НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 122 «Комп'ютерні науки»

освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Створення програмного засобу для моделювання динамічних режимів роботи сонячних панелей в складі мікроенергостанції»

Виконав (-ла):
студент (-ка) IV курсу, групи ТМ-72
Юрченко Ростислав Олександрович _____

Керівник:
д.т.н, професор
Отрох Сергій Іванович _____

Консультант:
_____

Рецензент:
д.т.н, професор
Жураковський Богдан Юрійович _____

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

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

Юрченко Ростиславу Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Створення програмного засобу для моделювання динамічних режимів роботи сонячних панелей в складі мікроенерго-станції

керівник роботи д.т.н, проф. Отрох Сергій Іванович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи мова програмування Java, середовище розробки Android Studio

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Дослідити та проаналізувати існуючі рішення для розрахунків режимів роботи сонячних електростанцій, описати програмну реалізацію програмного продукту, розробити продукт та інтерфейс користувача для нього.

5. Перелік ілюстративного матеріалу

Назва роботи, вступ, постановка задачі, огляд існуючих рішень, діаграма прецедентів, засоби розробки, опис бази даних, структура програмного забезпечення, головне меню, вікно карти, вікно розрахунку кута нахилу, вікно розрахунку електронних приладів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "_9_"____жовтня_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	09.10.2020	
2.	Вивчення та аналіз задачі	10.10.2020-20.12.2020	
3.	Розробка архітектури та загальної структури системи	10.02.2021-14.03.2021	
4.	Розробка структур окремих підсистем	14.03.2021-10.04.2021	
5.	Програмна реалізація системи	11.04.2021-06.05.2021	
6.	Оформлення пояснювальної записки	03.05.2021-02.06.2021	
7.	Захист програмного продукту	12.05.2021	
8.	Передзахист	24.05.2021	
9.	Захист	14.06.2021	

Студент _____ _____Юрченко Р. О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Отрох С. І._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи: розробити систему для розрахунків обсягів необхідної для генерування сонячної енергії на базі енергетичного потенціалу місцевості та визначення необхідних характеристик обладнання для сонячних мікроелектростанцій. Перевагами має бути доступність, зрозумілість та можливість використання додатку при відсутності інтернету.

Було проаналізовано наявні програмні рішення, виявлено їх переваги та недоліки. На їх основі обґрунтовано задачі та критерії, яким повинна відповідати система.

Результати бакалаврської роботи були апробовані у XIX міжнародній науково-практичній конференції молодих вчених та студентів «Сучасні проблеми наукового забезпечення енергетики».

Обсяг дипломної роботи складає 54 сторінок, 27 рисунків, 13 використаних літературних джерел, та 4 додатка.

Ключові слова: сонячні панелі, мобільний додаток, Android, Java, сонячна енергетика.

ABSTRACT

Purpose: to develop a system for calculating the solar energy amount needed to generate, based on the energy potential of the area and determine the necessary characteristics of solar power plant equipment.

The existing software solutions were analyzed; their advantages and disadvantages were identified. Based on them, the tasks and criteria that the system must meet are substantiated.

The volume of the thesis is 54 pages, 27 pictures, 13 sources of literature, and 4 appendices.

Keywords: solar panels, mobile application, Android, Java, solar energy.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СЕС — Сонячні електростанції

СКБД — Система керування базами даних

ОЗП — Оперативний запам'ятовувальний пристрій

ОС — Операційна система

ККД — Коефіцієнт корисної дії

АКБ — Акумуляторні батареї

NASA — National Aeronautics and Space Administration

IDE — Integrated Development Environment

SDK — Software Development Kit

JVM — Java Virtual Machine

APK — Android Package

API — Application Programming Interface

JSON — JavaScript Object Notation

XML — eXtensible Markup Language

HTML — HyperText Markup Language

ВСТУП

Питання енергозабезпечення є актуальним для мешканців України. Значною є проблема забезпечення країни твердим паливом для потреб ТЕС та ТЕЦ. Збільшення витрат бюджету на імпорт як ресурсів для електрозабезпечення, так і самої електроенергії призводить до підвищення тарифів на послуги електропостачання для населення. Складно спрогнозувати динаміку цін у майбутньому, тому все частіше люди шукають способи заощадити кошти.

Використання “зеленої енергетики” в Україні росте в експоненційному темпі. Все більше і більше людей переходять на альтернативні джерела енергії для власного використання, такі як сонячна. Її перевагою є відносна незалежність від ситуації на ринку електроенергії та можливість економити кошти. Ще одним фактором є існування в Україні «Зеленого тарифу» [1], що дозволяє приватним домогосподарствам продавати надлишки виробленої й сонячного випромінювання та/або вітру енергії за вищою ціною, ніж ринкова.

Сонячна енергетика має низку переваг [2], а саме:

- надійність та довговічність — сонячні панелі здатні працювати досить довго і термін їх роботи вимірюється десятками років. Більшість моделей має гарантію понад 25 років, при незначній втраті потужності (до 20%).
- екологічність — відсутні будь-які викиди, не становить шкоди ні людині, ні навколишньому середовищу.
- безшумність — одною з причин довговічності батарей є відсутність рухомих частин, що потребували б заміни та генерували б зайвий звук.
- можливість корисного використання площі даху будівлі.
- заощадження коштів у довгостроковій перспективі.

Метою роботи є створення програмного продукту для роботи з сонячними батареями, за допомогою якого користувач має можливість отримати інформацію про

ефективність використання сонячних панелей саме для нього та розрахунку оптимальної кількості обладнання для забезпечення його потреб.

Для розробки було обрано середовище розробки Android Studio 4.2.2, мову програмування Java, мову розмітки XML, набір засобів розробки Google Maps SDK. Використано систему керування базами даних SQLite. Також для розробки використовувалась операційна система Windows 10 Pro.

Розділи пояснювальної записки містять наступну інформацію: перший розділ містить постановку задачі; другий розділ складається з опису предметної області та аналізу існуючих рішень; третій розділ описує засоби програмної реалізації продукту; четвертий розділ пояснює роботу алгоритм роботи програми; в п’ятому розділі продемонстровано роботу користувача з додатком.

1. ПОСТАНОВКА ЗАДАЧИ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОДЕЛЮВАННЯ ДИНАМІЧНИХ РЕЖИМІВ РОБОТИ СОНЯЧНИХ ПАНЕЛЕЙ В СКЛАДІ МІКРОЕНЕРГОСТАНЦІЙ

Метою дипломної роботи є створення програмного засобу для розрахунків обсягів генерування сонячної енергії на основі географічного розташування та енергетичних характеристик обладнання сонячних електростанцій. Отримана інформація має відображати ефективність генерування сонячної енергії у даній місцевості та вказувати характеристики приладдя, для задоволення потреб користувача.

Постанова задачі роботи:

1. Створення системи для отримання інформації про геолокацію.
2. Створення системи для обчислення енергетичного потенціалу місцевості.
3. Розробка системи орієнтування мобільного приладу у просторі на базі вбудованого гіроскопу та акселерометра.
4. Створення бази даних приладів та обладнання, а також їх характеристик.
5. Розрахувати дефіцит енергії у системі
6. Розробити зрозумілий інтерфейс для користувача.

Створений додаток має працювати автономно, без потреб у постійному інтернет зв’язку.

2. АНАЛІЗ ПРОБЛЕМИ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ СОНЯЧНИХ ПАНЕЛЕЙ

У ході аналізу проблеми ефективності використання сонячних панелей було з’ясовано, що для функціонування системи потрібно, щоб профіцит отриманої сонячної енергії перевищував дефіцит. Отже, програма має містити одразу два модулі: перший — для розрахунку енергетичного потенціалу місцевості та підрахунку значень отриманої енергії (з урахуванням втрат); другий — для підрахунку спожитої енергії, мінімальне значення, що має задовольняти система генерації сонячної енергії.

2.1 Огляд існуючих програмних рішень

Питання сонячної енергетики для населення — не нове. Все стрімкіше зростає попит на неї серед українського населення [3]. Аналоги, які б дозволяли отримати інформацію про сонячні панелі, вже існують на ринку. Розглянемо деякі приклади таких систем.

2.1.1 Додаток “EasySolarCalculator”

Додаток EasySolarCalculator — безкоштовний додаток з крамниці застосунків Google Play для розрахунку даних для автономних систем на базі сонячних батарей (рисунок 2.1). Калькулятор використовує дані про потребу користувача в енергії або навантаження на систему та на її основі видає характеристики обладнання, які повинні задовільнити дану систему.

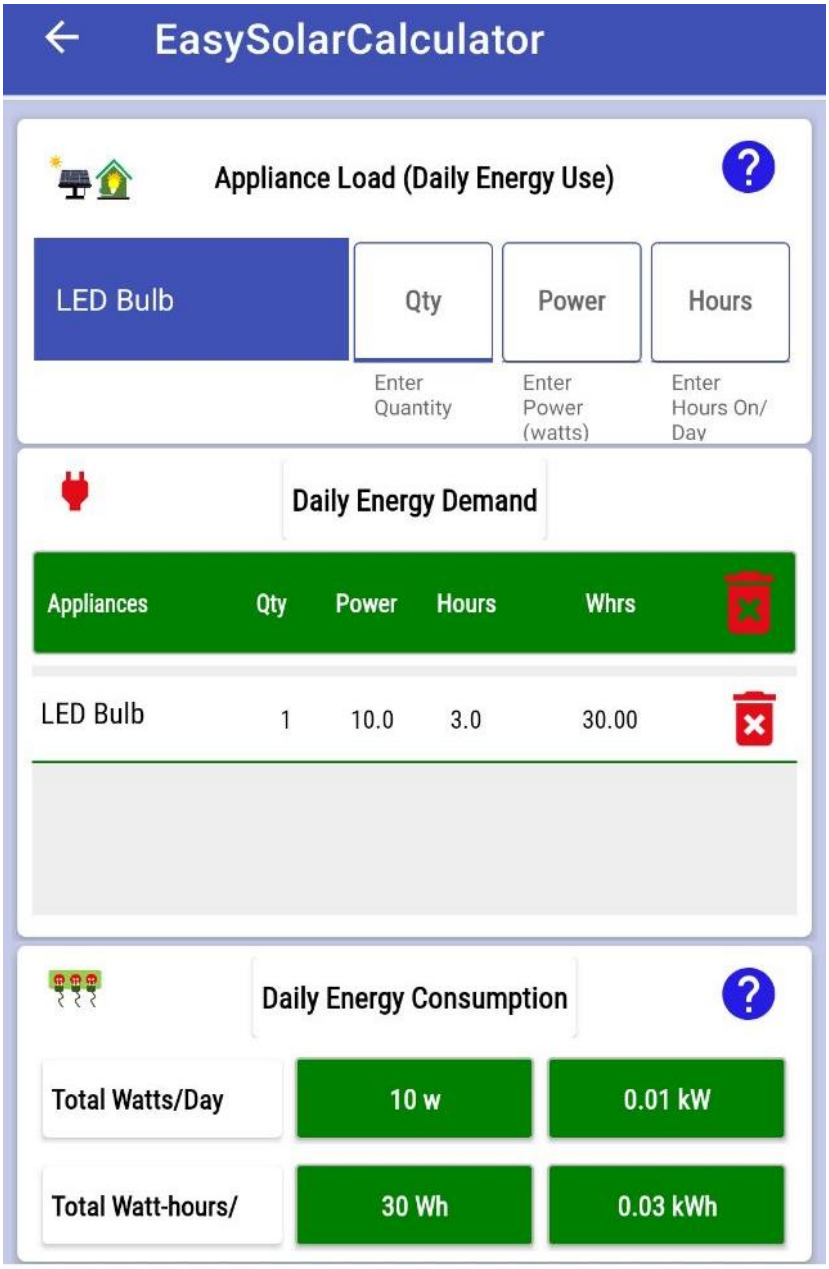

Рисунок 2.1 — Інтерфейс додатку My Solar Panel Lite

Програма спроможна обчислити наступний перелік даних:

- Напругу в мережі
- Об'єм акумулятора
- Показники інвертора
- Втрати при передачі енергії.

Дана інформація є корисною для користувача, бо дозволяє побачити, на чому ґрунтується ті, або інші характеристики приладів. Але в той самий час було виявлено низку вад даного продукту (рисунок 2.2).

По-перше, вся інформація про отриману сонячну енергію базується лише на даних введених користувачем (а саме кількість сонячних годин). Самостійно звичайний користувач не в змозі самостійно надати таку інформацію. Вона буде або неточною, що робить програму ненадійною, або користувач повинен самостійно шукати інформацію, що не є зручним і ускладнює процес отримання інформації.

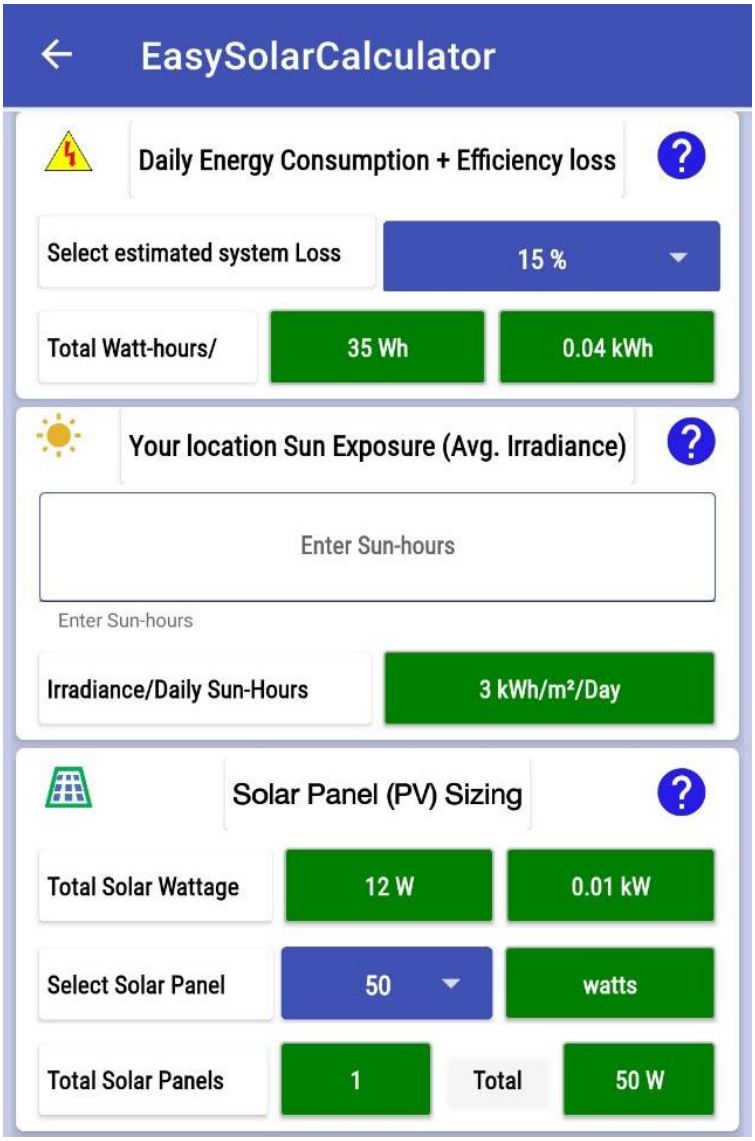

Рисунок 2.2 — Демонстрація проблеми (Ручний ввід кількості сонячних годин)

По-друге, додаток не дає ні інформації, ні рекомендацій щодо розміщення сонячних панелей, не враховує зміну кута нахилу протягом року. Для екваторіальний країн це не є проблемою, але для країн у помірних широтах (як Україна), це питання є досить важливим.

До незначних недоліків можна віднести підтримку лише англійської мови (що, тим не менш, не впливає на оцінку програми).

Отже, зважаючи на все вищесказане, можна зробити висновок, що програма добре обчислює дефіцит енергії в системі, а з розрахунком профіциту в неї є проблеми.

2.1.2 Додаток “My Solar Panel Lite”

Додаток “My Solar Panel Lite” — мобільна програма для підрахунку сонячного випромінювання для сонячних панелей. Додаток дозволяє отримати детальну інформацію про сонячний потенціал місцевості (рисунок 2.3).

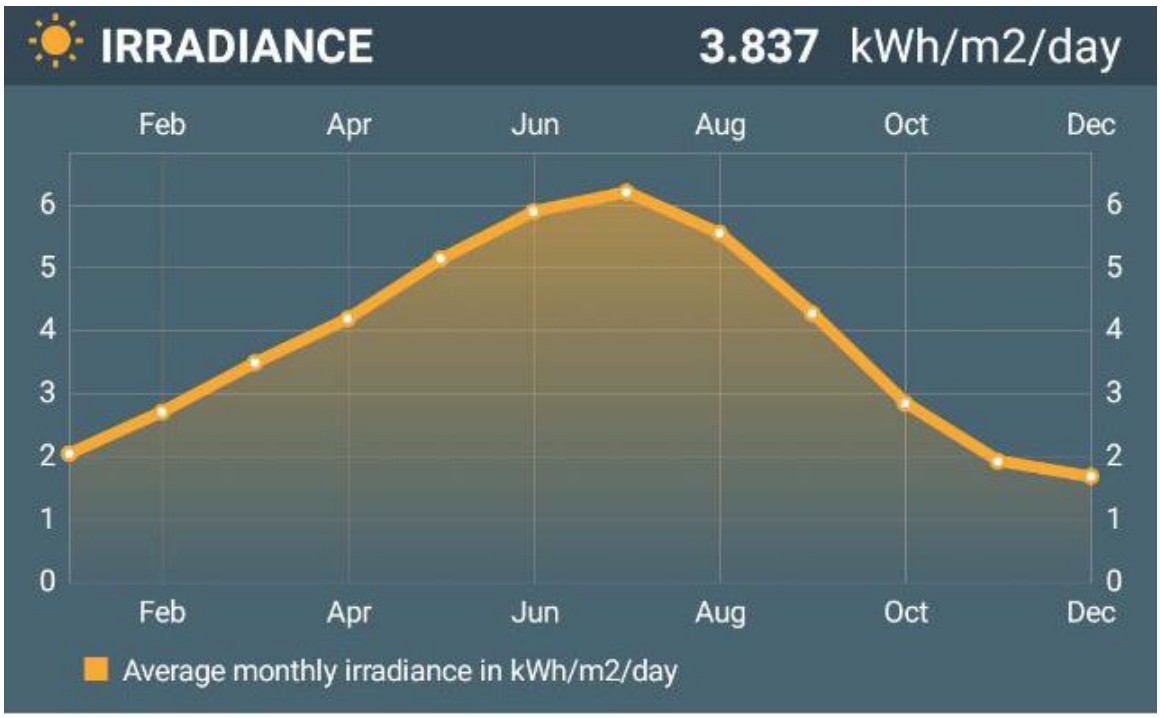

Рисунок 2.3 — Річний графік інсоляції місцевості

Додаток дає змогу отримати широкий спектр інформації (рисунок 2.4), а саме:

- Місячну інсоляцію (величину сонячного випромінювання)
- Щомісячну та щорічні значення добутої енергії
- Необхідна площа панелей
- Коефіцієнт використання встановленої потужності

- Період окупності
- Деградація панелей
- Вплив на навколишнє середовище

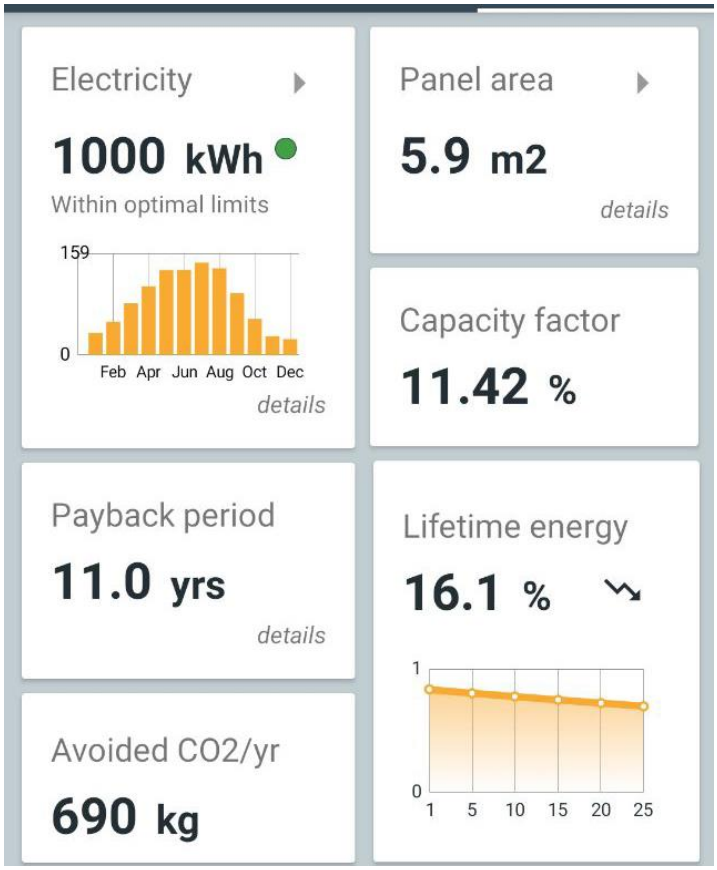

Рисунок 2.4 — Аналіз місцевості у додатку “My Solar Panel Lite”

Наведена інформація є істотною і може знадобитися, але її характер підходить більше для підприємств, ніж для звичайного користувача. Такий надлишок інформації може спантеличити і її необхідність є сумнівною. Також, даний додаток відображає лише отриману енергії, а спожиту користувач має розрахувати сам, що може призводити до похибок у розрахунках.

2.1.3 Веб-додаток “Real Solar”

Real Solar — веб-застосунок, що використовує дані про сонячну інсоляцію надану серверами NASA, які вважаються одними з найбільш достовірними. Калькулятор використовує дані про географічне положення на місцевості, звіряє їх з даними NASA та на їх основі визначає сонячний потенціал місцевості. Після цього,

використовуючи дані про положення та кут нахилу сонячних панелей визначає кількість енергії, що генерується (рисунок 2.5).

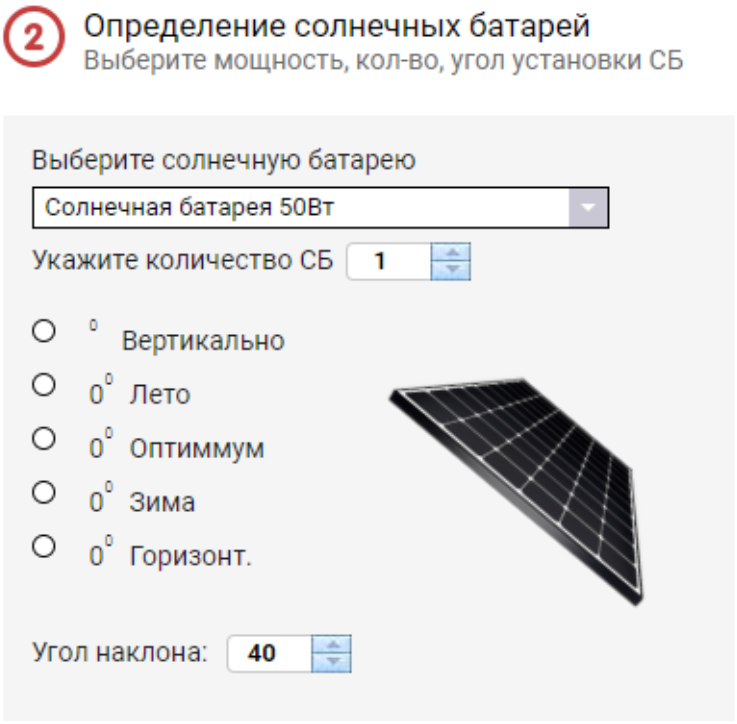

Рисунок 2.5 — Меню налаштування сонячних панелей у “Real Solar”

Дані про інсоляцію корегуються даними про кут нахилу сонячних панелей і результат розрахунків виводиться у вигляді графіку. На ньому вказані середньомісячні показники кількості енергії, яка потенційно буде генеруватися (рисунок 2.6).

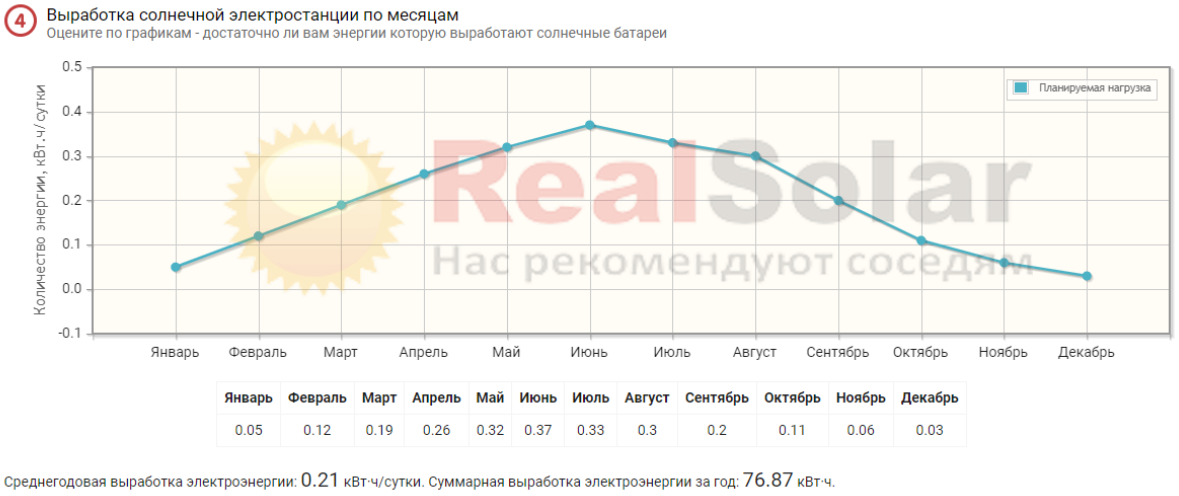

Рисунок 2.6 — Графік виробленої енергії сонячною електростанцією

Також на веб-сайті присутній калькулятор дефіциту сонячної енергії (рисунок 2.7). Таблиця обчислює дані про електроприлади, потребуючі наступні характеристики приладів:

- Назва прилад
- Кількість приладів одного типу
- Потужність приладу
- Час роботи приладу

На підставі отриманих даних ґрунтуються значення дефіциту та профіциту енергії.

Рисунок 2.7 — Меню вибору приладів у “Real Solar”

Дана програма поєднує в собі риси двох попередньо описаних додатків. Вона розраховує як дані про сонячний потенціал місцевості, так і значення дефіцити енергії. Також програма виводить тільки оптимальну кількість інформації (на відміну від додатку “My Solar Panel Lite”). Також застосунок є доступним, бо для його використання потрібне лише браузер та доступ в інтернет.

Але це також може бути і недоліком, якщо інформація потрібно перевірити на смартфоні або при відсутності інтернет-зв’язку. Також у додатку відсутні засоби підрахунку характеристик супутнього обладнання для сонячних панелей.

2.2 Висновок до розділу

У цьому розділі наведено три програми, що призначені для роботи зі сонячними панелями: програма для підрахунку спожитої енергії “EasySolarCalculator”, програма для аналізу сонячного потенціалу місцевості “My Solar Panel Lite” та гібридний веб-додаток, що об’єднує усі вказані функції. Було проаналізовано їх недоліки та переваги.

3. ЗАСОБИ РОЗРОБКИ

3.1 Середовище розробки

Android Studio — офіційне інтегроване середовище розробки (IDE) для Android, яка, як і сама платформа Android, була створена компанією Google. Android Studio надає найшвидші інструменти для створення додатків на всіх типах пристроїв Android.

Android Studio слугує редактором для таких популярних мов програмувань як Java, C ++, Kotlin, та містить компілятор, що дозволяє створювати APK файли і файлової системи для організації проекту [4]. Кожна програма, що створена для пристроїв на базі ОС Android, скомпільована і упакована в один файл, що складається з коду програми (.DEX файли), ресурсів, активи (assets), файл маніфесту AndroidManifest.xml і бібліотеки. Файли з даним розширенням зберігаються в магазинах додатків (наприклад, Google Play) і завантажуються з його допомогою в смартфон або планшетний комп'ютер для їх використання або встановлюються користувачем вручну [5].

Крім цього він містить редактор XML і розширений редактор макетів. Для тестування програмного продукту можна використовувати як сам пристрій на базі ОС Android так і вбудований эмулятор. Встановлені програми працюють швидше, ніж на фізичному пристрої, і є можливість в ручному режимі імітуйте різні конфігурації системи та її функції.

Android Studio працює на базі система збірки Gradle, що дозволяє налаштувати збірку програмного продукту, щоб створити і перевірити кілька варіантів збірки для різних пристроїв з одного проекту.

3.2 Мови програмування та розмітки

3.2.1 Мова програмування Java

Мовою програмування для дипломної роботи була обрана Java, оскільки вона відносну простоту та мультиплатформність. Оскільки код обробляється за допомогою віртуальної машини Java (Java Virtual Machine), то написаний на програма буде однаково працювати з будь-якою апаратною платформою, як персональний комп'ютер, смартфон чи додаток для розумного будинку [6].

Мова Java містить велику кількість корисних бібліотек, що означає — широкий спектр можливостей для програміста у розв’язанні його проблем. Всі бібліотеки — це класи, які відповідають за функціональність мови. Будь-який додаток на Java — набір класів, що описують різні об'єкти. Це дозволяє створювати складні програми, які прості в підтримці. Мова програмування підтримує безліч принципів програмування, що дозволяє ефективно вирішувати різні завдання.

Строга типізація мови програмування Java сприяє виявлення проблем у програмі. Будь-яка змінна або вираз має певний тип вже на момент компіляції, що, у разі помилки, підказує програмістові, де той її припускається, і не дає її здійснити.

3.2.2 Засоби розробки Google Maps SDK

Найважливішими даними для системи, на основі яких працює увесь подальший проект є дані про географічні координати користувача. Для їх отримання було вирішено використати засоби розробки інструменти, що надає набір розробника Google Maps SDK. Цей набір дає доступ до основних даних сервісу Google Картах. SDK підтримує мови Kotlin і Java, а також бібліотеки і розширення з додатковими функціями і технологіями [7].

3.2.3 Бібліотека GraphView

За замовчуванням, Android Studio не містить стандартних інструментів для побудови та зображення діаграм. Аби вирішити це питання, було обрано бібліотеку GraphView, що дає можливість будувати такі види діаграм як: лінійні, секторні, гістограми та точкові графіки. Також діаграми можуть оновлюватися в реальному часі, реагувати на взаємодію з ними користувача, відобразити легенду діаграми.

3.2.4 Мова розмітки XML

XML (eXtensible Markup Language, або розширювана мова розмітки) - це мова для описів документів, схожа на мову розмітки HTML, але більш універсальна. Переваги над іншими мовами розмітки (як JSON або HTML) наступні:

- Використання метаданих — одна з найбільших переваг XML, що ми можемо поміщати метадані в теги в формі атрибутів.
- Візуалізація браузера — більшість браузерів відображають XML в зрозумілій і організованій формі. Деревоподібна структура XML в браузері дозволяє користувачам природним чином згортати окремі елементи. Ця функція буде особливо корисна при налаштуваннях .
- Незалежність від платформи.

3.3 Система управлінням базою даних

SQLite — це вбудована бібліотека, яка реалізує автономний, нульової конфігурації, трансакційний механізм системи керування базами даних (СКБД) SQL, що не потребує сервера [8]. SQLite, на відміну від інших баз даних, не є автономним процесом, що означає — він може бути під'єднаний статично або динамічно, відповідно до ваших потреб. SQLite безпосередньо звертається до своїх файлів, зберігає файли на одному кроссплатформному диску. Широко використовується у мобільних додатках, чим обумовлюється його вибір для проекту.

3.4 Вимоги до системи

Система спроможна запуститися на системах Android версія 4.2 або вище.

3.5 Висновок до розділу

У цьому розділі було розглянуто середовища та засоби розробки системи для моделювання динамічних режимів роботи сонячних панелей. Оскільки програма розробляється для пристроїв на базі ОС Android, то були обрані доцільні мови програмування та середовища для розробки. Також у розділі описані бібліотеки та засоби розробки, аргументовано необхідність їх вибору.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Система моделювання режимів роботи сонячної електростанції має наступний алгоритм (рисунок 4.1) роботи:

1. Обробка даних о геолокації користувача.
2. Обчислення положення сонця та кут нахилу відносно площини місцевості протягом року
3. Обчислення кількості сонячних годин та величину сонячної інсоляція в даній місцевості.
4. Відображення отриманої інформації у вигляді графіку.
5. Створення бази даних приладів та обладнання, а також їх характеристик.
6. На основі даних з БД вивести значення споживання енергії системою.
7. Розрахувати характеристики обладнання для функціонування системи.

Рисунок 4.1 — Алгоритм роботи програми

4.1 Опис теоретичних відомостей

Для розуміння необхідності саме таких кроків та аргументації їх необхідності, розглянемо кожен крок наведеного вище алгоритму.

4.1.1 Розрахунок профіциту енергії

Першим кроком є отримання початкових даних. Таким даними у нашій системі є географічні координати (широта та довгота). Широта вказує на кут між вертикальною лінією проведеної з точки та екватором. Довгота — між площиною меридіани, що проходить через точку, та нульовим меридіаном (Гринвіч). Ці дані слугують для подальших обчислень (рисунок 4.2). Оскільки шлях сонця проходить перпендикулярно лінії екватора (у період весіннього та осіннього сонцестоянь), то основною характеристикою місцевості буде саме широта.

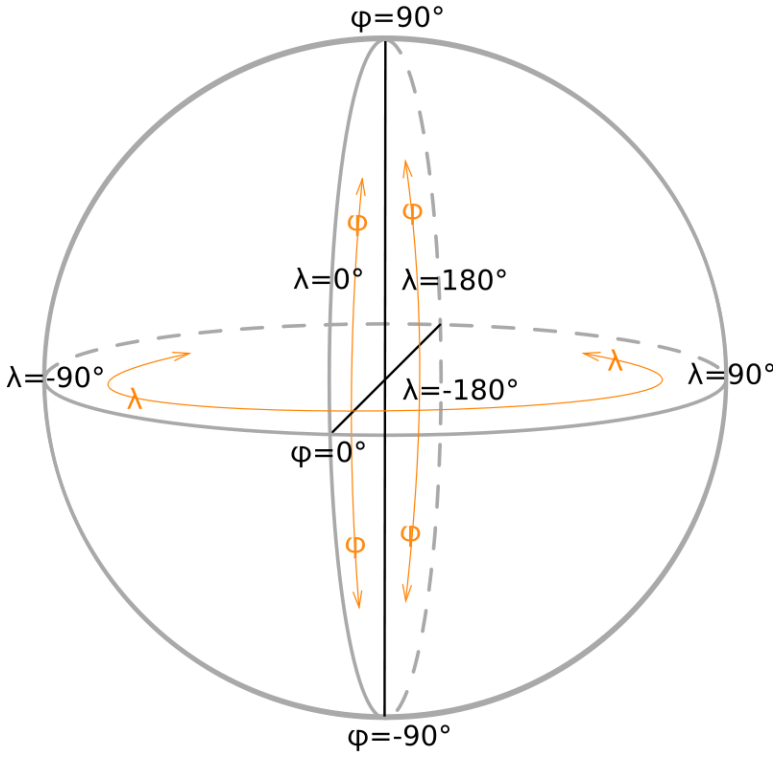

Рисунок 4.2 — Координати на географічній сфері

Рівень сонячного випромінювання не є постійним і коливається в залежності від часу та широти (рисунок 4.3). У екваторіальних широтах, між північним та південним тропіками (23° 26′ 15″) випромінювання є постійним і незначно коливається протягом року. В помірних широтах сонячне випромінювання ефективне в період літа (з квітня по вересень, для північної півкулі). А за 66°33' знаходиться

зона, де можливо спостерігати полярну ніч. Отже, знання широти є однією з найважливіших речей для рішення даної задачі.

Насамперед, визначається кутову висоту сонця до земної поверхні у певний момент часу (день). Це значення допоможе вирахувати оптимальний кут нахилу панелі [9].

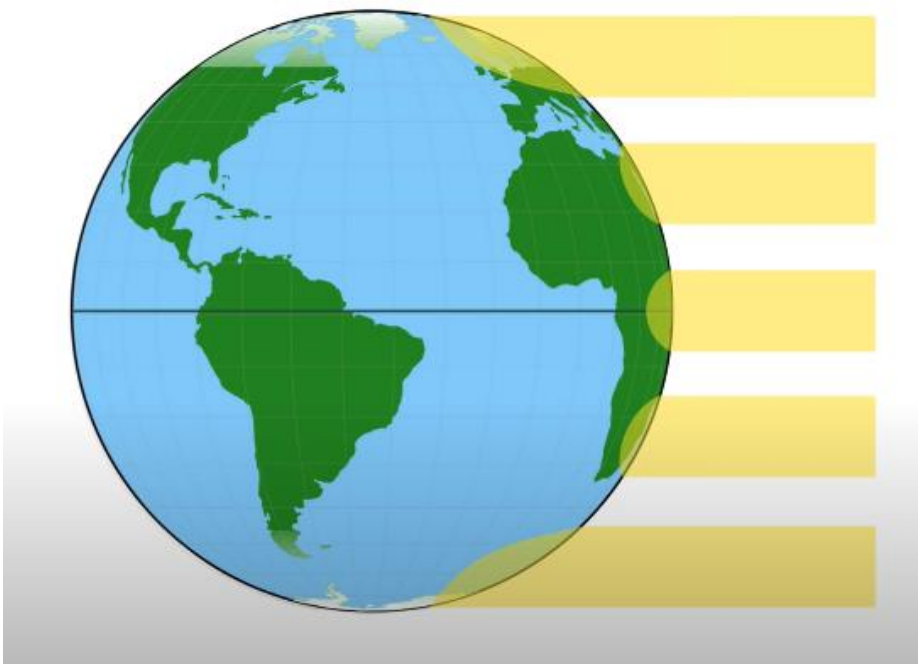

Рисунок 4.3 — Кут падіння сонячних променів (під час рівнодення)

Кутова висота сонця над горизонтом у даній місцевості розраховується за формулою:

$$\alpha = -(\frac{23.45\pi}{180}) * \cos(\frac{2\pi}{365.25} * (D + 10)), \quad (4.1)$$

де $\frac{-23.45\pi}{180}$ — широта південного тропіку у радіанній формі;

$\frac{2\pi}{365.25}$ — градусна міра, за яку змінюється висота сонця за день;

D — номер дня у році.

Оскільки для формули було обрано широту південного тропіку, то за початкове значення дня обирається день зимового сонцестояння (21 грудня) — 10й з кінця день року. Отже для поточного значення року потрібно додати 10 днів, щоб розпочати відлік з цього дня.

Кутова висота сонця над горизонтом відповідає значенню оптимального кута нахилу сонячної панелі. Розміщення сонячної панелі під даним кутом призводить до того, що кут падіння сонячних променів на панель становить 90°. Перпендикулярне падіння променів призводить до максимальної абсорбції енергії.

Також важливо, щоб панелі були повернені до Сонця в правильному напрямі. А саме – південь для північної півкулі, а північ для південної. Це також впливає на кількість отриманої енергії (рисунок 4.4).

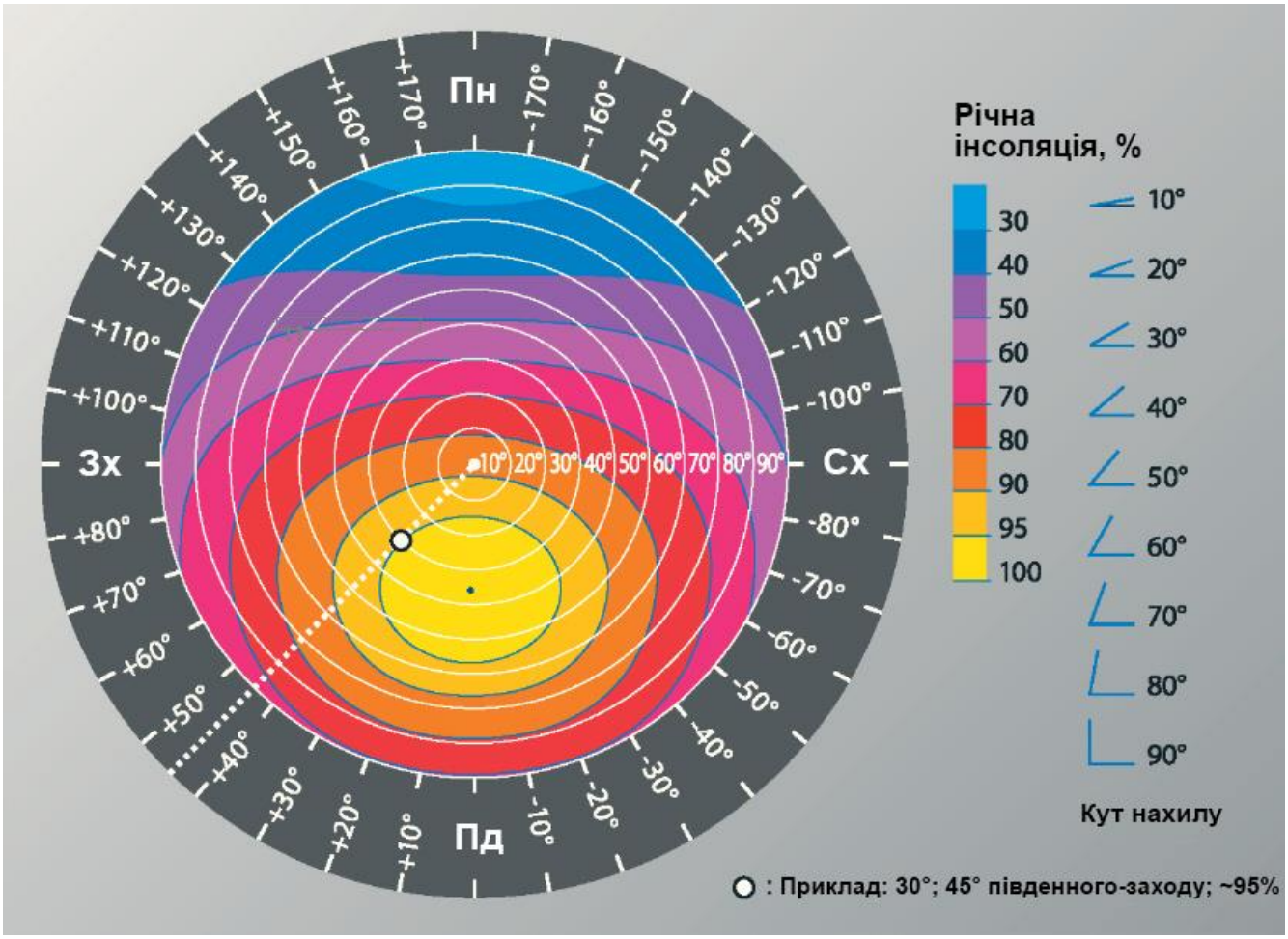

Рисунок 4.4 — Залежність отриманої енергії від кута нахилу і азимуту

Розміщення панелі перпендикулярно сонцю сприяє максимальному отриманню енергії. Відхилення призводить до втрат енергії (таблиця 4.1)

Кут падіння сонячних променів	Втрати
9	1.2%
18	4.9%
40	19.0%
45	29.0%

Таблиця 4.1 — Втрати енергії внаслідок відбиття сонячних променів

Наступним кроком є розрахунок інсоляції. Інсоляція – надходження сонячної радіації на земну поверхню. Інтенсивність її залежить від широти місцевості, хмарності та прозорості атмосфери. Інсоляція відіграє провідну роль у формуванні клімату [10]. Саме цей параметр впливає на кількість здобутою протягом дня енергії (рисунок 4.5).

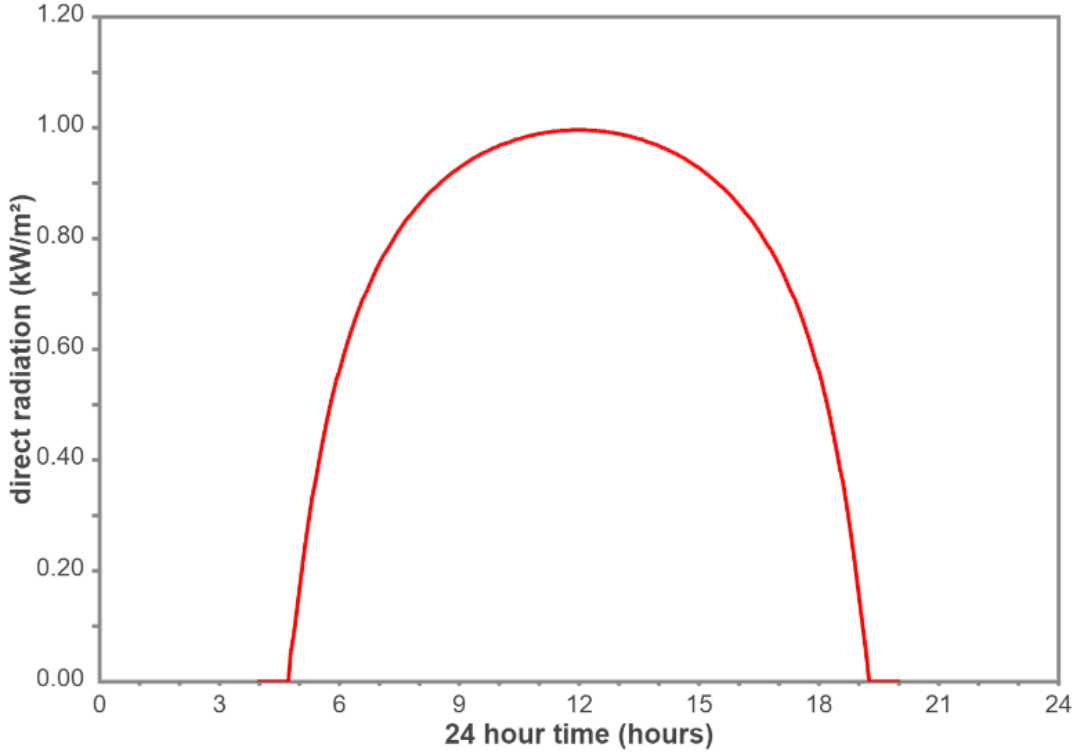

Рисунок 4.5 — Кількість сонячного випромінювання протягом дня

Значення сонячної інсоляції, до потрапляння в атмосферу (I_C) постійне протягом року, але із-за відбиття та поглинання частини сонячної енергії повітряними масами це значення зменшується. Тому для розрахунку інтенсивності сонячної інсоляції (I_d) використовується наступна формула:

$$I_d = I_C \cdot 0.7^{AM^{0.678}} \text{ (Вт/м}^2\text{)}, \quad (4.2)$$

де I_C — константа, значення сонячної інсоляції до атмосфери (1353 Вт/м^2);

AM — індекс повітряних мас;

0.7 — коефіцієнт сонячного випромінювання, що передається до Землі;

0.678 — емпіричне значення неоднорідності атмосферних шарів.

Повітряні маси поглинають та відбивають частину сонячної енергії, що призводить до її втрати, і чим більше кутова висота Сонця — тим більше мас, і відповідно менше енергії. Індекс повітряних мас розраховується за наступною формулою [11]:

$$\text{AM} = \frac{1}{\cos(\theta)}, \quad (4.3)$$

де θ — азимут місцевості (°).

Для знаходження азимуту використаємо формулу 4.4.

$$\theta = 90° + \alpha - \varphi, \quad (4.4)$$

де α — кутова висота сонця (з формули 4.1);

φ — широта місцевості.

Отже, фінальний вигляд формули [12]:

$$I_d = 1.353 \cdot 0.7^{\left(\frac{1}{\cos(90° + \alpha - \varphi)}\right)^{0.678}} \text{ (Вт/м}^2\text{)}, \quad (4.5)$$

де α — кутова висота сонця;

φ — широта місцевості;

Для розрахунку кількості отриманої енергії потрібно пам’ятати, що не вся сонячна енергія перетворюється сонячними панелями у струм. Тому отриманий результат потрібно розраховувати з урахуванням КПД сонячної панелі. Номінальне значення цього показника становить 17%.

$$Q_+ = I_d \cdot \eta \cdot S \cdot \frac{t}{24} \text{ (Вт·год)}, \quad (4.6)$$

де I_d — інтенсивність сонячного випромінювання (Вт/м^2)

η — ККД сонячної панелі (%);

S — площа сонячних панелей (м^2);

t — кількість сонячних годин (год).

4.1.2 Розрахунок дефіциту енергії

Для розрахунку дефіциту енергії потрібно визначити скільки енергії потребують прилади. Кожний електронний прилад має потужність, що вимірюється в Ватах (W). Потужність — величина, що вказує на швидкість передачі або перетворення електричної енергії енергетичні характеристики приладів [13]. Для розрахунку споживання енергії приладом в день використаємо наступну формулу:

$$Q_n = P \cdot \frac{t}{24} \left(\frac{\text{Вт}}{\text{день}}\right) \quad (4.7)$$

де P — потужність (Вт);

t — кількість годин у день, протягом яких прилад працює (год).

На підставі формули 4.7 розробимо формулу 4.8, що буде розраховувати потреби в енергії усієї системи.

$$Q_{-} = \sum_{i=0}^{n} \left(k_i \cdot P_i \cdot \frac{t_i}{24} \right) \left(\frac{\text{Вт}}{\text{день}} \right), \tag{4.8}$$

де n — загальна кількість видів приладів;

i — номер поточного приладу;

k – кількість приладів одного типу,

P – потужність приладу (Вт),

t – час роботи приладу в день (год)

Подключаемое устройство	Мощность, Вт	Количество устройств	Время работы (часов в сутки)	Потребляемая энергия (в день)
Холодильник:	250	1 шт	0 ч 0 м	0 кВт·ч
Стиральная машина:	1500	1 шт	0 ч 0 м	0 кВт·ч
Микроволновка:	1000	1 шт	0 ч 0 м	0 кВт·ч
Электрочайник:	2000	1 шт	0 ч 0 м	0 кВт·ч
Кофеварка:	750	1 шт	0 ч 0 м	0 кВт·ч
Электроплита:	2000	1 шт	0 ч 0 м	0 кВт·ч
Электрообогреватель:	1000	1 шт	0 ч 0 м	0 кВт·ч

Рисунок 4.6 — Приклад калькулятора для обчислення потреб в електроенергії

Отримана в результаті розрахунків сума — необхідна кількість енергії в день для роботи системи.

4.1.3 Визначення характеристик обладнання

Системи генерування сонячної енергії (рисунок 4.7) складається з таких основними компонентів є:

- Сонячні панелі
- Контролер заряду АКБ
- Акумуляторні батареї (АКБ)
- Інвертор

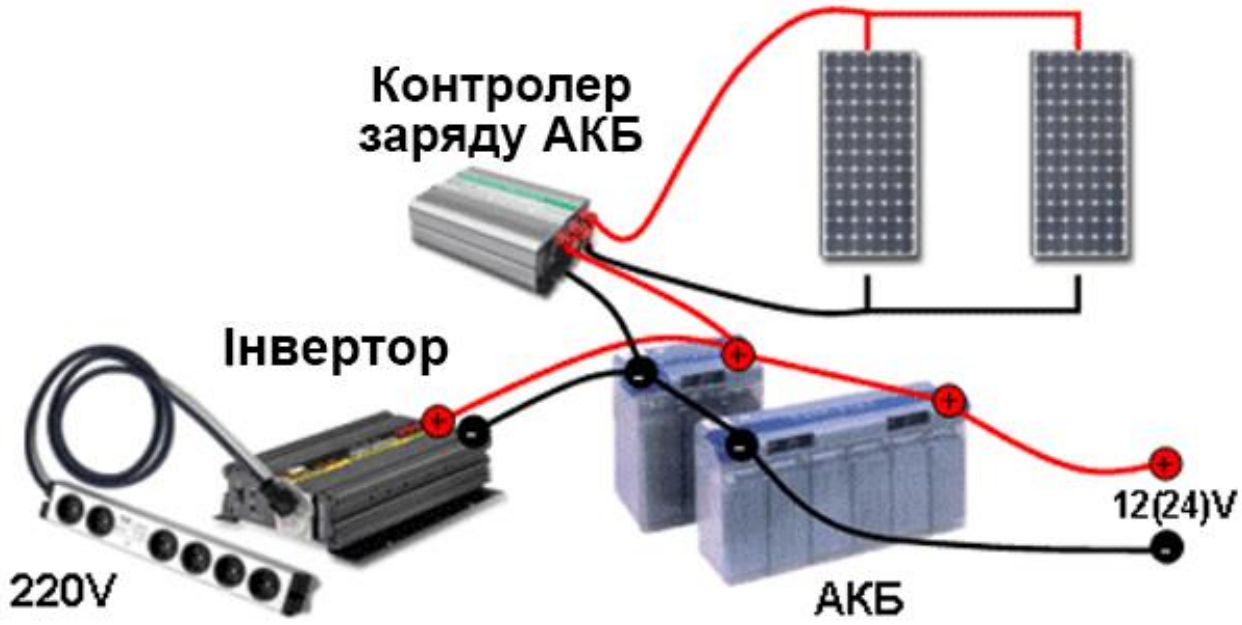

Рисунок 4.7 — Схема сонячної мікроенергостанції

Сонячні панелі — головна частина системи генерування сонячної мікроелектростанції. Саме вони абсорбують сонячну енергію і перетворює її у електричну.

Розрахунок площі сонячних панелей був розглянутий у попередніх розділах і становить:

$$S > \frac{Q_-}{I_d \cdot \eta \cdot \frac{t}{24}} + 20\% \text{ (м}^2\text{)}, \tag{4.9}$$

де S — мінімальна площа панелей (м^2);

Q_- — потреба системи в енергії (Вт/день);

I_d — значення інсоляції (Вт/м^2);

t — кількість сонячних годин в день (год);

η — ККД панелі.

Оскільки струм, що проходить надходить від сонячних панелей не є постійним, то система потребує контролер, що буде забезпечувати нормування вихідної напруги, зарядку акумуляторів та подачу низьковольтного постійного струму. Значення напруги, що видає контролер є номінальними і становлять 12В, 24В або 48В в залежності від приладу. Його розрахунку система не потребує.

Акумуляторні батареї призначені для зберігання енергії та її використання у час, коли енергія не генерується, або дефіцит системи перевищує профіцит. Як і контролер струму, АКБ мають власну напругу, яка є номінальною (6В, 12В, 24В), при цьому значення напруги батареї має бути менше або дорівнювати напрузі інвертора.

Кількість енергії, що необхідно акумулювати (Ампер·год) розраховується за наступною формулою і залежить від напруги у системі. Формула для обчислення становить:

$$q = \frac{Q_- \cdot t}{U_{\text{кнтр}} \cdot R} \ \text{(Ампер} \cdot \text{год)}, \qquad (4.10)$$

де Q_- — потреба системи в енергії (Вт/день);

t — час автономної роботи АКБ (день);

$U_{\text{кнтр}}$— напруга від контролера (В);

R — рівень розряду акумулятора (%).

R — це показник, що вказує наскільки батарея має розряджатися. Його важливо врахувати, бо часта розрядка АКБ до нуля призводить до її швидкої деградації. Оптимальним вважається значення розряду 65-70%.

Для розрахунку кількості АКБ, потрібно знати їх номінальну ємність (Ампер · год).

$$n_{\text{АКБ}} = \left\lfloor \frac{q}{q_n} \right\rfloor \cdot \frac{U_{\text{кнтр}}}{U_{\text{акб}}}, \tag{4.11}$$

де q — загальна кількість енергії для акумулювання (Ампер · год);

q_n — номінальну ємність (Ампер · год);

$U_{\text{кнтр}}$— напруга від контролера (В);

$U_{\text{акб}}$ — напруга акумулятора (В).

Інвертор — прилад, що перетворює постійний низьковольтовий струм від акумуляторів у стандартний для господарства або підприємства (220В/380В).

Його характеристика дорівнює значенню потужності усіх приборів увімкнутих одночасно + деякий відсоток (20-25%). Він потрібен, щоб у майбутньому можливо було приєднати нові прилади до системи. Також, деякі електропристрої потребують значно більше напруги при увімкненні, ніж при звичайній роботі.

4.2 Структура програмного проекту

Для реалізації програмного продукту було створено додаток у середовищі Android Studio. Система програмного продукту складається з Android Manifest, файлів класів мовою Java, в яких описані функції програми, та файлів розмітки сторінок XML, що описують розташування елементів у вікнах (рисунки 4.8 та 4.9).

Кожна Android програма містить у свої основі файл AndroidManifest.xml. В ньому описані основні відомості про систему:

- ім'я Java-пакета застосунку — його унікальний ідентифікатор;
- опис компонентів програми - активності, служби, контент-провайдери, що дозволяє викликати класи, які реалізують кожен з компонентів, і оголошує їх наміри;

- містить список необхідних дозволів для звернення до захищених частин API і взаємодії з іншими додатками;
- оголошує дозволу, які сторонні додатки повинні мати для взаємодії з компонентом цього додатка;
- оголошує мінімальний рівень API Android, необхідний для роботи програми;
- перелік бібліотек;

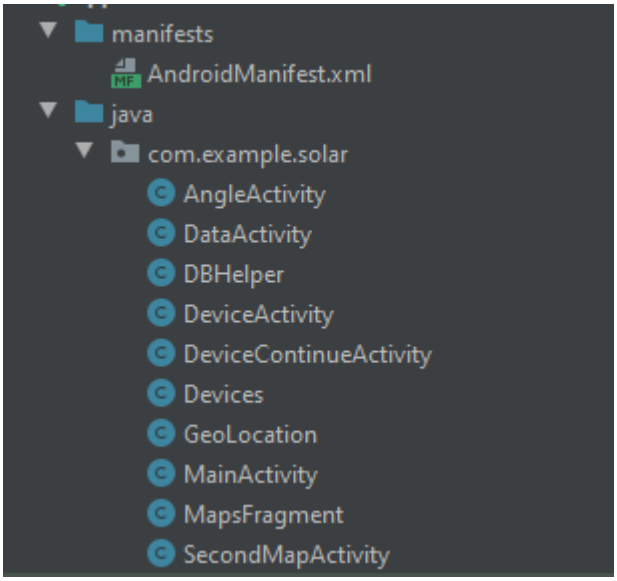

Рисунок 4.8 — Android Manifest та класи Java

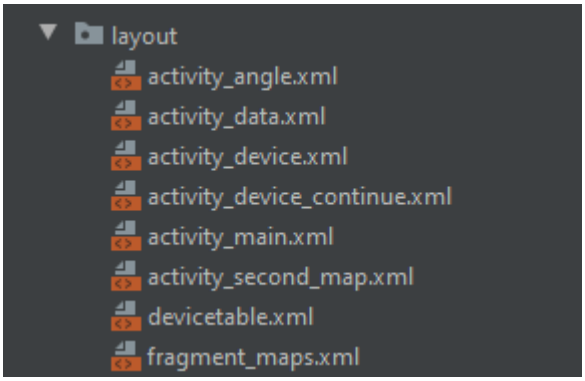

Рисунок 4.9 — Layout файли

Також у папці java зберігається перелік використаних .java, в яких описан функції, що виконує програма, а також допоміжні класи.

Опис класів програмного продукту:

- AngleActivity.java — клас, в якому описаний функціонал вікна “Кут нахилу” та функції для розрахунку getActualDeviceOrientation(), для розрахунку положення пристрою в просторі та функція розрахунку оптимального кута для сонячної панелі;
- DataActivity.java — клас, в якому описано функцію графіку та проводяться розрахунки значення інсоляції, використовує допоміжний клас GeoLocation.
- DBHelper.java — допоміжний клас для роботи з SQLite;
- DeviceActivity.java — клас, що описує функції вікна з приладами, містить розрахунки обладнання для сонячної електростанції, використовує клас Devices.java та DBHelper.java;
- DeviceContinueActivity.java — клас аналогічний до DeviceActivity.java, продовжує розрахунки обладнання.
- Devices.java— клас, що описує поля та методи електропристроїв;
- GeoLocation.java — клас, що описує поля та методи географічних координат;
- MainActivity.java — клас головного меню в якому описані функції переходу у інші вікна;
- MapsFragment.java — клас для роботи зі сервісами Google Maps API;
- SecondMapActivity.java — клас, у якому описано функціонал вікна з картою, використовує додаткові класи MapsFragment.java та GeoLocation.java.

Також програма містить файли розмітки мовою XML, що визначають порядок розміщення елементів у вікні програми (рисунок 4.10 та 4.11).

Опис вікон розмітки:

- activity_angle.xml — вікно “Кут нахилу”, складається з динамічних зображень компасу та сонячних панелей, що управляються класом AngleActivity.java;
- activity_data.xml — вікно “Інсоляція”, складається з графіку функції інсоляції та керується класом DataActivity.java;

- activity_device.xml — вікно “Пристрої №1”, складається з таблиці наведеної у devicetable.xml та меню для обчислення характеристик обладнання. Керується класом DeviceActivity.java ;
- activity _device_continue.xml — вікно “Пристрої №2” також слугує для обчислення значень обладнання і керується класом DeviceContinue-Activity.java;
- activity_main.xml — вікно “Головне меню”, що містить кнопки переходу у інші розділи;
- activity_second_map.xml — вікно “Карта”, що містить у собі віджет Google Maps і використовує функції класу SecondMapActivity.java;

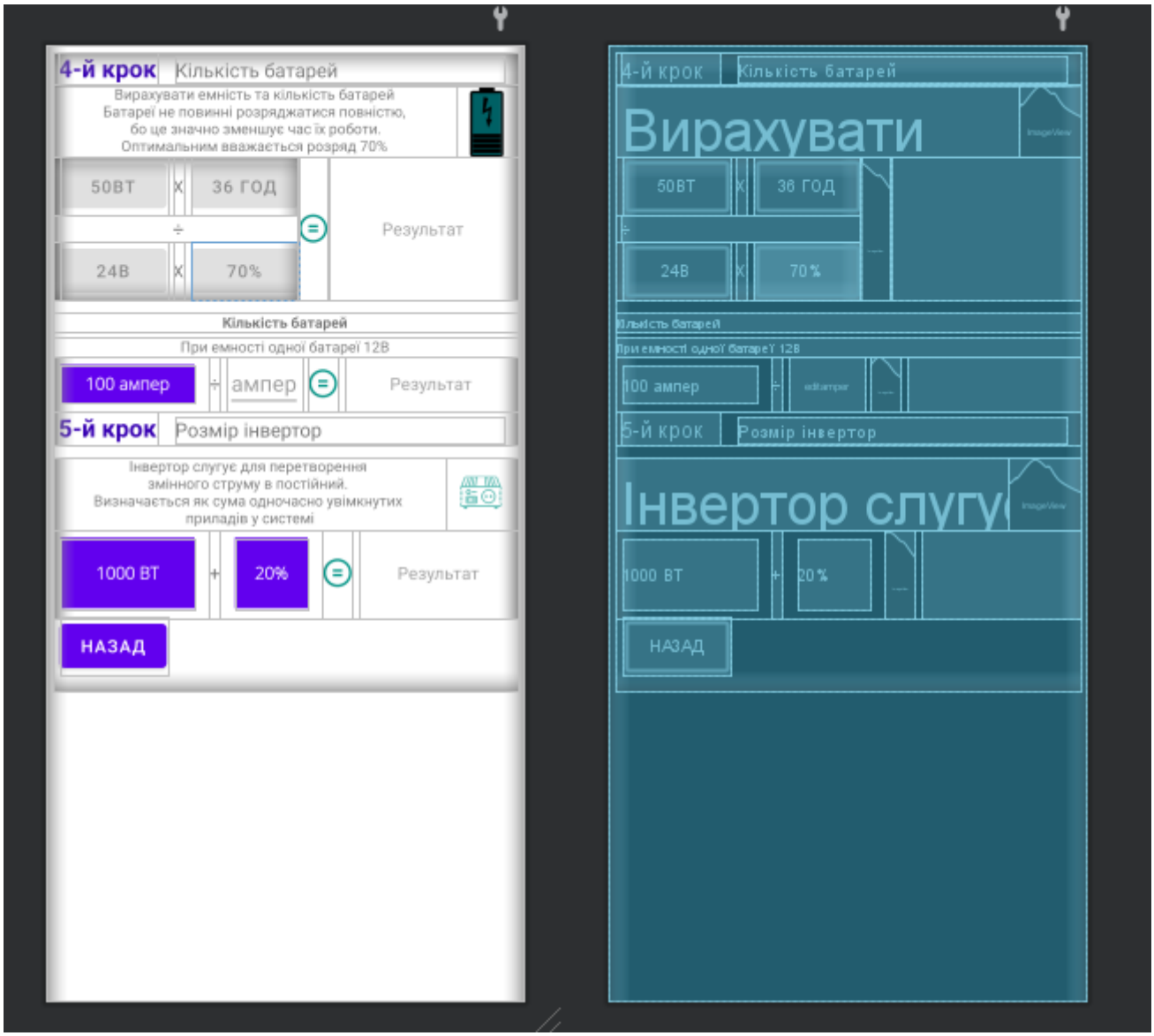

Рисунок 4.10 — Графічне та схематичне зображення вікна

- devicetable.xml — містить у собі шаблон таблиці споживання енергії домашніми електроприладами для використання у вікні “Пристрої №1” ;
- fragment_map.xml — шаблон для використання карти у вікні “Карта”.

Рисунок 4.11 — Структура коду XML на прикладі вікна DeviceContinueActivity

4.3 Взаємодія користувача з системою

Для демонстрації варіантів взаємодії користувача з розробленою системою було створено діаграму прецедентів (рисунок 4.12).

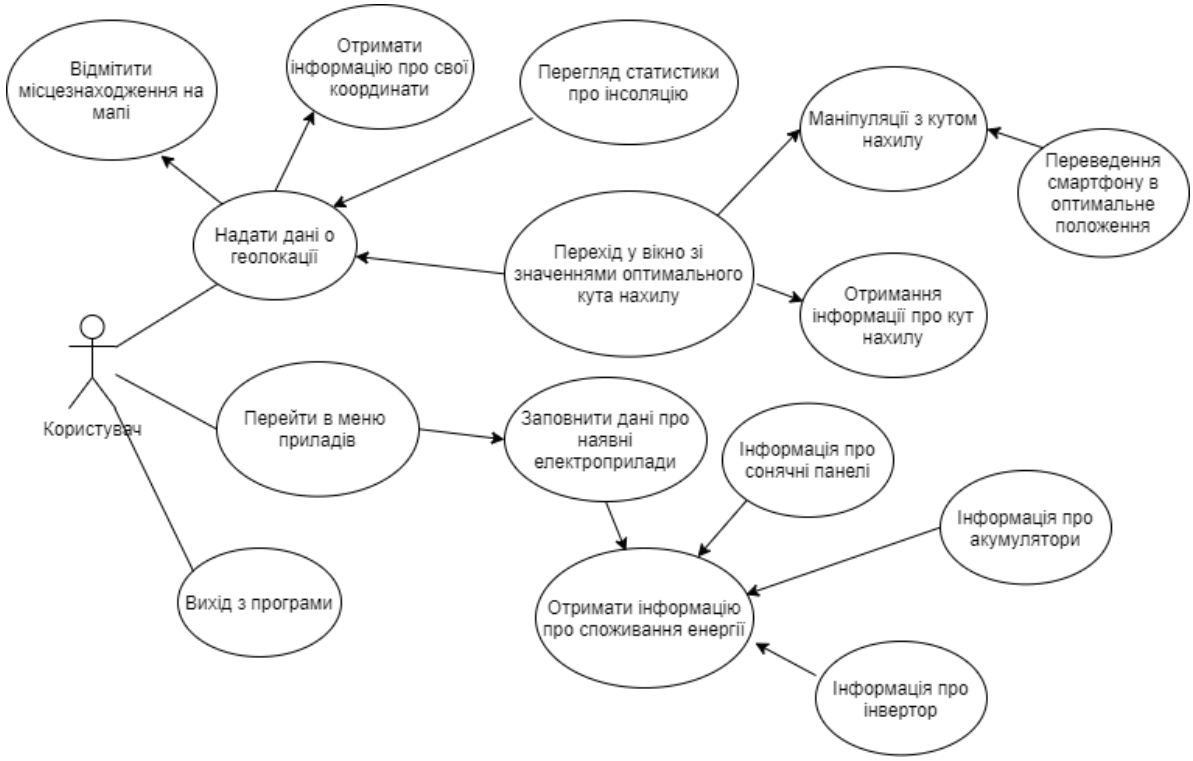

Рисунок 4.12 — Діаграма прецедентів

На діаграмі прецедентів відображено шляхи взаємодію користувача (актора) зі системою.

4.4 База даних системи

База даних системи складається з трьох взаємопов’язаних таблиць. Вони створені для зберігання та отримання доступу до даних про енергетичні характеристики приладів користувача.

Таблиця “List” є головною і зберігає в собі значення енергетичних характеристик обладнання конкретного користувача: час роботи приладу, його потужність та кількість.

Таблиця “Device” містить інформацію про номінальну потужність деяких приладів.

Таблиця “User” визначає список користувачів програмою.

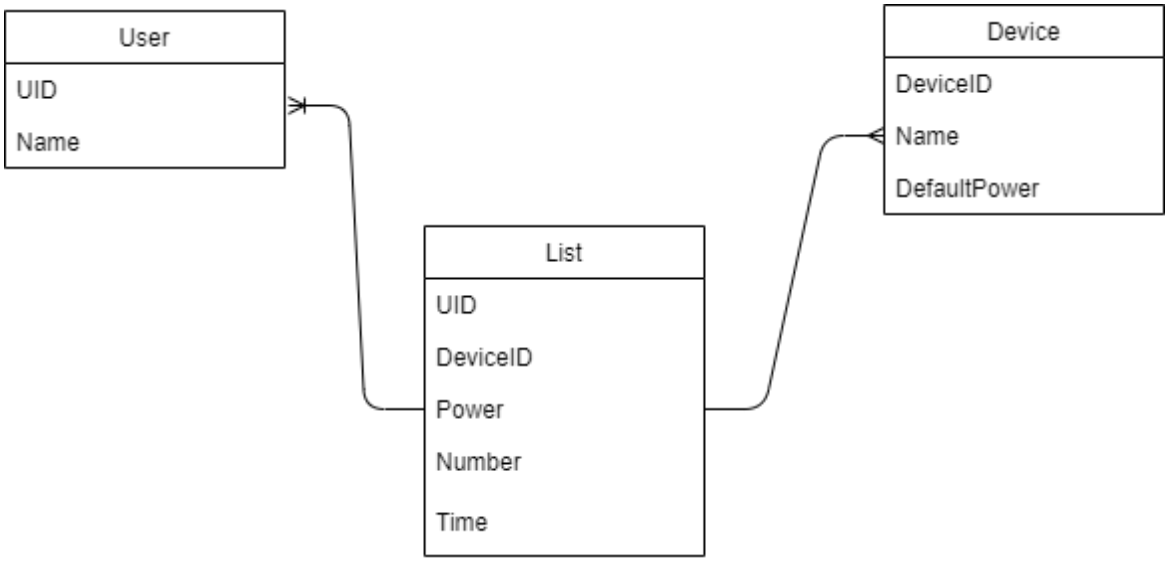

Рисунок 4.13 — Модель БД системи

Розглянемо детально структуру кожної з таблиць бази даних. У таблицях 4.2 — 4.4 продемонстровано ім'я поля, його тип та короткий опис.

Назва поле	**Тип поля**	**Опис**
ID	INTEGER	Первинний ключ
UID	INTEGER	Зовнішній ключ на таблицю User
DeviceID	INTEGER	Зовнішній ключ на таблицю Device
Power	REAL	Значення потужності, за замовчуванням дорівнює 0 або значенню відповідного DefaultPower з таблиці Device
Number	INTEGER	Кількість приладів
Time	REAL	Час роботи пристроїв (у годинах)

Таблиця 4.2 — таблиця “List”

Назва поле	Тип поля	Опис
UID	INTEGER	Первинний ключ
Name	NVARCHAR	Ім’я користувача

Таблиця 4.3 — таблиця “User”

Назва поле	Тип поля	Опис
DeviceID	INTEGER	Первинний ключ, номер пристрою
Name	NVARCHAR	Назва пристрою
DefaultPower	REAL	Номінальне значення потужності

Таблиця 4.4 — таблиця “Device”

4.5 Висновок до розділу

У цьому розділі розглянемо теоретичне обґрунтування системи та опис її програмної реалізації. Продемонстровано файлову структура та структура класів програмного продукту. Було надано приклад реалізації бази даних та відображено шляхи взаємодії користувача зі системою.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ

5.1 Встановлення та системній вимоги

Для встановлення програмної системи мобільний пристрій повинен мати наступні характеристики:

- Операційна система: Android 4.2 або вище.
- Оперативна пам’ять (ОЗП): мінімум 512МБ, рекомендовано 2 ГБ
- Пам’ять: 10 МБ або більше.

Програма встановлюється шляхом завантаження APK (Android Package) файлу на пристрої, що відповідає попереднім вимогам.

5.2 Інструкція по використанню програмного продукту

При завантаженні, користувач потрапляє до головного меню (рисунок 5.1). Перед ним знаходяться три кнопки (“Карта”, “Інсоляція”, “Прилади”) для переходу у відповідні меню.

Рисунок 5.1 – Кнопки головного меню

Для оптимального отримання інформації користувач має пройтись пунктами програми у наступному порядку: “Карта” → “Інсоляція” → “Прилади”. Але такий крок є опціональним.

При натисканні на кнопку “Карта” користувач перейде до вікна з мапою (рисунок 5.2).

Рисунок 5.2 – Вікно з мапою

Вікно містить наступні елементи: мапа з маркером, текстові поля зі значенням широти та довготи, дві кнопки для переходу на минуле та наступне вікна.

Дана мапа побудована на базі Google Map і підтримує функції наближення та віддалення, а також переміщення об'єктів.

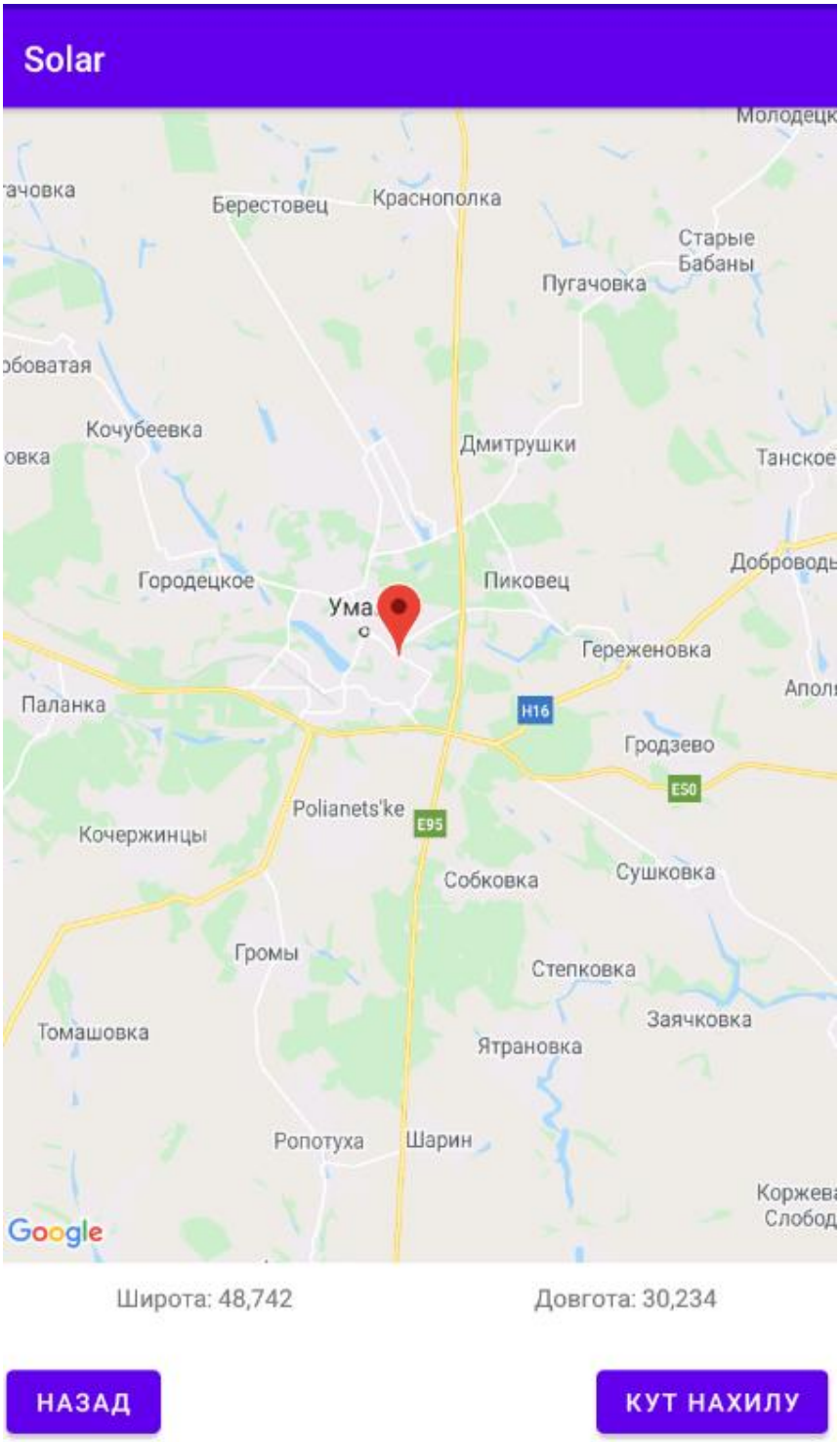

Рисунок 5.3 – Демонстрація переміщення маркеру.

Вона містить маркер, який можна переміщати для вибору потрібної позиції. Координати точки відображаються на екрані.

Після вибору точки та отримання інформації що-до її географічного розміщення, користувач переходить у наступне вікно, натиснувши на кнопку “Кут нахилу”. Це відкриває вікно для визначення кута нахилу панелі (рисунок 5.4).

Рисунок 5.4 – Вікно для становлення кута нахилу панелі

Воно містить компас, що вказує на північ, а також інформацію, в яку сторону має бути повернута сонячна панель. Нижче знаходиться інтерактивне зображення

сонячної панелі, яка складається з двох частин. Рухомої “панелі” та нерухомої “рамки”. Нахиляючи телефон відносно осі Y (ширина телефону), користувач змінює кут його нахилу. Тип самим він може як дізнатися кут нахилу вже існуючої площини, так побачити наочно, яким є оптимальний кут нахилу панелей.

Обравши в головному меню пункт “Інсоляція”, користувач переходить до наступного вікна (рисунок 5.5).

Рисунок 5.5 – Вікно інсоляції

У вікні інсоляція наведений графік сонячної активності (W/м^2/день) на кожний місяць року для обраної раніше місцевості. Також відображається інформація щодо значень екстремумів графіку (мінімальне та максимальне значення що набуває функція), а також середнє значення. Ця інформація відображає ефективність сонячних панелей в певні місяці для аналізу їх необхідності в електросистемі будинку.

Обравши в головному меню пункт “Прилади”, користувач переходить до меню розрахунку, що складається з двох вікон (рисунок 5.6 та 5.7), необхідних характеристик обладнання для надійного функціонування системи. Сторінка поділення на кроки для кращого розуміння.

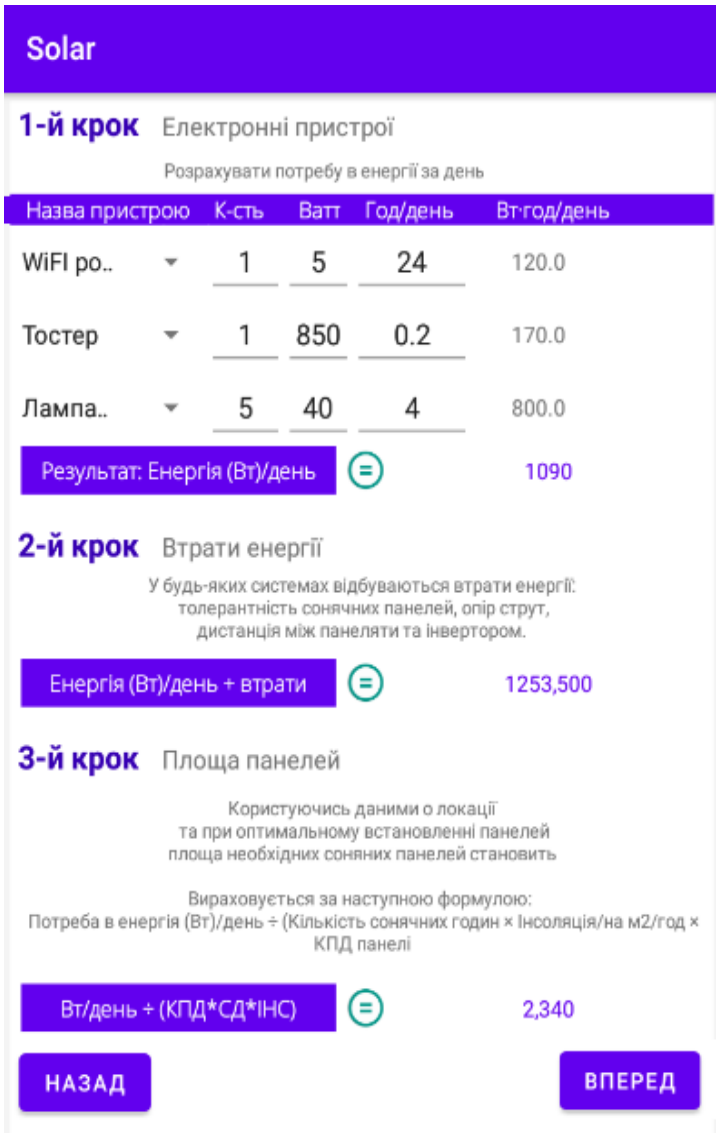

Рисунок 5.6 – Перше вікно панелі приладів

Рисунок 5.7 — Друге вікно панелі приладів

“1-й крок. Електронні пристрої” — пункт потребує від користувача навести список електронних приладів, що будуть використовуватися у системі. У ній можна обрати один з існуючих приладів, або занести дані свого (для існуючих можливо змінити характеристики). Також потрібно обрати кількість та час роботи приладів. В результаті буде дана загальна кількість денно спожитої енергії, як самим приладом, так і системою в цілому.

“2-й крок. Втрати енергії” — спираючись на дані про необхідну кількість енергії, система враховую втрати, що обов’язково будуть у системі і збільшує величину необхідної енергії.

“3-й крок. Площа панелей” — за допомогою даних про інсоляцію, розраховується необхідна площа сонячних панелей у $м^2$.

Після цього, користувач має натиснути кнопку “Вперед” для переходу на наступне вікно (рисунок 5.7). В ньому продовжується розрахунок обладнання.

“4-й крок. Кількість батарей” — розраховує необхідну величину енергії для запасання та кількість акумуляторів для системи. Величина потрібної енергії розрахована в попередніх пунктах вже занесена до таблиці. Користувач має обрати час, протягом якого система має працювати автономно, напругу, що видає інвертор та величину розряду акумулятора. За замовчуванням ця величина становить 70%, бо вона вважається оптимальною. При її збільшенні, значно зменшується термін служби батарей. Отримана величина — показник характеристика ємності акумулятора у Ампер·годинах.

“5-й крок. Розмір інвертора” — оскільки напруга в інверторі не є постійною, то потрібен прилад для її перетворення змінний струм у постійний. У цьому кроці розраховується потужність інвертора, який потребує система користувача.

5.3 Висновок до розділу

У цьому розділі було розглянуто системні характеристики обладнання для роботи програми, а також наведена покрокова демонстрація роботи програми для користувача. Описана роботи таких елементів програмного продукту як карта, вікно розрахунку кута нахилу, вікно інформації про інсоляцію та вікно розрахунку значень сонячної енергії.

ВИСНОВКИ

У ході виконання дипломної роботи було створено програмний додаток для розрахунку обсягів отриманої сонячної енергії та визначення необхідного обладнання для функціонування системи.

Система обчислює дані щодо отриманої сонячної енергії на базі географічного розміщення та видає потрібну для користувача інформацію.

В процесі дослідження було проаналізовано схожі системи, що існують на сьогодні. Був проведений їх аналіз в порівнянні з розробленою, та виявлені наступні переваги:

- Зрозумілість та простота у використанні;
- Доступність для більшості користувачів;
- Надання оптимальної кількісті потрібної інформації;

Було вирішено розробити систему на базі мобільного додатку. Мобільний додаток задовольняв потребу у зручності використання і доступності для кінцевого користувача. Розроблений продукт низку можливостей:

- Демонстрацію геолокації;
- Наведення індивідуальних рекомендацій, щодо розміщення сонячних панелей
- Аналіз енергетичного потенціалу місцевості;
- Розрахунок дефіциту енергії
- Вивід усієї обчисленої інформації користувачу.

СПИСОК ЛІТЕРАТУРИ

1. Закон України “Про альтернативні джерела енергії” [Електронний ресурс] / Відомості Верховної Ради України (ВВР), 2003, № 24, ст.155 – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/555-15#Text.

2. Плачкова С. Г. Енергетика: історія, сучасність і майбутнє.– Кн. 5 Електроенергетика та охорона навколишнього середовища. Функціонування енергетики в сучасному світі [Електронний ресурс] / Светлана Григорьевна Плачкова – Режим доступу до ресурсу: http://energetika.in.ua/ru/books/book-5.

3. Сонячні електростанції у приватних домогосподарствах (СЕСд): динаміка розвитку [Електронний ресурс] – Режим доступу до ресурсу: https://saee.gov.ua/uk/content/sesd.

4. Документація Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/docs.

5. APK File Extension [Електронний ресурс] – Режим доступу до ресурсу: https://www.file-extension.info/format/apk.

6. Документація мови програмування Java [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/.

7. Google Maps Platform Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation.

8. Документація SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://sqlite.org/docs.html.

9. General Solar Position Calculation [Електронний ресурс] // NOAA Global Monitoring Division – Режим доступу до ресурсу: https://gml.noaa.gov/grad/solcalc/solareqns.PDF.

10. Термінологічний словник-довідник з будівництва та архітектури / Р. А. Шмиг, В. М. Боярчук, І. М. Добрянський, , В. М. Барабаш. – Львів, 2010. – 222 с.

11. Fritz K. Revised optical air mass tables and approximation formula / K. Fritz, Y. Andrew // Applied Optics / K. Fritz, Y. Andrew., 1989. – С. 4735–4738.

12. Calculation of Solar Insolation [Електронний ресурс] – Режим доступу до ресурсу: https://www.pveducation.org/pvcdrom/properties-of-sunlight/calculation-of-solar-insolation.

13. Матвієнко М. П. Основи Електротехніки. Підручник / М. П. Матвієнко. – Київ: Ліра-К, 2018. – 228 с. – (2-ге перероблене і доповнене).

ДОДАТОК А

Програмне забезпечення для аналізу складу певного контингенту

Специфікація

УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ-72207_21Б

Аркушів 1

Київ — 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕП С_ТМ-72207_24Б	Пояснювальна записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕП С_ТМ-72207_24Б 12-1	MainActivity.java	Компонент з головним меню програми
УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕП С_ТМ-72207_24Б 12-2	SecondMapActivity.java	Компонент з картою
УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕП С_ТМ-72207_24Б 12-3	AngleActivity.java	Компонент з кутом нахилу сонячної панелі
УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕП С_ТМ-72207_24Б 12-4	DeviceActivity.java	Компонент з розрахунком обладнання №1
УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕП С_ТМ-72207_24Б 12-5	DeviceContinue Activity.java	Компонент з розрахунком обладнання №2
УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕП С_ТМ-72207_24Б 13-1	Додаток В	Опис програмного модуля

ДОДАТОК Б

Програмне забезпечення для аналізу складу певного контингенту

Лістинг програми

УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ-72207_21Б 12-1

Аркушів 14

Київ — 2021

MainActivity.java

```
package com.example.solar;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity  {

    double Lanx;
    double Lngx;
    double InsolMID;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
      //  Toast toast = Toast.makeText(getApplicationContext(),
       //         "Hello", Toast.LENGTH_SHORT);
       // toast.show();

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.buttonmap);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent1 = new Intent(MainActivity.this,SecondMapActivity.class);
                Bundle arguments = getIntent().getExtras();
                if (arguments != null) {
                     double Lan = arguments.getDouble("Lan");
                     double Lng = arguments.getDouble("Lng");
                    intent1.putExtra("Lan", Lan);
                    intent1.putExtra("Lng", Lng);
                }
                else{
                    intent1.putExtra("Lan", Lanx);
                    intent1.putExtra("Lng", Lngx);
                }
                startActivity(intent1);
                finish();
            }
        });

        Button buttondevice = (Button)findViewById(R.id.buttondevice);
        buttondevice.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent2 = new Intent(MainActivity.this,DeviceActivity.class);
                Bundle arguments = getIntent().getExtras();
                if (arguments != null) {
                    double InsolMID = arguments.getDouble("InsolMID");
                    intent2.putExtra("InsolMID", InsolMID);
                }
                else{
                    intent2.putExtra("InsolMID", InsolMID);
                }
                startActivity(intent2);
```

```
                finish();
            }
        });

        Button buttontestxxxx = (Button)findViewById(R.id.buttontestx);
        buttontestxxxx.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent3 = new Intent(MainActivity.this, DataActivity.class);
                Bundle arguments = getIntent().getExtras();
                if (arguments != null) {
                    double Lan = arguments.getDouble("Lan");
                    double Lng = arguments.getDouble("Lng");
                    intent3.putExtra("Lan", Lan);
                    intent3.putExtra("Lng", Lng);
                }
                else{
                    intent3.putExtra("Lan", Lanx);
                    intent3.putExtra("Lng", Lngx);
                }
                startActivity(intent3);
                finish();
            }
        });

    }
}
```

SecondMapActivity.java

```
package com.example.solar;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.location.Location;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import java.text.DecimalFormat;



public class SecondMapActivity extends AppCompatActivity implements OnMapReadyCallback,
        GoogleMap.OnMapClickListener {
    private GoogleMap mMap;
    double Lan = 0.4547;
    double Lng = 0.5238;

    //SharedPreferences gData;
```

```
    public static String LatLngtoString(LatLng Marker){
        String curpos = String.valueOf(Marker.latitude);
        curpos+=", ";
        curpos += String.valueOf(Marker.longitude);
        return curpos;
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second_map);

       // gData = getSharedPreferences(APP_PREFERENCES, Context.MODE_PRIVATE);

        Button button = (Button) findViewById(R.id.returnbutton);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(SecondMapActivity.this,MainActivity.class);
                intent.putExtra("Lan",Lan);
                intent.putExtra("Lng",Lng);
                startActivity(intent);
            }
        });


        Button buttonnext = (Button) findViewById(R.id.Angle);
        buttonnext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(SecondMapActivity.this,AngleActivity.class);
                intent.putExtra("Lan",Lan);
                intent.putExtra("Lng",Lng);
                startActivity(intent);
            }
        });

        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        mMap.setOnMapClickListener(this);

        Bundle arguments = getIntent().getExtras();
        LatLng KievPos = new LatLng(arguments.getDouble("Lan"), arguments.getDouble("Lng"))
;
        ChangeText(KievPos);
        mMap.addMarker(new MarkerOptions().position(KievPos).draggable(true));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(KievPos));
    }



    @Override
    public void onMapClick(LatLng latLng) {
        mMap.clear();
        /*
```

```
        Toast.makeText(SecondMapActivity.this,
                "onMapLongClick:\n" + latLng.latitude + " : " + latLng.longitude,
                Toast.LENGTH_LONG).show();

         */
        //Add marker on LongClick position

        MarkerOptions markerOptions = new
MarkerOptions().position(latLng).title(latLng.toString());
        markerOptions.draggable(true);

        mMap.addMarker(markerOptions);

        ChangeText(latLng);

    }



    public void ChangeText(LatLng latLng){
        TextView text1 = (TextView) findViewById(R.id.shir);
        TextView text2 = (TextView) findViewById(R.id.dolg);
        DecimalFormat decimalFormat = new DecimalFormat( "#.###" );
        String result1 = "Широта: " +
                decimalFormat.format(latLng.latitude);
        String result2 = "Довгота: " +
                decimalFormat.format(latLng.longitude);



        text1.setText(result1);
        text2.setText(result2);

        Lan = latLng.latitude;
        Lng = latLng.longitude;

    }

}
```

AngleActivity.java

```
package com.example.solar;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.renderscript.Sampler;
import android.view.Display;
import android.view.Surface;
import android.view.View;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
```

```
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.Timer;
import java.util.TimerTask;

public class AngleActivity extends AppCompatActivity implements SensorEventListener {

    double Lan ;
    double Lng ;

    TextView tvText;
    SensorManager sensorManager;
    Sensor sensorAccel;
    Sensor sensorMagnet;

    StringBuilder sb = new StringBuilder();

    Timer timer;

    int rotation;

    private ImageView imageView;
    private float[] mGrav = new float[3];
    private float[] mGeo = new float[3];
    private float azimuth = 0f;
    private float currentAzimuth = 0f;
    private SensorManager mSensorManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_angle);


        tvText = (TextView) findViewById(R.id.tvText);
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        sensorAccel = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorMagnet = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);


        imageView = (ImageView) findViewById(R.id.compass);
        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        Bundle arguments = getIntent().getExtras();
        Lan = arguments.getDouble("Lan");
        Lng = arguments.getDouble("Lng");

        Button button = (Button) findViewById(R.id.returnbutton);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(AngleActivity.this, MainActivity.class);
                intent.putExtra("Lan", Lan);
                intent.putExtra("Lng",Lng);
                startActivity(intent);
            }
        });

        ImageView shadow = (ImageView) findViewById(R.id.solarpanelshadow);
        Calendar cal = new GregorianCalendar();
        cal.setTime(new Date());
        int day = cal.get(Calendar.DAY_OF_YEAR);
```

```
        double optimalangle = -(23.44*Math.PI/180)* Math.cos(((2*Math.PI)/365)*(day+10));
        float angle = (float)Lan - (float)Math.toDegrees(optimalangle);
        shadow.setRotation(Math.abs(angle));
         String angletextstring = String.format("%1$.1f",Math.abs(angle));
        TextView angletext = (TextView) findViewById(R.id.tvText2);
        angletext.setText(angletextstring+"°");

        TextView NS = (TextView) findViewById(R.id.textViewNS);
         if(angle < 0){
             NS.setText("S\nN");
         }
         else {
             NS.setText("N\nS");
         }

    }

    @Override
    protected void onResume(){
        super.onResume();

mSensorManager.registerListener(this,mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FI
ELD),
                SensorManager.SENSOR_DELAY_GAME);

mSensorManager.registerListener(this,mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMET
ER),
                SensorManager.SENSOR_DELAY_GAME);
        sensorManager.registerListener(listener, sensorAccel,
SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(listener, sensorMagnet,
SensorManager.SENSOR_DELAY_NORMAL);


        timer = new Timer();
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        getDeviceOrientation();
                        getActualDeviceOrientation();
                        showInfo();

                    }
                });
            }
        };
        timer.schedule(task, 0, 100);

        WindowManager windowManager = ((WindowManager)
getSystemService(Context.WINDOW_SERVICE));
        Display display = windowManager.getDefaultDisplay();
        rotation = display.getRotation();
    }

    @Override
    protected void onPause(){
        super.onPause();
        mSensorManager.unregisterListener(this);
        sensorManager.unregisterListener(listener);
        timer.cancel();
```

```
    }
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        final float alp = 0.97f;
        synchronized (this) {
            if(sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER){
                mGrav[0] = alp*mGrav[0] + (1-alp)*sensorEvent.values[0];
                mGrav[1] = alp*mGrav[1] + (1-alp)*sensorEvent.values[1];
                mGrav[2] = alp*mGrav[2] + (1-alp)*sensorEvent.values[2];
            }
            if(sensorEvent.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD){
                mGeo[0] = alp*mGeo[0] + (1-alp)*sensorEvent.values[0];
                mGeo[1] = alp*mGeo[1] + (1-alp)*sensorEvent.values[1];
                mGeo[2] = alp*mGeo[2] + (1-alp)*sensorEvent.values[2];
            }
            float R[] = new float[9];
            float I[] = new float[9];

            boolean success = SensorManager.getRotationMatrix(R,I,mGrav,mGeo);
            if(success){
                float orientation[] = new float[3];
                SensorManager.getOrientation(R,orientation);
                azimuth = (float)Math.toDegrees(orientation[0]);
                azimuth = (azimuth+360)%360;

                Animation anim = new RotateAnimation(-currentAzimuth,-
azimuth,Animation.RELATIVE_TO_SELF,0.5f,Animation.RELATIVE_TO_SELF,0.5f);
                currentAzimuth = azimuth;

                anim.setDuration(500);
                anim.setRepeatCount(0);
                anim.setFillAfter(true);

                imageView.startAnimation(anim);
            }
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }
    String format(float values[]) {
        return String.format("%1$.1f",Math.abs(values[1]), Math.abs(values[1]),
Math.abs(values[1]));
    }
    void showInfo() {
        ImageView panel = (ImageView) findViewById(R.id.solarpanel);
        sb.setLength(0);
        sb.append("" + format(valuesResult) + "°");
        tvText.setText(sb);
        panel.setRotation(Math.abs(valuesResult[1]));
    }
    float[] r = new float[9];
    void getDeviceOrientation() {
        SensorManager.getRotationMatrix(r, null, valuesAccel, valuesMagnet);
        SensorManager.getOrientation(r, valuesResult);

        valuesResult[0] = (float) Math.toDegrees(valuesResult[0]);
        valuesResult[1] = (float) Math.toDegrees(valuesResult[1]);
        valuesResult[2] = (float) Math.toDegrees(valuesResult[2]);
        return;
    }
```

```
float[] inR = new float[9];
float[] outR = new float[9];

void getActualDeviceOrientation() {
    SensorManager.getRotationMatrix(inR, null, valuesAccel, valuesMagnet);
    int x_axis = SensorManager.AXIS_X;
    int y_axis = SensorManager.AXIS_Y;
    switch (rotation) {
        case (Surface.ROTATION_0): break;
        case (Surface.ROTATION_90):
            x_axis = SensorManager.AXIS_Y;
            y_axis = SensorManager.AXIS_MINUS_X;
            break;
        case (Surface.ROTATION_180):
            y_axis = SensorManager.AXIS_MINUS_Y;
            break;
        case (Surface.ROTATION_270):
            x_axis = SensorManager.AXIS_MINUS_Y;
            y_axis = SensorManager.AXIS_X;
            break;
        default: break;
    }
    SensorManager.remapCoordinateSystem(inR, x_axis, y_axis, outR);
    SensorManager.getOrientation(outR, valuesResult2);
    valuesResult2[0] = (float) Math.toDegrees(valuesResult2[0]);
    valuesResult2[1] = (float) Math.toDegrees(valuesResult2[1]);
    valuesResult2[2] = (float) Math.toDegrees(valuesResult2[2]);
    return;
}

float[] valuesAccel = new float[3];
float[] valuesMagnet = new float[3];
float[] valuesResult = new float[3];
float[] valuesResult2 = new float[3];


SensorEventListener listener = new SensorEventListener() {

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        switch (event.sensor.getType()) {
            case Sensor.TYPE_ACCELEROMETER:
                for (int i=0; i < 3; i++){
                    valuesAccel[i] = event.values[i];
                }
                break;
            case Sensor.TYPE_MAGNETIC_FIELD:
                for (int i=0; i < 3; i++){
                    valuesMagnet[i] = event.values[i];
                }
                break;
        }
    }
};
```

DeviceActivity.java

```
package com.example.solar;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.text.DecimalFormat;


 public class DeviceActivity extends AppCompatActivity {

    final String LOG_TAG = "myLogs";
   double InsolMID ;
   double M2 ;
   double loss ;
   double invertor;
   double[] inv = {0,0,0,0,0,0,0,0,0,0,0,0,0,0};


    String namedevice[] = {" ","Світлодіодний світильник", "Енергозберугаюча лампа", "Лампа
накалювання",
    "LED телевізор", "ЖК-телевізор", "Радіоприймач", "Гральна консоль", "DVD/CD приймач",
"WiFI роутер",
    "Ноутбук","Стаціонарний комп'ютер","Вентилятор", "Холодильник","Тостер","СВЧ","Пральна
машина",
    "Фен", "Інше"};
   int powerdevice[] = {0 ,15, 20, 40, 55, 70, 30, 150, 70, 5, 60, 200, 45, 350, 850, 1300,
1500, 60, 0};

    String nul = "";


    Spinner spin1;
    EditText pw1,pw2;
    EditText timefield1;
    EditText numberfield1;
    EditText powerfield1;
    TextView totalpower1;

    DecimalFormat decimalFormat = new DecimalFormat( "#.#" );

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_device);
```

```
Bundle arguments = getIntent().getExtras();
InsolMID = arguments.getDouble("InsolMID");

spin1 = (Spinner)findViewById(R.id.spinner);
timefield1 = (EditText)findViewById(R.id.time);
numberfield1 = (EditText)findViewById(R.id.numberfield);
powerfield1 = (EditText)findViewById(R.id.powerfield);
totalpower1 = (TextView)findViewById(R.id.totalpower);

TextView fullpower = (TextView)findViewById(R.id.fullpower);
TextView fullpowerloss = (TextView)findViewById(R.id.fullpowerwithloss);
TextView panelm2 = (TextView)findViewById(R.id.panelm2);

btnAdd = (Button) findViewById(R.id.btnAdd);
btnAdd.setOnClickListener(this);
btnRead = (Button) findViewById(R.id.btnRead);
btnRead.setOnClickListener(this);
btnClear = (Button) findViewById(R.id.btnClear);
btnClear.setOnClickListener(this);



etName = (EditText) findViewById(R.id.etName);
etEmail = (EditText) findViewById(R.id.etEmail);


dbHelper = new DBHelper(this);



ListView lvMain = (ListView) findViewById(R.id.lvMain);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, namedevice);
lvMain.setAdapter(adapter);

Button button = (Button) findViewById(R.id.returnbutton);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(DeviceActivity.this,MainActivity.class);
        startActivity(intent);
    }
});
Button buttonnext = (Button) findViewById(R.id.nextbutton);
buttonnext.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(DeviceActivity.this,DeviceContinueActivity.class);
        for(int i=0; i<14;i++){
            invertor+=inv[i];
        }
        intent.putExtra("M2",M2);
        intent.putExtra("loss",loss);
        intent.putExtra("invertor",invertor);
        startActivity(intent);
    }
});

ImageView equilimg1 = (ImageView)findViewById(R.id.equilbut1);
equilimg1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        double absolutepower = 0;
        if(!nul.equals(totalpower1.getText().toString())) absolutepower+=
Double.parseDouble(totalpower1.getText().toString());
fullpower.setText(decimalFormat.format(absolutepower));
    }
});
ImageView equilimg2 = (ImageView)findViewById(R.id.equilbut2);
equilimg2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(!nul.equals(fullpower.getText().toString())) {
            double temp = Double.parseDouble(fullpower.getText().toString());
            temp *= 1.15;
            decimalFormat.format(temp);
            loss = temp;
            String result = String.format("%.3f",temp);
            fullpowerloss.setText(result);
        }
    }
});



ImageView equilimg3 = (ImageView)findViewById(R.id.equilbut3);
equilimg3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(!nul.equals(fullpowerloss.getText().toString())) {
            double temp = loss;
            temp /= (InsolMID*0.17);
            if (temp < 0.03){
                temp = 0.03;
            }
            decimalFormat.format(temp);
            M2 = temp;
            String result = String.format("%.3f",temp);
            panelm2.setText(result);
        }
    }
});

spin1.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int
position, long id) {
        pw1 = (EditText)findViewById(R.id.powerfield) ;

        String text = spin1.getSelectedItem().toString();
         for(int i = 0; i < 18; i++){
             if( text.equals(namedevice[i])){
                 pw1.setText(String.valueOf( powerdevice[i]));
             }
         }

    }
    @Override
    public void onNothingSelected(AdapterView<?> parentView) {
    }
});

timefield1.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
```

```
        if (!hasFocus) {
            String timetext = timefield1.getText().toString();
            if(!nul.equals(timefield1.getText().toString())){
                if(Double.parseDouble(timetext)<0){
                    timefield1.setText("0");
                }
                if(Double.parseDouble(timetext)>24){
                    timefield1.setText("24");
                }
            }
        }
        if(!nul.equals(numberfield1.getText().toString()) &&
!nul.equals(timefield1.getText().toString()) &&
!nul.equals(powerfield1.getText().toString())){
            double x = Double.parseDouble(timefield1.getText().toString());
            double y = Double.parseDouble(powerfield1.getText().toString());
            double z = Double.parseDouble(numberfield1.getText().toString());
            inv[0]=y*z;
            totalpower1.setText(Double.toString(x*y*z));
        }
    }

});

numberfield1.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if (!hasFocus) {
            if(!nul.equals(numberfield1.getText().toString()) &&
!nul.equals(timefield1.getText().toString()) &&
!nul.equals(powerfield1.getText().toString())){
                double x = Double.parseDouble(timefield1.getText().toString());
                double y = Double.parseDouble(powerfield1.getText().toString());
                double z = Double.parseDouble(numberfield1.getText().toString());
                inv[0]=y*z;
                totalpower1.setText(Double.toString(x*y*z));
            }
        }

    }

});
powerfield1.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if (!hasFocus) {
            if(!nul.equals(numberfield1.getText().toString()) &&
!nul.equals(timefield1.getText().toString()) &&
!nul.equals(powerfield1.getText().toString())){
                double x = Double.parseDouble(timefield1.getText().toString());
                double y = Double.parseDouble(powerfield1.getText().toString());
                double z = Double.parseDouble(numberfield1.getText().toString());
                inv[0]=y*z;
                totalpower1.setText(Double.toString(x*y*z));
            }
        }

    }

});
```

DeviceContinueActivity.java

```
package com.example.solar;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.text.DecimalFormat;


public class DeviceContinueActivity extends AppCompatActivity {

    double M2;
    double loss;
    String nul = "";
    double invertor;

    DecimalFormat decimalFormat = new DecimalFormat( "#.#" );

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_device_continue);
        Bundle arguments = getIntent().getExtras();
        if(arguments != null){
            M2 = arguments.getDouble("M2");
            loss = arguments.getDouble("loss");
            invertor = arguments.getDouble("invertor");
        }

        Button button = (Button) findViewById(R.id.returnbutton);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(DeviceContinueActivity.this,MainActivity.class);
                startActivity(intent);
            }
        });

        Button b1 = (Button) findViewById(R.id.but1);
        Button b2 = (Button) findViewById(R.id.but2);
        Button b3 = (Button) findViewById(R.id.but3);
        Button b4 = (Button) findViewById(R.id.but4);

        decimalFormat.format(loss);
        String resultlost = String.format("%.0f",loss);
        b1.setText(resultlost+"BT");

        TextView bat = (TextView) findViewById(R.id.fullpower);
        double batter = (loss*36)/(24/24*0.7);
        decimalFormat.format(batter);
        String resultbat = String.format("%.0f",batter);
        bat.setText(resultbat +" ампер·год");
        TextView amperid = (TextView) findViewById(R.id.amperid);
        amperid.setText(resultbat +" ампер·год");
```

```
        EditText amperfield =(EditText) findViewById(R.id.editamper);
        ImageView equilbut2 = (ImageView) findViewById(R.id.equilbut2);
        TextView batterynumber = (TextView) findViewById(R.id.batterynumber);

        equilbut2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!nul.equals(amperfield.getText().toString())){
                    double amper = Double.parseDouble(amperfield.getText().toString());
                    double amount_of_battery = batter/amper;
                    if (amount_of_battery < 2){
                        amount_of_battery = 2;
                    }
                    amount_of_battery = Math.ceil(amount_of_battery);
                    batterynumber.setText(Double.toString(amount_of_battery));
                }
            }
        });

        TextView invertordata = (TextView) findViewById(R.id.invertordata);
        double invtmp1 = Math.ceil(invertor);
        invertordata.setText(invtmp1 +" BT");

        TextView invertorresult = (TextView) findViewById(R.id.invertorresult);
        double invtmp2 = Math.ceil(invertor*1.2);
        invertorresult.setText(invtmp2 +" BT");
    }


}
```

ДОДАТОК В

Програмне забезпечення для аналізу складу певного контингенту

Опис програмного модулю

УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ-72207_21Б 13-1

Аркушів 8

Київ — 2021

АНОТАЦІЯ

Цей розділ описує створений мобільний додаток для аналізу режимів роботи сонячних панелей. В програмі можливо отримати інформацію про оптимальне розміщення сонячних панелей, енергетичний потенціал місцевості та необхідні енергетичні характеристики обладнання для приватної сонячної електростанції. Програма була на базу середовища Android Studio з використанням мови програмування Java та мови розмітки XMLз використанням бібліотеки для роботою з базами даних SQLite.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Додаток створений для роботи з сонячними панелями. Застосунок доступний для мобільних пристроїв з операційною системою Android 4.2 або вище. Програма створена на платформі Android Studio з використанням мови програмування Java та мови розмітки XML. В якості системи керування базами даних використано бібліотеку SQLite.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програма була створена для розрахунку сонячного потенціалу місцевості та вибору оптимального обладнання для генерування та зберігання сонячної енергії. Користувач має наступні можливості:

- Відмітити свою геолокацію або геолокацію місцевості, інформацію для якої потрібно розрахувати;
- Використовуючи вбудований гіроскоп, визначити оптимальний кут нахилу сонячних панелей;
- Переглянути інформацію про сонячний потенціал місцевості;
- Заповнити таблицю зі значеннями свої електроприладів;
- Розрахувати характеристики обладнання.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Додаток складається з шести вікон, а саме:

- Головне меню.
- Вікно “Карта”.
- Вікно “Кут нахилу”.
- Вікно “Інсоляція”.
- Вікно “Прилади №1”.
- Вікно “Прилади №2”.

Головне меню надає доступ до інших вікон.

Вікно “Карта” містить карту для визначення своєї геолокації та інформацію про географічні координати.

Вікно “Кут нахилу” містить зображення компасу та сонячної панелі, що вказують на напрямок та кут нахилу сонячної панелі. Також містить чисельне значення оптимального та поточного кута.

Вікно “Інсоляція” містить графік річної інсоляції та показники за кожен місяць.

Вікно “Прилади №1” містить меню для розрахунків характеристик приладів.

Вікно “Прилади №2” продовжує розрахунки характеристик приладів, розпочатих у вікні Прилади №1”.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Android Studio 4.2.1 була обрана як середовище розробки програми, бо воно є оптимальний для створення програми для Android смартфонів.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програма міститься у APK файлі для Android пристроїв. Для його завантаження потрібно перенести APK файл на пристрій та завантажити. Після інсталяції програма буде готова для використання.

Для завантаження програми потрібно, щоб було вибрано пункт «Встановити додаток зі сторонніх джерел» у налаштуваннях пристрою.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є геолокація, дані з сенсорів телефону та інформація про електроприлади користувача.

Вихідними даними програми є дані про характеристики обладнання для сонячних батарей, розміщення сонячних панелей та сонячний потенціал місцевості.

ДОДАТОК Г

Програмне забезпечення для аналізу складу певного контингенту

Апробації

УКР.НТУУ“КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ-72207_21Б

Аркушів 2

Київ — 2021

Матеріали XIX Міжнародної науково-практичної конференції молодих вчених і студентів м. Київ, 20–23 квітня 2021 року

ТОМ 2, ст. 143-144

УДК 004.042

Студент 4 курсу, гр. ТМ-72 Юрченко Р.О.; студент 4 курсу, гр. ТМ-72 Щур В.Є.
Проф., к.т.н. Отрох С.І.

СТВОРЕННЯ ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОДЕЛЮВАННЯ ДИНАМІЧНИХ РЕЖИМІВ РОБОТИ СОНЯЧНИХ ПАНЕЛЕЙ В СКЛАДІ МІКРОЕНЕРГОСТАНЦІЇ

Наразі, питання енергозабезпечення в Україні є досить актуальним. Значною є проблема забезпечення країни твердим паливом для потреб ТЕС та ТЕЦ [1]. Також значною мірою збільшилась частка імпортованої енергії з сусідніх країн: Угорщина, Словаччина, Румунія, Білорусь [2]. Збільшення витрат бюджету на імпорт як ресурсів для електрозабезпечення, так і самої електроенергії призводить до підвищення тарифів на послуги електропостачання для населення [3]. Складно спрогнозувати динаміку цін у майбутньому, тому все частіше люди шукають способи заощадити кошти.

Все більше і більше людей переходять на альтернативні джерела енергії для власного використання, такі як сонячна та вітрова енергії [4]. Їх перевагами є незалежність від ситуації на ринку електроенергії. Також відсутня необхідність підключатися до електромережі у випадках, якщо це викликає труднощі або потребує додаткових витрат. Ще одним фактором є існування в Україні «Зеленого тарифу». Він дозволяє приватним домогосподарствам продавати надлишки виробленої й сонячного випромінювання та/або вітру енергії за вищою ціною, ніж ринкова. Відповідно до закону України «Про електроенергетику», вартість такої енергії прив'язана до євро та не потребує спеціальних ліцензій, якщо видобуток здійснюється приватними домогосподарствами [5].

Серед двох альтернатив: сонячною енергією та вітровою, мною була обрана на розгляд сонячна, бо вона має достатню (в додачу до наведених вище) кількість переваг. Термін їх зберігання та роботи досить високий і може вимірюватися десятками років. Більшість моделей має гарантію понад 25 років, при незначній втраті потужності. Одною з причин цього є відсутність рухомих частин, які б потребували б заміни, а також їх відсутність дає змогу панелям працювати безшумно. Така перевага, як безшумність, на відміну від вітряків, дозволяє встановлювати сонячні батареї на дахах будинків без додаткових незручностей. Окрім, відсутності звукового забруднення, перевагою сонячних панелей є відсутність впливу на середовище (як, наприклад, викиди шкідливих речовин). Також саму кількість згенерованої енергії можна передбачити, бо він залежить від циклу дня і ночі, тривалості сонячного дня та клімату місцевості. Але, щоб інвестиції в обладнання окупилися, потрібно спрогнозувати кількість електроенергії, необхідної для забезпечення потреб домогосподарства.

Метою даної роботи є створення програмного засобу, який, використовуючи дані про геолокацію, кількість сонячних днів, клімат місцевості та потреби будинку, на основі наявних електроприладів, вираховує значення профіциту та дефіциту енергії в визначенні моменти часу. Оскільки дані о тривалості освітлення та його інтенсивності можна вважати надійними, то джерелом невизначеності є погода в визначений момент часу, яка буде прогнозуватися за допомогою інформації о кліматі. Також користувач зможе обрати перелік пристроїв, які плануються під'єднати до мережі, та побачити, які саме характеристики вони мають. При відсутності деяких приладів, їх можливо буде додати самостійно.

Дана інформація слугує основою для визначення характеристик обладнання, що задовольнить потреби власника. Кількість енергії, яка потрібна для забезпечення території протягом доби (умовно кВт·день), використовується для вибору сонячних батарей. Ці дані також використовується для визначення ємності акумуляторних батарей, так, щоб зібрати достатню кількість енергії для забезпечити дому світлом в період, коли вона не генерується. І ще одним важливим приладом є інвертор – прилад, що перетворює постійний струм від сонячних батарей у змінний, який використовується у мережах з напругою 220В або 380В. Його характеристики визначаються максимальною потужністю яку може потребувати система. Наступні прилади: лампочки, зарядні пристрої телефонів та ноутбуків потребують невеликого значення потужності інвертора та споживають незначну кількість енергії (енергоощадні лампи потребують 15 - 50В, зарядні пристрої - 10В, ноутбук – 50 В). Холодильники, кондиціонери та телевізори потребують більшої потужності (150 - 250В), але незважаючи на постійне підключення до мережі, вони більшу частину часу неактивні, тому не потребують надто багато енергії. Найбільше навантаження на систему створює велика побутова техніка та нагрівачі. Праска або електричний чайник потребують потужності понад 2 кВ, що потребує потужного інвертора, а одночасної роботи декількох таких приладів система може і не витримати. На підставі цих даних планується створення графіку, який продемонструє пікові значення потужностей необхідних системі, та рекомендацій, щодо зміни часу роботи деяких приладів (наприклад підігрів води бойлером у час з мінімальним навантаженням на систему).

Передбачається, що даний програмний продукт буде корисний для власників літніх будинків, дач та садиб, адже сонячні батареї найефективніше саме у період весна-літо. Також можлива інтеграція з іншими джерелами енергії, як вітрогенератор, для отримання енергії взимку, або підключення до електромережі та використання власної мікроенергостанції для зменшення оплати за електроенергію.

Перелік посилань:

1. nerc.gov.ua/?news=11181
2. ua.energy/category/analityka/
3. nerc.gov.ua/?id=15006
4. saee.gov.ua/sites/default/files/2018_19_report_07_02_2019.pdf
5. zakon.rada.gov.ua/laws/show/514-19#Text