager.StartInterrogators();
            interrogationManager.InterrogatorsList.Clear();
        }
        else
        {
            Logger.WriteInfoLog("MonitoringManager : " +
"Network is not available. Unable to interrogate invert-
ers");
        }
    }
    catch (Exception exp)
    {
        Logger.WriteErrorLog(exp.Message);
    };
}
```

3.5.3 Розробка та функціонал API комп'ютерної системи моніторингу сонячних електростанцій

Для взаємодії між клієнтським веб-застосунком та базою даних системи моніторингу було реалізовано API за архітектурним стилем REST.

Застосунок складається з наступних класів, які містять його основний функціонал:

1. Класи `InvertersController` та `YieldHistoryController` є контролерами, тобто використовуються для обробки вхідних HTTP-запитів щодо інверторів та історії вироблення електроенергії відповідно. Вони наслідують від базового класу `ControllerBase` та містять методи, що відповідають конкретним запитам до API на отримання, додавання, редагування або видалення даних (CRUD-запитів). При формуванні відповіді вони використовують розроблені класи `InvertersProvider` та `YieldHistoryProvider`.

Клас `InvertersController` реалізує запити на отримання інвертору за вказаним ідентифікатором (див. Лістинг 11), списку інверторів за вказаними параметрами фільтрації та пагінації, а також редагування, додавання та видалення окремих інверторів.

Лістинг 11 — Реалізація алгоритму обробки запиту на отримання даних інвертору за вказаним ідентифікатором.

```
[HttpGet("{id}")]
public async Task<ActionResult<Inverter>> GetInverter(int
id)
{
    try
    {
        var inverterReply = await _invertersProvider.Get-
Inverter(id);

        if (!inverterReply.IsSucces)
        {
            return StatusCode(StatusCodes.Status500Inter-
nalServerError);
        }

        if(inverterReply.Value == null)
            return StatusCode(StatusCodes.Status404Not-
Found);

        return Ok(inverterReply.Value);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);

        return StatusCode(StatusCodes.Status500Internal-
ServerError, ex.Message);
    }
}
```

Клас `YieldHistoryController` обробляє запити на отримання журналів помилок та опитування інверторів, генерації статистичної інформації для подальшого відображення на графіках та формування загальних даних щодо роботи системи в поточний момент часу (див. Лістинг 12).

Лістинг 12 — Реалізація алгоритму обробки запиту на отримання статистичної інформації щодо поточної роботи системи.

```
[HttpGet("statistics")]
public async Task<ActionResult<YieldStatistics>> GetYield-
Statistics()
{
    try
    {
        var currentPowerReply = await _yieldHistoryPro-
vider.GetCurrentPower();
        var dailyEnergyProductionReply = await
_yieldHistoryProvider.GetDailyEnergyProduction();
        var invertersOnlineReply = await _yieldHistoryPro-
vider.GetInvertersOnline();
        var totalInvertersReply = await _invertersPro-
vider.GetTotalInvertersCount();

        if (!currentPowerReply.IsSucces || !dailyEnergyPro-
ductionReply.IsSucces || !invertersOnlineReply.IsSucces ||
!totalInvertersReply.IsSucces)
        {
            return StatusCode(StatusCodes.Status500Inter-
nalServerError);
        }

        double power = (double)currentPowerReply.Value;
        double dailyEnergyProduction = (double)dailyEner-
gyProductionReply.Value;
        int invertersOnline = (int)invertersOnlineRe-
ply.Value;
        int totalInverters = (int)totalInvertersRe-
ply.Value;

        YieldStatistics yieldStatistics = new YieldStatis-
tics()
        {
            Power = Math.Round(power, 2).ToString("F2",
CultureInfo.InvariantCulture),
            TotalProduction = Math.Round(dailyEnergyProduc-
tion, 2).ToString("F2", CultureInfo.InvariantCulture),
            PvSystems = $"{invertersOnline}/{totalInvert-
ers}"
        };

        return Ok(yieldStatistics);
    }
    catch (Exception ex)
    {
```

```
        Console.WriteLine(ex.Message);

        return StatusCode(StatusCodes.Status500Internal-
ServerError, ex.Message);
    }
}
```

2. Класи `InvertersProvider` та `YieldHistoryProvider` відповідають за зв'язок з БД за допомогою LINQ-запитів та формування необхідних моделей або наборів даних. Методи цих класів повертають значення у вигляді іменованих кортежів з трьох елементів: булевого прапору коректності виконання операції, даних, що були отримані при виконанні запиту, та тексту помилки. Такий підхід представляє уніфіковану модель передачі даних, що значно спрощує обробку помилок та створення відповідей на HTTP-запити контролерів.

Клас `InvertersProvider` містить методи, що отримують дані з БД, які стосуються інверторів, а саме:

- Обрахування загальної кількості пристроїв (див. Лістинг 13).
- Отримання списку типів інверторів.
- Отримання списку інверторів за вказаними параметрами фільтрації.
- Виконання CRUD-операцій над окремими інверторами.

Лістинг 13 — Реалізація алгоритму обрахування загальної кількості інверторів.

```
public async Task<(bool IsSucces, int? Value, string? Er-
rorMessage)> GetTotalInvertersCount()
{
    try
    {
        int totalinvertersCount;

        using (SharedDBContext dbContext = await _dbCon-
textFactory.CreateDbContextAsync())
        {
            totalinvertersCount = dbContext.In-
verter.Count(x => x.InUse);
        }
        return (true, totalinvertersCount, null);
    }
```

```
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);

        return (false, null, ex.Message);
    }
}
```

Клас `YieldHistoryProvider` відповідає за отримання інформації, пов'язаної з виробленням електроенергії та історією опитування інверторів, та реалізує наступні операції:

- Визначення поточної потужності загальної системи СЕС.
- Визначення кількості активних інверторів.
- Обрахування кількості виробленої електроенергії за поточну добу (див. Лістинг 14).
- Отримання статистичних даних щодо вироблення електроенергії за добу.
- Отримання статистичних даних щодо вироблення електроенергії за вказаний проміжок часу.
- Отримання журналу опитування інверторів за заданими параметрами.
- Отримання журналу помилок при опитуванні інверторів за заданими параметрами.

Лістинг 14 — Реалізація алгоритму обрахування загальної кількості виробленої електроенергії за поточну добу.

```
public async Task<(bool IsSucces, double? Value, string?
ErrorMessage)> GetDailyEnergyProduction()
{
    try
    {
        DateTime currentDateTime = DateTime.UtcNow;
        double dailyProducedEnergy = 0;

        using (SharedDBContext dbContext = await _dbCon-
textFactory.CreateDbContextAsync())
        {
```

```
            var inverters = dbContext.Inverter.ToList();

            foreach (var inverter in inverters)
            {
                var historyRecord = dbContext.Ener-
gyYieldHistory.OrderByDescending(x => x.Cre-
atedAt).FirstOrDefault(x => x.InverterId == inverter.Id);

                if (historyRecord != null && histo-
ryRecord.CreatedAt.Date == currentDateTime.Date && histo-
ryRecord.IsInterrogated)
                {
                    dailyProducedEnergy += histo-
ryRecord.DailyYield;
                }
            }
        }

        return (true, dailyProducedEnergy, null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);

        return (false, null, ex.Message);
    }
}
```

3.6 Розробка та функціонал клієнтської частини комп'ютерної системи для моніторингу сонячних електростанцій

3.6.1 Проектування клієнтського веб-застосунку

В якості проектування перед реалізацією веб-застосунку було створено діаграму варіантів використання (англ. Use case diagram). Це діаграма, яка використовується з метою моделювання функціоналу системи чи застосунку з перспективи користувачів та допомагає у визначенні та моделюванні функціональних вимог.

Створена згідно до функціональних вимог, зазначених в пункті 3.1.1, діаграма варіантів використання відображена на рисунку 3.5. Її проектування значною мірою спростило відображення та виділення окремих потреб користувача, а також визначення сценаріїв взаємодії та можливостей системи.

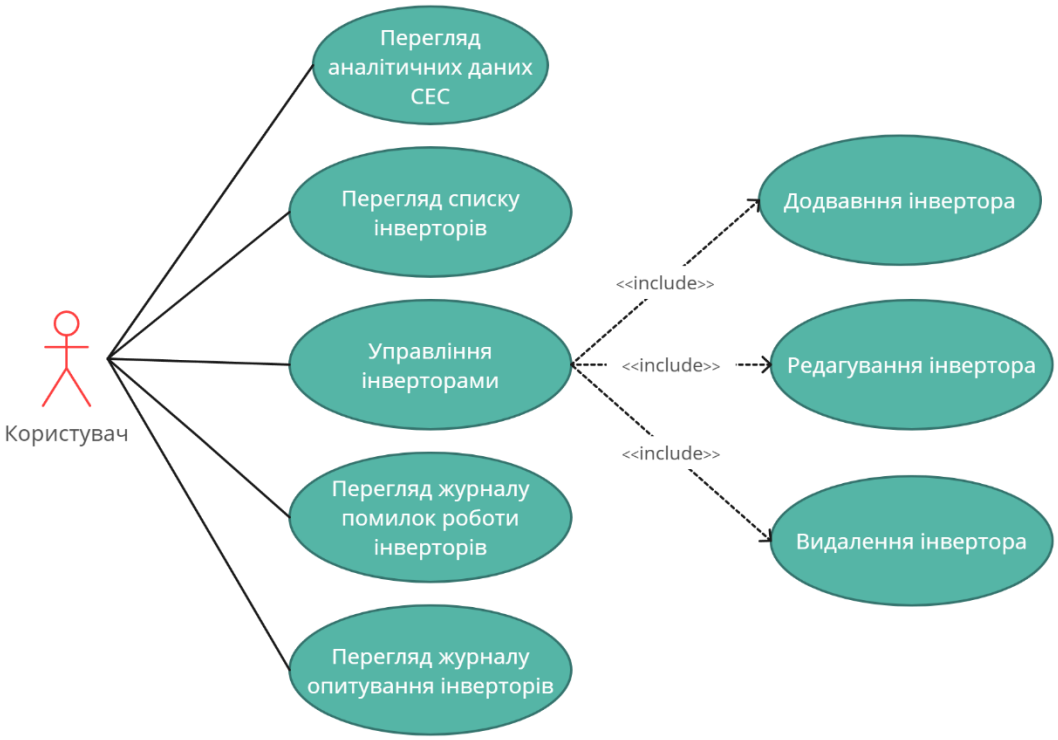

Рисунок 3.5 — Діаграма варіантів використання клієнтського застосунку

3.6.2 Функціонал клієнтського застосунку

Клієнтський застосунок виступає в якості шару взаємодії користувача з загальною системою моніторингу СЕС та надає йому усі необхідні функції для роботи з нею. Комунікація веб-застосунку з системою, зокрема з базою даних, відбувається за допомогою надсилання запитів до RESTful API, описаного в пункті 3.5.3.

Інтерфейс застосунку складається з верхньої панелі навігаційного меню, що включає в себе назву системи, список посилань на відповідні сторінки і селектор мови локалізації, та робочої області, яка містить певне наповнення в залежності від обраної сторінки. Сам застосунок містить чотири сторінки, що відповідають окремим тематичним розділам.

1. Сторінка огляду системи. Вона надає користувачеві загальну інформацію про стан усіх систем моніторингу СЕС. Верхня частина сторінки складається з трьох блоків даних, що містять статистичну інформацію про стан загальної системи електростанцій на момент проведення останнього опитування. Вони представляють собою наступні дані:

- Загальна потужність усіх наявних систем.
- Кількість активних та неактивних інверторів.
- Загальне добове вироблення електроенергії.

В центральній частині знаходиться графік, що відображає статистичну інформацію загального вироблення електроенергії усіма системами СЕС в певний момент часу (годину або добу). Окрім цього, користувач може обирати день або інтервал днів за який необхідно отримати наведені дані.

Реалізований інтерфейс сторінки наведено на рисунку 3.6.

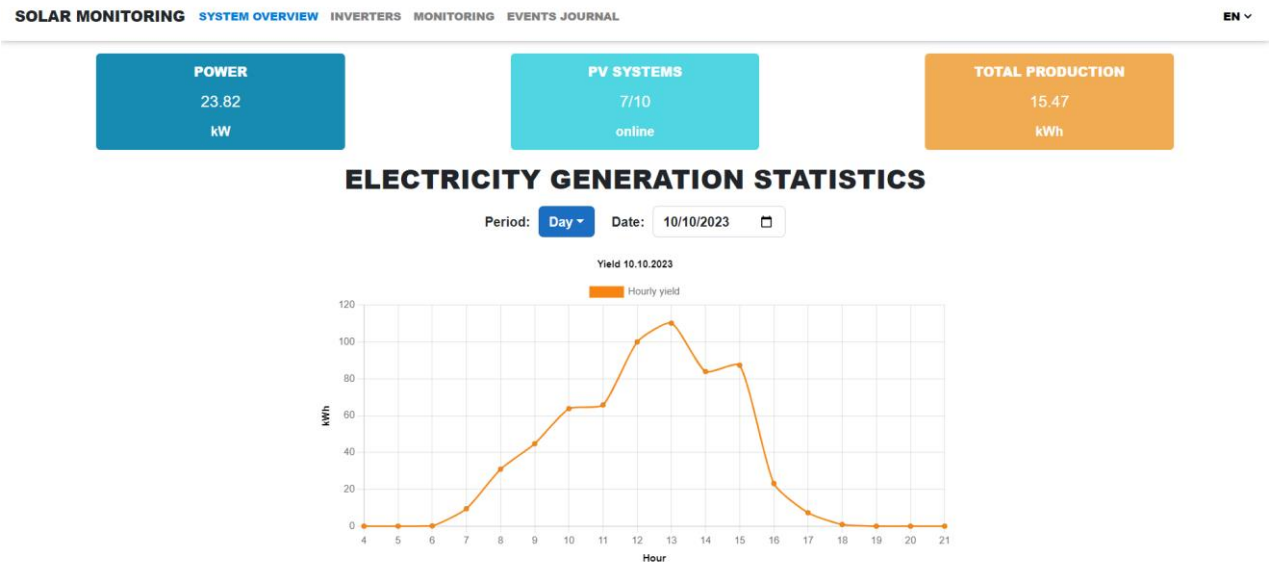

Рисунок 3.6 — Сторінка огляду системи

2. Сторінка списку інверторів (див. Рис. 3.7). Вона містить загальний список усіх підключених до системи сонячних електростанцій. Дані на сторінці представлені у вигляді таблиці, що містить наступні стовпці даних:

- Статус інвертора. Він представлений у вигляді іконок, що відображають стан приладу у спрощеному вигляді.
- Назва сонячної електростанції.
- Потужність системи під час останнього опитування.
- Кількість виробленої електроенергії системою за поточну добу під час останнього опитування.

- Дата та час останнього опитування.
- Загальна кількість помилок інвертора за поточну добу.

Такий формат полегшує виявлення несправностей в системі та надає коротку актуальну інформацію щодо роботи кожної з СЕС.

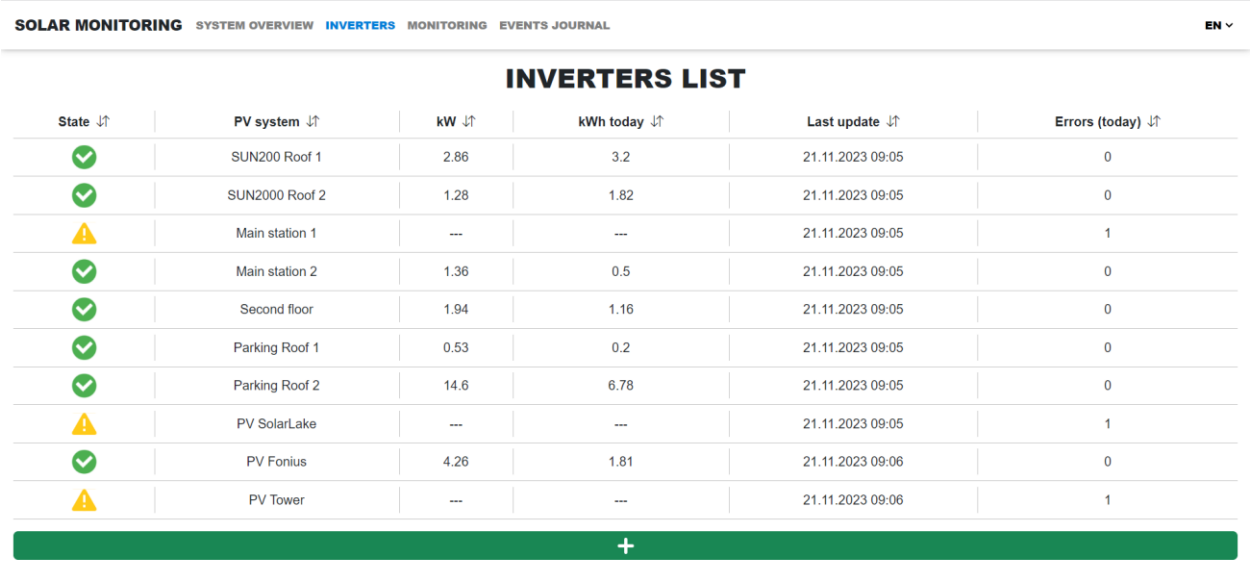

Рисунок 3.7 — Сторінка списку інверторів

Також, однією з головних можливостей цієї сторінки є перегляд та редагування інформації окремих інверторів, а також їх видалення. Для цього необхідно натиснути на необхідний запис в таблиці, після чого відкриється модальне вікно інвертора, що надає користувачеві представлені можливості (див. Рис. 3.7). Окрім цього, в нижній частині таблиці розташовано кнопку для додавання нового інвертору, після натискання на яку відкривається аналогічне модельне вікно.

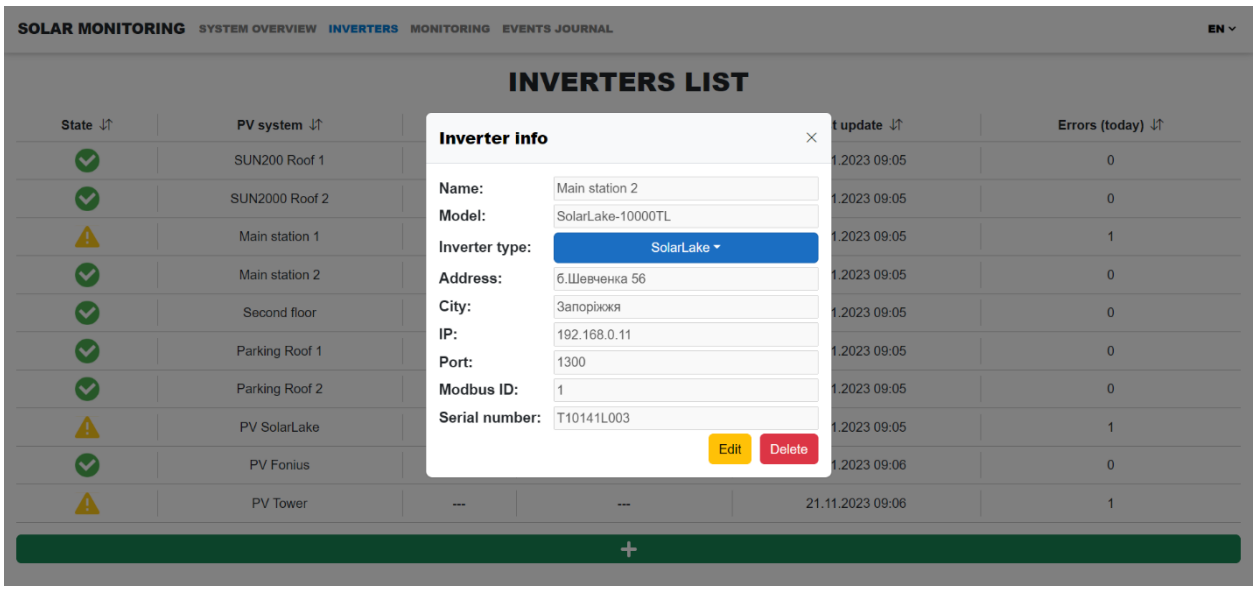

Рисунок 3.8 — Модальне вікно з інформацією про інвертор

3. Сторінка історії опитування СЕС (див. Рис. 3.9). Вона надає користувачеві список записів опитування усіх наявних інверторів. У верхній частині міститься елемент, що представляє опції фільтрації записів. Він складається з селектору статусу інвертора, поля вводу назви станції, селектору періоду створення записів та кнопки застосування фільтрів для пошуку даних за наведеними параметрами.

Основну частину сторінки займає таблиця записів опитування сонячних електростанцій та містить наступні стовпці даних:

- Статус інвертора під час опитування.
- Назва сонячної електростанції.
- Потужність системи під опитування.
- Кількість виробленої електроенергії системою за день на момент опитування.
- Кількість виробленої електроенергії СЕС за останню годину на момент опитування.
- Дата та час опитування.

В нижній частині таблиці розташовано панель пагінації, що дозоляє обирати необхідну сторінку з даними та відображає загальну кількість записів.

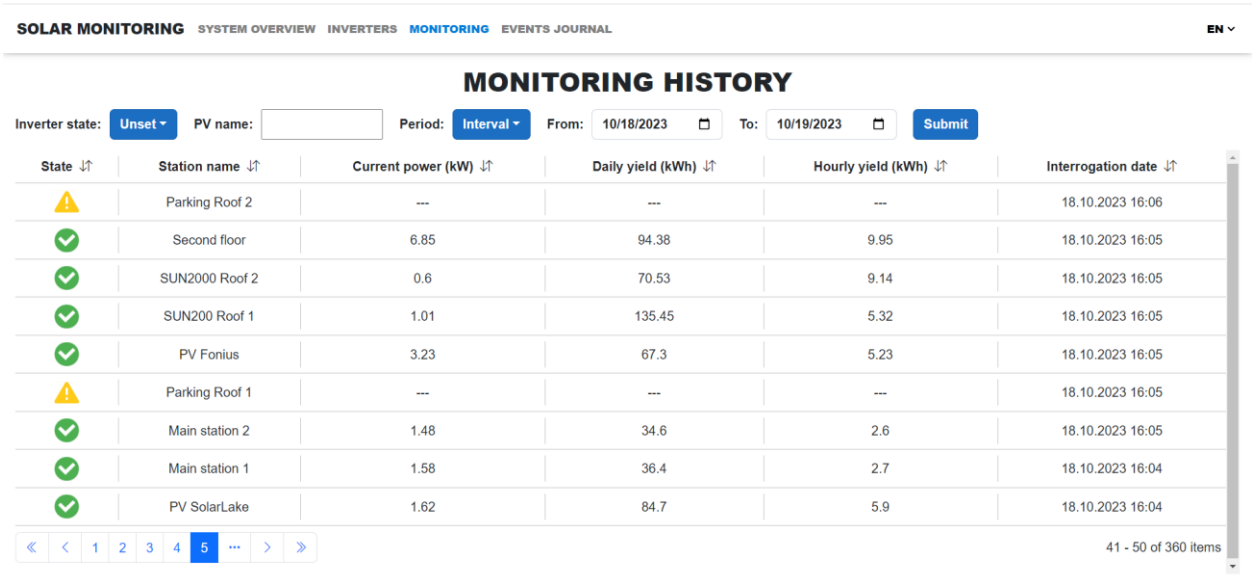

Рисунок 3.9 — Сторінка історії опитування СЕС

4. Сторінка журналу подій (див. Рис. 3.10). Вона містить список усіх не коректних статусів, які були отримані у відповідях від інверторів під час опитувань СЕС. У верхній частині міститься елемент, що містить параметри фільтрації таблиці за назвою станції та періоду створення записів.

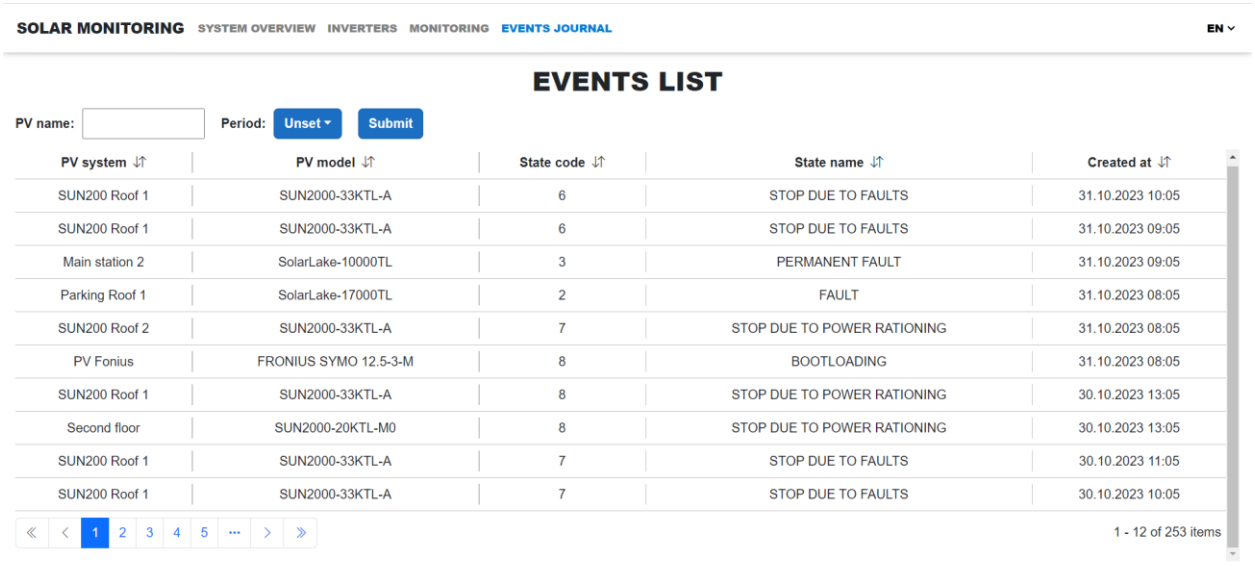

Рисунок 3.10 — Сторінка журналу подій

Основну частину сторінки займає таблиця записів помилок інверторів, що складається з наступних стовпців:

- Назва сонячної електростанції.
- Модель інвертора
- Код статусу інвертора.
- Коротка назва статусу інвертора.
- Дата створення запису.

Взаємодія клієнтського застосунку з API відбувається з допомогою HTTP запитів до відповідних контролерів. Наприклад, у випадку необхідності отримання даних щодо історії опитування інверторів, система формує JSON контент для повідомлення з моделі параметрів фільтрації таблиці та надсилає його до серверу за відповідною адресою. Після цього веб-застосунок отримує відповідь від API, десереалізує її в необхідний об’єкт та використовує його в своїй подальшій роботі. Реалізацію наведеного алгоритму відображено на лістингу 15.

Лістинг 15 — Реалізація алгоритму отримання списку записів опитування інверторів.

```
private async Task<ListResponseModel<YieldHistoryRecord>>
GetYieldHistoryRecordsList(ListRequestModel<YieldHisto-
ryRecord> listRequestModel)
{
    JsonContent content = JsonContent.Create(listRequest-
Model);

    try
    {
        string apiAddress = Configuration.GetSection("Api-
Uri").Value;

        using HttpResponseMessage response = await
s_httpClient.PostAsync($"{apiAddress}/history", content);

        if (!response.IsSuccessStatusCode)
            return new ListResponseModel<YieldHisto-
ryRecord>();

        ListResponseModel<YieldHistoryRecord>? yieldHisto-
ryRecords = await response.Content.ReadFromJsonAsync<Lis-
tResponseModel<YieldHistoryRecord>?>();
```

```
        return yieldHistoryRecords ?? new ListResponse-
Model<YieldHistoryRecord>();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex);
        return new ListResponseModel<YieldHistoryRecord>();
    }
}
```

3.7 Висновки з розділу 3

1. Було виділено функціональні та нефункціональні вимоги до системи моніторингу СЕС, спроектовано загальну архітектуру та базу даних, що задовольняє усі вимоги системи.

2. Було налаштовано середовище розробки системи з урахуванням особливостей використовуваних технологій.

3. Було створено базу даних системи та розроблено усі складові системи моніторингу СЕС: сервіс опитування інверторів з використанням протоколів Modbus RTU over TCP та Modbus RTU, а також HTTP, консольний застосунок RESTful API, та клієнтський веб-застосунок.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ПРОТОКОЛІВ КОМУНІКАЦІЇ MODBUS ПРИ ОПИТУВАННІ ІНВЕРТОРІВ

4.1 Аналіз протоколів Modbus TCP та Modbus RTU over TCP

Використовувані при розробці системи моніторингу СЕС різновиди протоколу Modbus для комунікації з інверторами мають низку особливостей, що певною мірою відрізняють їх один від одного.

Протокол Modbus TCP необхідний для обміну даними мережею Ethernet за допомогою TCP/IP та використовує стандартний метод представлення даних за протоколом Modbus. Фрейм за наведеним протоколом формується в дещо відмінному від стандартного Modbus RTU форматі. Він створюється згідно до структури TCP/IP пакету, де заголовок використовується для керування потоком даних, а дані Modbus зберігається в полі даних такого пакету. Блок даних Modbus складається з двох частин, що називаються MBAP та PDU (див. Рис. 4.1) та може складатися максимум з 258 байтів.

Рисунок 4.1 — Формат фрагменту фрейму Modbus TCP, що зберігається в полі даних пакету TCP

PDU містить дані, які необхідні для виконання певної функції або задачі, та є стандартним елементом усіх різновидів протоколу Modbus, а MBAP використовується для забезпечення вкладеності Modbus протоколу в пакеті TCP та містить наступні поля:

- Ідентифікатор транзакції, який використовується для ідентифікації повідомлення між клієнтом та сервером у випадку асинхронного надсилання повідомлень в межах одного з’єднання.
- Ідентифікатор протоколу, що завжди містить значення 0 та є зарезервованим.
- Довжина — це поле, що містить значення довжини частини повідомлення, яка розташована за ним.
- Ідентифікатор пристрою використовується для адресації окремого підлеглого пристрою в мережі. Однак, зазвичай воно дорівнює 0, через те що адресація необхідного пристрою відбувається за допомогою IP адреси в пакеті TCP.

Протокол Modbus RTU over TCP дозволяє обмінюватися повідомленнями з пристроями, що підтримують протокол Modbus RTU, мережею Ethernet за допомогою TCP/IP. За наведеним протоколом інформація між ведучим та підлеглими пристроями передається в TCP-пакеті, де в полі даних зберігається ADU протоколу Modbus RTU (див. Рис. 4.2) з максимальним розміром в 254 байти.

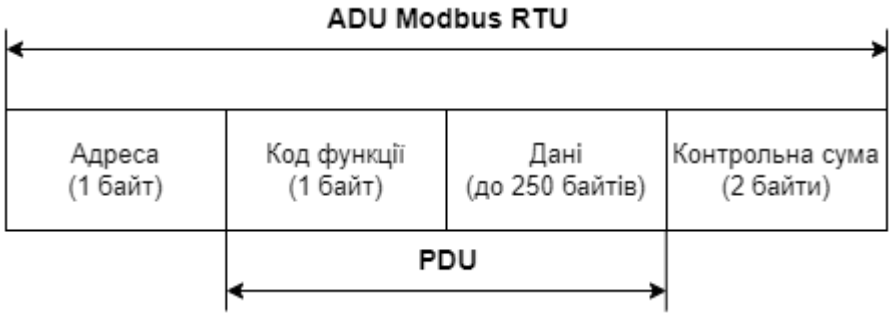

Рисунок 4.2 — Формат фрейму Modbus RTU

Такий засіб зв’язку необхідний у випадках використання перетворювачів інтерфейсів для інтеграції з мережею Ethernet пристроїв, що використовують RS-485 або RS-282, або для отримання інформації з регістраторів даних інверторів, до яких під’єднано низку пристроїв, що використовують протокол

Modbus. Фактично, цей протокол представляє собою обгортку TCP/IP над стандартним Modbus RTU.

Таким чином, проаналізувавши особливості наведених протоколів було зроблено наступні висновки:

1. Обидва протоколи використовують однакові фізичний, канальний, мережевий та транспортний рівні моделі OSI для передачі даних.

2. Фрейм Modbus TCP побудовано згідно до структури TCP/IP з використанням заголовку MBAP, характерного лише для цього протоколу. Modbus RTU over TCP, в свою чергу, зберігає формат фрейму Modbus RTU всередині TCP-пакету.

3. Адресація пристроїв за протоколом Modbus TCP виконується за допомогою IP-адрес, а Modbus RTU over TCP окрім цієї адреси використовує унікальний ідентифікатор Modbus. Це необхідно для виділення окремого пристрою у випадку використання проміжних пристроїв до яких під’єднано декілька підлеглих.

4. Обидва варіанти протоколу підтримують одні й ті ж типи даних та функціональні коди.

5. Наведені протоколи використовують TCP для забезпечення надійної передачі даних, що включає перевірку їх цілісності.

4.2 Аналіз роботи протоколів Modbus при опитуванні інверторів

Для дослідження роботи протоколів Modbus TCP та Modbus RTU over TCP було проаналізовано взаємодію реалізованої системи моніторингу сонячних електростанцій з інверторами.

Тестування проводилося з використанням інверторів від компанії Huawei, а саме з пристроями моделей SUN2000-33KTL-A та SUN2000-20KTL-M0, що підтримують Modbus RTU over TCP та Modbus TCP відповідно. Для пристрою SUN2000-33KTL-A було під’єднано перетворювач інтерфейсів

Ethernet/RS-485 фірми VKmodule моделі VTR-E/485. Сервіс опитування інверторів було розгорнуто на сервері з операційною системою Microsoft Server 2019 Standard. Усі наведені пристрої були під'єднані до однієї мережі для встановлення взаємодії між ними.

Для тестування було проаналізовано використання команди читання вмісту регістрів зберігання, що відповідає коду «03», на регістрах для вищезазначених моделей інверторів відображених в таблиці 2.

Таблиця 2 — Адреси регістрів для необхідних параметрів

Параметр	**Одиниці вимірювання**	**Адреси регістрів**	
		SUN2000-33KTL-A	SUN2000-20KTL-M0
Статус пристрою	-	32319	32000
Активна потужність	Вт	32290	32080
Вироблена енергія за поточний час	Вт·год	32298	32112
Добове виробництво енергії	Вт·год	32300	32114

Тестування включало в себе аналіз повідомлень з командами ведучого пристрою (серверу) та відповідями підлеглих пристроїв (інверторів), аналіз безперебійності роботи протоколів, аналіз роботи протоколів при нормальних умовах та аналіз роботи при надсиланні некоректних повідомлень з допомогою аналізатору трафіку комп'ютерних мереж Wireshark.

Аналіз повідомлень Modbus відбувався шляхом надсилання команд на інвертори та перегляду характеристик і структури надісланих фреймів. При аналізі фреймів Modbus TCP було виявлено, що ці повідомлення розпізнаються, як окремий протокол (див. Рис. 4.3). Повідомлення запитів містять 66 бітів, а відповіді від 65 до 67 бітів в залежності від значення, що повертається.

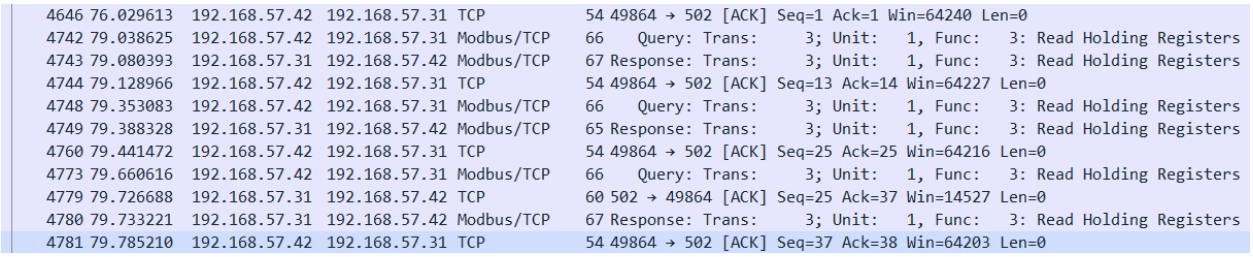

Рисунок 4.3 — Мережевий трафік при опитувані інвертору за протоколом Modbus TCP

Для аналізу структури повідомлення було обрано команду читання регістрів зберігання підлеглого пристрою для отримання значення активної потужності. Отже, з серверу було надіслано команду «03» з адресою регістру 32080 та параметром кількості байтів зі значенням 2. В результаті аналізу було виявлено, що замість типового поля «Data» відображається «Modbus/TCP», що представляє ADU цього протоколу (див. Рис. 4.4), розбір якого було виконано в пункті 4.1. PDU частина повідомлення, що підписана «Modbus» на рисунку 4.4, містила усі необхідні дані, які були задані сервером.

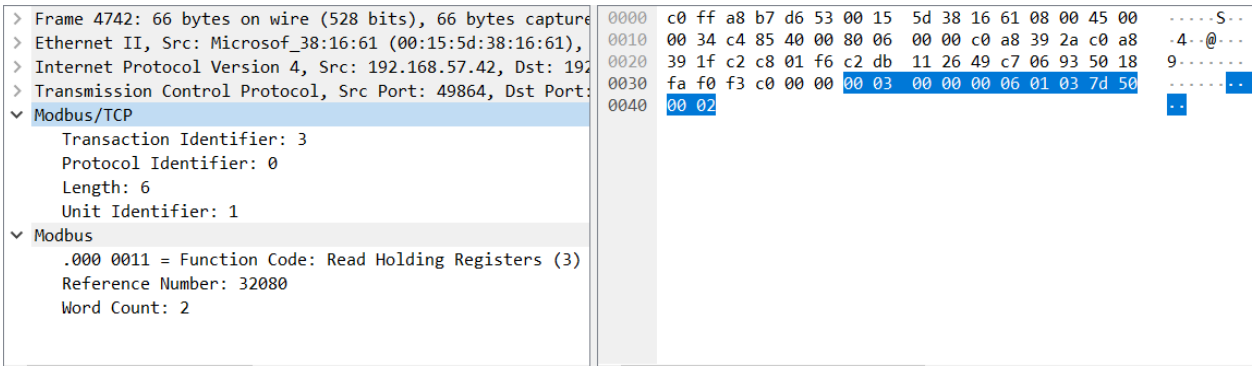

Рисунок 4.4 — Повідомлення з командою читання регістрів зберігання за протоколом Modbus TCP

Відповідь на таке повідомлення має дещо іншу структуру. Відмінність в них полягає у різниці в PDU, який складається з наступних частин (див. Рис. 4.5):

- Код функції (займає 1 байт). Він співпадає з тією функцією, для якої було сформовано повідомлення.

- Кількість байтів (займає 1 байт), що відповідає розміру даних необхідного значення параметру.
- Значення регістрів, розміром в 2 байти кожен, що в сукупності представляють собою значення параметру, який зберігається в них.

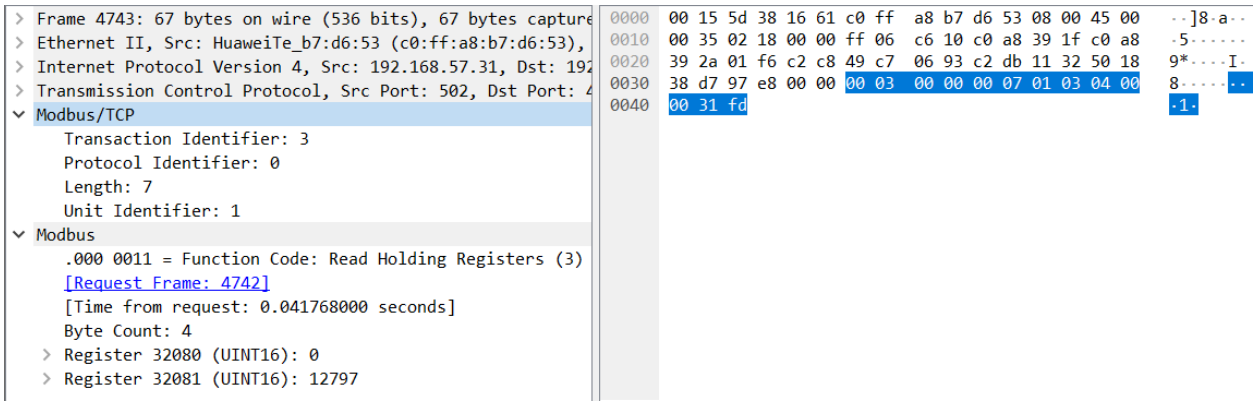

Рисунок 4.5 — Повідомлення з відповіддю на команду читання регістрів збереігання за протоколом Modbus TCP

Аналіз повідомлень Modbus RTU over TCP було проведено аналогічним чином. Спочатку було зафіксовано мережевий трафік з повідомленнями під час опитування інвертора (див. Рис. 4.6) та виділено декілька особливостей. По-перше, фрейми не ідентифікуються, як окремий протокол, а відображаються у вигляді TCP-повідомлень, що вказує на обгортання Modbus RTU повідомлення в TCP-пакет. По-друге, повідомлення запитів містять 62 біти, а відповіді від 61 до 63 бітів в залежності від значення, що повертається. Тобто, в середньому, вони менші на 4 біти, аніж аналогічні повідомлення Modbus TCP.

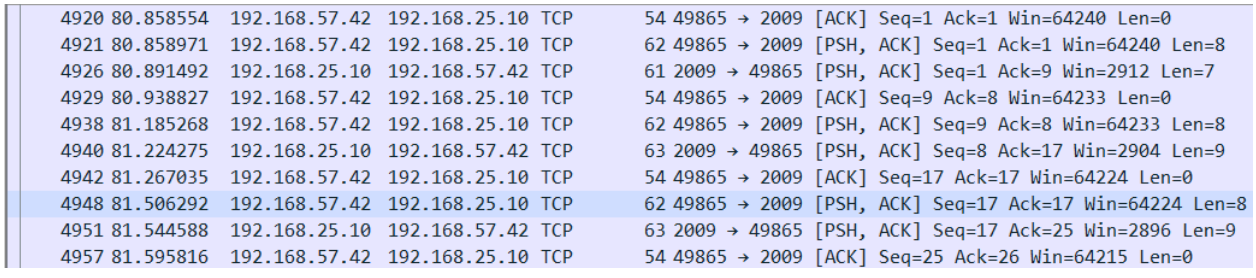

Рисунок 4.6 — Мережевий трафік при опитувані інвертору за протоколом Modbus RTU over TCP

Для аналізу структури повідомлення було обрано таку ж команду читання регістрів зберігання підлеглого пристрою для отримання значення активної потужності, але з іншою адресою необхідної комірки пам’яті 32290. В результаті аналізу було виявлено, що в полі даних пакету знаходиться, фрейм Modbus RTU, що представляє ADU цього протоколу (див. Рис. 4.7) та значно відрізається від Modbus TCP за розміром. Це і визначає розбіжності в середній кількості бітів повідомлень цих протоколів.

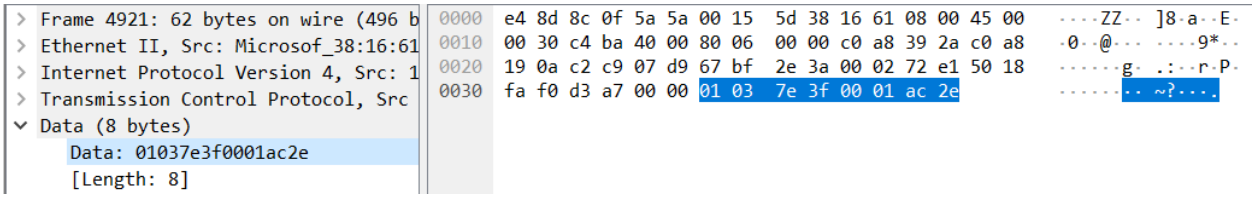

Рисунок 4.7 — Повідомлення з командою читання регістрів зберігання за протоколом Modbus RTU over TCP

Відповідь на повідомлення від ведучого пристрою також має структуру, що відрізняється від запиту. Відмінність полягає у різниці в даних PDU, а саме повідомлення Modbus в цьому випадку складається з наступних частин (див. Рис. 4.8):

- Ідентифікатор Modbus (займає 1 байт). Він співпадає з власним ідентифікатором пристрою.
- Код функції (займає 1 байт), що повторює код у запиті.
- Кількість бітів даних (займає 1 байт).
- Значення регістрів, розміром в 1 байт, що в сукупності представляють собою значення параметру, який зберігається в них.
- Контрольна сума (займає 2 байти).

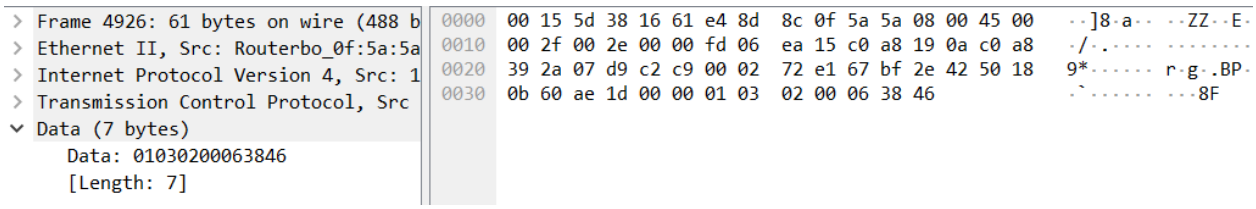

Рисунок 4.8 — Повідомлення з відповіддю на команду читання регістрів збереігання за протоколом Modbus RTU over TCP

Окрім цього було проаналізовано безперебійність роботи системи при надсиланні повідомлень за наведеними протоколами при нормальних умовах. Для наведеного тестування було запущено сервіс з опитування інверторів, який безперервно працював на сервері Windows Server 2019 Standard протягом тижня. Також, до цього програмного застосунку було додано відповідний функціонал для фіксації лог-повідомлень у разі виникнення помилок при комунікації з інверторами за використовуваними протоколами Modbus. За результатами роботи сервісу не було виявлено жодних помилок при комунікації з інверторами за протоколами Modbus, що свідчить про їх високу надійність та безперебійність. Однак, при використанні Modbus TCP час відправки та прийому повідомлення варіювався між 7 та 12 мілісекундами, в той час, як наведений показник для Modbus RTU over TCP дорівнював від 15 до 23 мілісекунд.

Останнім етапом було проведено аналіз роботи протоколів при надсиланні на підлеглі пристрої некоректних повідомлень. Для цього код сервісу опитування інверторів було модифіковано таким чином, щоб він надсилав пристроям не існуючі адреси регістрів або коди функцій. При аналізі відповідей від інверторів за протоколами Modbus TCP (див. Рис. 4.9) та Modbus RTU over TCP (див. Рис. 4.10) було виявлено, що PDU в обох видах повідомлень не відрізняється за змістом та форматом і складається з двох частин:

- Код функції запиту, який призвів до виникнення винятку. Для цього параметру старший біт встановлюється на початку значення.

- Код помилки. При тестуванні було отримано коди 01 — «Непідтримуваний код функції» та 03 — «Вказана адреса даних не визначена на підлеглому пристрої».

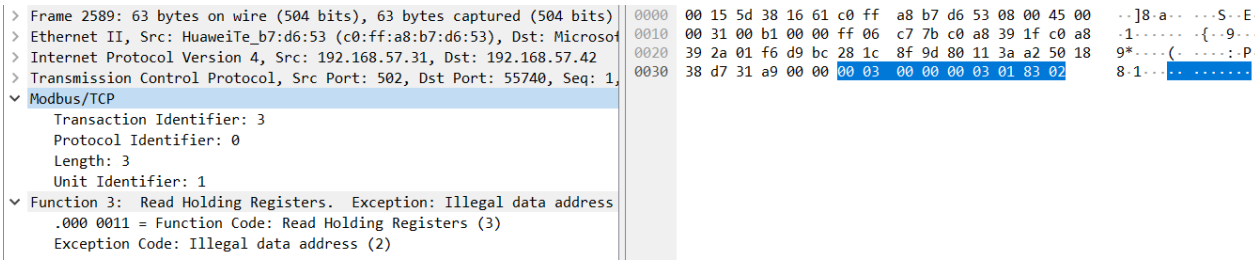

Рисунок 4.9 — Відповідь від підлеглого пристрою з помилкою за протоколом Modbus TCP

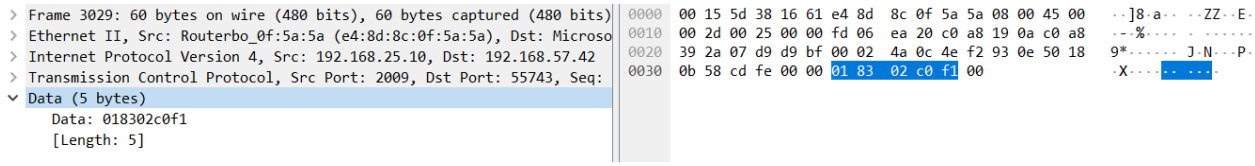

Рисунок 4.10 — Відповідь від підлеглого пристрою з помилкою за протоколом Modbus RTU over TCP

Результати аналізу роботи обох протоколів при опитуванні інверторів відображено в таблиці 3.

Таблиця 3 — Характеристики протоколів Modbus TCP та Modbus RTU over TCP

Характеристика	**Modbus TCP**	**Modbus RTU over TCP**
Формат повідомлення	TCP-пакет з ADU Modbus TCP в полі даних	TCP-пакет з ADU Modbus RTU в полі даних
Адресація	Через IP-адреси (пристрої Modbus TCP зазвичай не мають унікальних Modbus адрес)	Через IP-адреси, але пристрої мають унікальні ідентифікатори Modbus всередині мережі

Характеристика	Modbus TCP	Modbus RTU over TCP
Цілісність даних	Забезпечується протоколом TCP	Забезпечується протоколом TCP та Modbus RTU (CRC)
Середній розмір повідомлення	66 байтів	62 байти
Розмір ADU (максимальний)	258 байт	254 байти
Час між відправкою та прийомом повідомлень	7-12 мс	15-23 мс

4.3 Висновки з аналізу роботи протоколів Modbus TCP та Modbus RTU over TCP

1. Обидва протоколи Modbus використовують Ethernet та TCP/IP для передачі даних, що забезпечує зручність та ефективність їхнього використання в сучасних мережах.

2. Modbus TCP використовує структуру кадру із заголовком MBAP, що забезпечує сумісність із TCP/IP, а Modbus RTU over TCP зберігає формат Modbus RTU всередині пакету TCP.

3. Протокол Modbus TCP використовує адресацію за допомогою IP-адрес, в той час як Modbus RTU over TCP окрім цього також потребує унікальні адреси Modbus RTU. Це певною мірою ускладнює налаштування мережі при використанні інверторів, що підтримують Modbus RTU через необхідність налаштування їх самих та проміжних пристроїв, які забезпечують під’єднання до загальної мережі.

4. В реалізованій системі Modbus TCP показав себе як більш ефективний протокол для Ethernet-мереж, що підтримує великі обсяги даних та високу швидкість їх передачі. В свою чергу, Modbus RTU over TCP є невід’ємним рішенням при інтеграції з обладнанням, підтримуючим лише протокол Modbus RTU.

ВИСНОВКИ

В результаті проведеного дослідження було визначено, що існуючі програмні системи моніторингу сонячних електростанцій для своєї роботи потребують встановлення додаткового обладнання або працюють лише з одним типом інверторів. Це призводить до додаткових витрат компанії та необхідності використовувати різні застосунки моніторингу для окремих СЕС. Окрім цього, був визначений ряд недоліків та переваг досліджуваних систем.

Під час вивчення існуючих технологій для реалізації системи моніторингу було розглянуто протоколи комунікації інверторів, існуючі бібліотеки для реалізації взаємодії з ними, фреймворк для розробки серверного застосунку та популярні системи управління базами даних. Окрім цього, було проведено аналіз можливості створення клієнтської десктопної та веб-програми, а також проаналізовано інструмент для розробки веб-застосунку. Засобами для реалізації системи були обрані: .NET, C#, PostgreSQL та Blazor.

Виходячи з цього, було розроблено інформаційну систему для моніторингу СЕС, що може працювати з різними видами інверторів, враховує слабкі та сильні сторони аналогів, а також потребує підключення інверторів до локальної мережі без встановлення додаткового обладнання для збору даних.

Окрім цього, на базі розробленої системи було проведене дослідження роботи використовуваних протоколів Modbus TCP та Modbus RTU over TCP при опитуванні інверторів. В результаті проведеного дослідження було визначено особливості цих протоколів, проаналізовано їх сильні та слабкі сторони і зроблено висновок щодо ефективності та доцільності їхнього використання для реалізації інформаційної системи моніторингу сонячних електростанцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Antonios Makris, Konstantinos Tserpes, Giannis Spiliopoulos, Dimitrios Zissis, Dimosthenis Anagnostopoulos. MongoDB Vs PostgreSQL: A comparative study on performance aspects. GeoInformatica. 2021. Vol. 25. P. 243-268.

2. Chris Sanity. Blazor in Action. New York : Manning Publications, 2022. 380 p.

3. Connectivity libraries and frameworks for Microsoft SQL Server. Microsoft : веб-сайт. URL: https://learn.microsoft.com/en-us/sql/linux/sql-server-linux-develop-connectivity-libraries?view=sql-server-ver16 (дата звернення 26.11.2023).

4. Găitan V., Zagan I. Modbus protocol performance analysis in a variable configuration of the physical Fieldbus architecture. *IEEE Access*. 2022. Vol. 10. P. 123942–123955.

5. Hans-Jurgen Schonig. Mastering PostgreSQL 13: Build, administer, and maintain database applications efficiently with PostgreSQL 13. Birmingham : Packt Publishing, 2020. 476 p.

6. Herath H.M.K.K.M.B., Ariyathunge S.V.A.S.H., Priyankara H.D.N.S. Development of a data acquisition and monitoring system based on Modbus RTU communication protocol. *International Journal of Innovative Science and Research Technology*. 2020. Vol. 5, No 6. P. 433–440.

7. Joel Murach, Bryan Syverson. Murach's SQL Server 2019 for Developers. Fresno : Mike Murach & Associates, 2019. 674 p.

8. Jon P Smith. Entity Framework Core in Action. New York : Manning, 2018. 520 p.

9. Michele Aponte. Building Single Page Applications in .NET Core 3. Berkley : Apress, 2020. 116 p.

10. Modbus Interface Definitions (V2.0). Huawei : веб-сайт. URL: https://support.huawei.com/view/PdfRead/EDOC1100050690/SUPE_DOC/6001/document.pdf (дата звернення 26.11.2023).

11. Protocol Description. Modbus tools : веб-сайт URL: https://www.modbustools.com/modbus.html (дата звернення 26.11.2023).

12. Remarkable Innovation. Global Reach. APsystems : веб-сайт. URL: https://apsystems.com/ (дата звернення 26.11.2023).

13. Sherzod Elamanov, Hyeonseo Son, Bob Flynn, Seong Ki Yoo, Naqqash Dilshad, JaeSeung Song. Interworking between Modbus and internet of things platform for industrial services. *Digital Communications and Networks*. 2022. Vol. 9, No 5. P. 14-28.

14. Silvia Botros, Jeremy Tinley. High Performance MySQL: Proven Strategies for Operating at Scale. Sebastopol : O'Reilly Media, 2021. 388 p.

15. Sineglazov V. M., Daskal E. V. Wireless solar power plant monitoring system. *Electronics and Control Systems*. 2018. № 3 (57). С. 54–57.

16. V. G. Găitan and I. Zagan. Experimental implementation and performance evaluation of an IoT access gateway for the modbus extension. Sensors. 2021 Vol. 21, No 1. P. 246.

17. What is a solar inverter and how does it work? Fallon solutions : веб-сайт. URL: https://www.fallonsolutions.com.au/solar/information/what-is-a-solar-inverter-and-how-does-it-work (дата звернення 26.11.2023).

18. What's new in .NET 6. Microsoft : веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-6 (дата звернення 11.12.2022).

19. Xia K., Ni J., Ye Y., Xu P., Wang Y. A real-time monitoring system based on ZigBee and 4G communications for photovoltaic generation. *CSEE Journal of Power and Energy Systems*. 2020. Vol. 6, No. 1. P. 52–63.

20. Близько 3,5 тис. домогосподарств встановили сонячні електростанції у II кварталі. Держенергоефективності : веб-сайт. URL: https://saee.gov.ua/uk/news/3918 (дата звернення 26.11.2023).

21. Дзядикевич Ю.В., Буряк М.В., Любезна І.В. Розвиток сонячної енергетики в Україні. *Інноваційна економіка*. 2018. № 1–2 (73). С. 120–125.

22. Кицмен Х.Р., Стрілець Х.В. Використання сонячної енергії як передумова сталого розвитку природи та суспільства. *Відновлювана енергетика та енергоефективність у XXI столітті* : матеріали міжнар. наук.-практ. конф., м. Київ, 15-16 трав. 2019 р. Київ, 202019. С. 705–709.

23. Лісовець С. М., Коляда Д. М. Застосування комунікаційного протоколу Modbus для обміну даними між елементами систем автоматизації. *Технології та дизайн*. 2016. № 2 (19). С. 1–7.

24. Моніторинг фотовольтаїчної системи на порталі Fronius Solar.Web. Fronius : веб-сайт. URL: https://www.fronius.com/uk-ua/ukraine/sonyachna-enerhiya/instalyatory-ta-partnery/produkty-ta-rishennya/tsyfrovi-dodatky-ta-zasoby-monitorynhu/monitorynh-fotovoltayichnoyi-systemy-na-portali-fronius-solarweb#anc_get-started (дата звернення 26.11.2023).

25. Рижков В.А., Міхайлуца О.М. Аналіз інструментів для створення серверної частини системи моніторингу сонячних електростанцій. *Молода наука-2023* : матеріали XVI унів. наук.-практ. конф. студентів, аспірантів, докторантів і молодих вчених. Запоріжжя : ЗНУ, 2023. Т. 5. С. 116-117.

26. Рижков В.А., Міхайлуца О.М. Аналіз фреймворку для створення клієнтської частини системи моніторингу сонячних електростанцій. *«Актуальні питання сталого науково-технічного та соціально-економічного розвитку регіонів України»* : матеріали III Всеукр. наук.-практ. конф. за участі молодих вчених. Запоріжжя : Запорізький національний університет, 2023. С. 135-137.

27. Рижков В.А., Міхайлуца О.М. Комп’ютерна система моніторингу сонячних електростанцій з можливістю опитування різних типів інверторів. *«Green Construction»* : матеріали II міжнар. наук.-практ. конф., м. Київ, 13-14 квіт. 2023р. Київ, 2023. С. 352-353.

28. Рубаненко О. Є., Гунько І. О., Рубаненко О. О. Дослідження системи моніторингу параметрів режиму роботи сонячної панелі. *Техніка, енергетика, транспорт АПК*. 2018. № 1 (100). С. 91–98.

29. Сікомас А.Ю., Матвійчук І.І., Мартинюк В.В. Система моніторингу сонячних панелей у реальному масштабі часу. *Відновлювана енергетика та*

енергоефективність у XXI столітті : матеріали міжнар. наук.-практ. конф., м. Київ, 14-15 трав. 2020 р. Київ, 2020. С. 702–707.

30. Соколовський О. Ф., Поліщук П. А. Моніторинг фотоелектричних систем. *Біоенергетичні системи* : матеріали міжнар. наук.-практ. конф., м. Житомир, 29 трав. 2020 р. Житомир, 2020. С. 158–161.

31. Чому Sunpower? MBI : веб-сайт. URL: https://mbidevelop.com/sunpower (дата звернення 26.11.2023).

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Рижков Валерій Анатолійович, студент 2 курсу, денної форми навчання, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти valur010@gmail.com,

- підтверджую, що написана мною кваліфікаційна робота на тему: «**Комп'ютерна система моніторингу та оптимізації роботи сонячних електростанцій**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений;

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою Інтернет - системи, в також архівування моєї роботи у базі даних цієї системи.

Дата 30.11.2023 Підпис _____ Рижков Валерій Анатолійович
(студент)

Дата 30.11.2023 Підпис _____ Міхайлуца Олена Миколаївна
(науковий керівник)