НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ
КАФЕДРА ЕЛЕКТРОННИХ ПРИСТРОЇВ ТА СИСТЕМ

«На правах рукопису»
УДК 621.314

«До захисту допущено»
Завідувач кафедри

_____ Євген ВЕРБИЦЬКИЙ
(підпис) (ім'я ПРІЗВИЩЕ)

"_____" _____2024р.

Магістерська дисертація
на здобуття ступеня магістра

зі спеціальності 171 Електроніка
(код і назва)

освітня програма Електронні компоненти та системи

на тему: Керування електроспоживання в SmartGrid

Виконав: студент ІІ курсу, групи ДС-11мп
(шифр групи)

Яцишин Василь Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник проф.каф. ЕПС, д.т.н. проф. Юлія ЯМНЕНКО _____
(посада, науковий ступінь, вчене звання, ім'я ПРІЗВИЩЕ) (підпис)

Консультант _____ _____
(посада, науковий ступінь, вчене звання, ім'я ПРІЗВИЩЕ) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, ім'я ПРІЗВИЩЕ) (підпис)

Консультант по нормоконтролю доц. каф. ЕПС, к.т.н., доц. Лариса БАТРАК _____
(посада, науковий ступінь, вчене звання, ім'я ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024 року

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

Факультет Електроніки
(повна назва)

Кафедра Електронних компоненти та системи
(повна назва)

Рівень вищої освіт – другий (магістерський) за освітньо - професійною програмою

Спеціальність 171 Електроніка
(шифр і назва)

Освітня програма Електронні прилади та пристрої

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Євген ВЕРБИЦЬКИЙ
(підпис) (ім’я ПРІЗВИЩЕ)

«___» _____

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Яцишину Василю Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації Керування електроспоживання в SmartGrid

науковий керівник дисертації д.т.н. проф. Юлія ЯМНЕНКО
(ім’я ПРІЗВИЩЕ, посада, науковий ступінь, вчене звання)

затверджені наказом по університету
від « » 2024 року
2. Термін подання студентом дисертації 03.06.2024р.
3. Об’єкт дослідження енергоспоживання різними типами споживачів

4. Вихідні дані кількість годин світлового дня протягом всього року, використання електроенергії з різних приладів одного з ресторанів.

5. Перелік завдань, які потрібно розробити розробити алгоритм, який буде обчислювати середнє споживання електроенергії за обраний період, а також обчислювати кількість енергії яку можна зекономити за допомогою дімера. Прогнозування електровитрат за допомогою нейромережі.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу презентація

7. Орієнтовний перелік публікацій Яцишин В. С., Ямненко Ю. С., Сарибога Г. В. Система виявлення небезпечних речовин у повітрі на базі інтернету речей. Вісник КрНУ імені Михайла Остроградського. 2023. № 1. С. 67–72. DOI https://doi.org/10.32782/1995-0519.2023.1.9 – опубліковано.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-5	проф. Т.О. Терещенко	15.09.2022	03.02.2024

9. Дата видачі завдання 15 вересня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз енерговитрат у системі SmartGrid	03.02.2024 – 22.02.2024	
2	Основні методи керування споживанням електроенергії	21.02.2024 – 01.03.2024	
3	Використання нейронних мереж та машинного навчання в керуванні SmartGrid	02.03.2024 – 17.03.2024	
4	Розробка алгоритму відстеження використання електроенергії	18.03.2024 – 11.04.2024	
5	Реалізація системи відстеження споживання	12.04.2024 – 17.04.2024	
6	Розробка блок схем для програми	18.04.2024 – 20.04.2024	
7	Написання дисертації та тестування і налаштування системи	21.04.2024 - 15.05.2024	

Студент _____ Василь ЯЦИШИН
(підпис) (ім'я ПРІЗВИЩЕ)

Науковий керівник дисертації _____ Юлія ЯМНЕНКО
(підпис) (ім'я ПРІЗВИЩЕ)

АНОТАЦІЯ

В магістерській дисертації описується проблема, яка пов’язана з використанням електроенергії, а саме не ефективне її використання, велике навантаження на енергосистему та економія на використанні. Розглядаються способи відстеження енерговитрат для приміщень, що дозволить моніторити витрати і відповідно оперувати цими даними. Окрім цього описується можливість використання нейронних мереж для системи SmartGrid, їхні переваги та недоліки. Саме це дозволяє поглиблено розібрати керування енергоносіями за допомогою машинного навчання. Розглядається можливість керування освітленям за допомогою дімерів, керування споживанням різних пристроїв на основі історичних даних зібраних під час вимірювань, прогнозування споживання електроенергії за допомогою нейронних мереж, економія електроенергії при керуванні освітленням, а також керування загальною потужністю споживання системи. Для алгоритму було розроблено програмне забезпечення, яке за допомогою даних взятих з баз даних, та користувацьких налаштувань може розрахувати електроспоживання, а також його економію при різних факторах впливу на систему. Крім цього було описано можливість виведення алгоритму на сервер та використання його користувачами за допомогою прикладного програмного інтерфейсу. Реалізація системи відстеження споживання за допомогою мобільного застосунку.

Ключові слова: енергоспоживання, енергоефективність, SmartGrid, дімер, алгоритм, сервер.

ANNOTATION

The master's thesis describes the problem associated with the use of electricity, namely its inefficient use, heavy load on the power system, and savings on consumption. It considers ways of tracking energy consumption for premises, which will allow monitoring costs and manipulating this data accordingly. In addition, the

possibility of using neural networks for the SmartGrid system, their advantages and disadvantages are described. This is what allows us to analyze energy management using machine learning in depth. We consider the possibility of controlling lighting, controlling the consumption of various devices based on historical data collected during measurements, predicting electricity consumption using neural networks, saving electricity when controlling lighting, and controlling the total power consumption of the system. Software has been developed for the algorithm, which, using data taken from databases and user settings, can calculate electricity consumption and its savings under various factors affecting the system. In addition, the possibility of outputting the algorithm to the server and using it by users through an application program interface was described. Implementation of a consumption tracking system using a mobile application.

Keywords: energy consumption, energy efficiency, SmartGrid, dimmer, algorithm, server.

ЗМІСТ

ВСТУП

Актуальність. В умовах сучасного розвитку електроенергетичної галузі питання ефективного використання електроенергії стоїть надзвичайно гостро. Одним з ефективних напрямків розвитку передбачає широке розповсюдження систем SmartGrid, що має ряд переваг як для індивідуальних споживачів, так і на рівні об”єднань домогосподарств, котеджних містечок, районів та навіть цілих міст. Керування електроспоживанням у цьому контексті вирішально важливе для керування та балансування попитом та пропозицією електроенергії. Дослідження в галузі керування електроспоживанням має велике значення для подальшого розвитку енергетичної галузі та забезпечення сталого енергетичного майбутнього.

Одним із ключових аспектів SmartGrid є керування електроспоживанням, що передбачає можливість активно реагувати на зміни у споживанні електроенергії, забезпечувати ефективний розподіл енергії та зменшувати втрати.

Таким чином, тема магістерської роботи, що присвячена вирішенню задачі керування та прогнозування, є актуальною. Вирішення поставлених у дисертації задач сприятиме підвищенню енергетичної ефективності SmartGrid, зменшенню витрат електроенергії та грошових коштів.

Зв’язок роботи з науковими програмами, планами, темами. Дисертація була підготовлена відповідно до науково-дослідного плану кафедри електронних пристроїв та систем» Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського.

Мета і завдання дослідження. Мета дослідження полягає у розробці ефективних алгоритмів керування, вивченні та аналізі стратегій керування у системі SmartGrid, моделюванні системи інтелектуального керування та розробці відповідного програмного забезпечення з можливістю прогнозування за допомогою нейронних мереж. Завдання полягають у вивченні сучасних рішень керування системою та її прогнозування, аналіз вже наявних

інфраструктур, встановлення факторів, які можуть впливати на ефективність, а також розробка нових алгоритмів та оцінку їх ефективності.

Об'єкт і предмет дослідження. Об'єктом досліджень є процес керування та прогнзування електроспоживанням у SmartGrid, а предметом дослідження є прогнозування споживання електроенергії в системі SmartGrid.

Методи дослідження. Для досягення результатів використовувались наступні методи дослідення, а саме літературний аналіз, математичне моделювання, а також статистичний аналіз. Літературний аналіз був використаний для аналізування наукових джерел повязаних з керуванням електроспоживанням в системі SmartGrid. Математичне моделювання використовується для розробки алгоритмів керування електроспоживанням у системі. Статистичний аналіз використовується для обробки та аналізу даних, також це дозволяє здійснювати кількісну оцінку результатів досліження, робити висновки щодо отриманих результатів.

Наукова новизна одержаних результатів: полягає в розробці нового підходу до керування електроживленняи у системі, використовуючи програмування та електроніку. Дослідження вдосконалює вирахування втрат електроенергії при різних сценаріях, а також при впливі різних факторів. Отримані результати дозволяють підвищити ефективність системи в сфері електроспоживання, збільшити її надійність.

Практичне значення одержаних результатів:

- Розроблено алгоритм який вираховує електроспоживання в залежності від різних типів споживачів та величини світлового дня;
- Анлізування отриманих даних для зменшення споживання електроенергії.

Даний алгоритм може убти використаний як для прогнозування використання електроенергії так і для керування споживачами, що значно полегшує моніторинг та керування системою.

Публікації. Система виявлення небезпечних речовин у повітрі на базі інтернету речей для вісника КрНУ імені Михайла Остроградського. Участь у

конференції молодих вчених «Електроніка - 2024» з темою: «Керування загальною потужністю в системі Smart Grid».

Структура та обсяг дисертації. Робота складається з 4 розділів. В першому розділі розглянуто енерговитрати у системі Smart Grid, а саме способи відстеження енерговитрат та можливі способи прогнозування; в другому розділі розглянуті методи керування енергоспоживанням, використовуючи дімери, сценарне керування, керування за допомого. нейромереж; в третьому розділі описана розробка алгоритму прогнозування енерговитрат лампами різних типів протягом темної пори доби, а також прогнозування використання електроенергії використовуючи нейромережу; в четвертому розділі описаний проект стартап проекту. Робота містить 73 сторінки, 21 рисунок, 25 таблиць, список використаних джерел з 30 найменувань на 3 сторінках.

1. АНАЛІЗ ЕНЕРГОВИТРАТ У СИСТЕМІ SMARTGRID

1.1. Поняття Smart Grid

У сучасному світі сталого розвитку та технологічних інновацій електроенергетика займає особливе місце. Концепція SmartGrid, або "розумної мережі", є однією з найбільш важливих і перспективних галузей в галузі електроенергетики. Smart Grid’s – це електромережі, які використовують цифрові технології, датчики та програмне забезпечення для кращого узгодження пропозиції та попиту на електроенергію в реальному часі, мінімізуючи витрати та зберігаючи стабільність і надійність мережі[1].

Основними функціями такої системи є автоматизація та керування, інтеграція відновлювальних джерел енергії, моніторинг та діагностика, взаємодія зі споживачами. Розгладаючи автоматизацію та керування, можна сказати, що Smart Grid використовує автоматизовані системи керування, що дозволяють забезпечити ефективність роботу системи та швидко реагувати на зовнішні збурення. Важливою для забезпечення стабільності електропостачання є реалізація розподіленої генерації за рахунок інтеграції відновлюваних джерел енергії, наприклад, сонячні чи вітрові. Постійний моніторинг стану електромережі та окремих її складових не менш важливий, адже дозволяє вчасно виявляти проблеми чи нештатні ситуації та реагувати на них шляхом застосування відповідних стратегій керування. Навіть за відсутності проблем або нештатних ситуацій моніторинг роботи системи дозволяє забезпечити ефективне керування у відповідності до поточного режиму роботи. Споживач як складова частина Smart Grid активно взаємодіє із системою, що призводить до адаптації режимів її роботи та ефективному споживанню електроенергії.

Smart Grid має величезну роль у створення високоефективних систем для майбутнього, адже сприяє забезпеченню безпеки, економічному зростанню та зниженню забруднення навколишнього середовища.

Дана система надзвичайно поширена в інших країнах світу і має певні відмінності, адже повинна відповідати діючим стандартам та особливостям роботи електротехнічних комплексів. Водночас незалежно від географічного розташування Smart Grid функціонують згідно прийнятим міжнародним стандартам - зокрема, стандартам щодо якості електроенергії [2,3].Наприклад, в США, де електромережа є однією з найбільших у світі, системи Smart Grid впроваджуються для підвищення ефективності та надійності електропостачання. Так, в місті Болдвін Парк була впроваджена система, яка дозволяє виробляти, зберігати та використовувати енергію більш ефективно за допомогою сонячних панелей, акумуляторів та автонматизованих систем керування[4].

В ЄС також дуже активно почали використовувати дану концепцію для використання електроенергії. Прикладом є Амстердам в Нідерландах, де на великій кількості будинків є сонячні панелі та сучасні лічильники, що дозволяє оптимізувати споживання та виробництво електроенергії[5].

Система працює наступним чином: лічильники передають дані про споживання електроенергії в реальному часі, і на підставі цих даних та запрограмованих алгоритмів роботи система керування Smart Grid реалізує ту чи іншу стратегію, що впливає на режими роботи окремих пристроїв, функціональних підсистем та системи в цілому. В інтервали ранкових та вечірніх піків споживання (характерних для побутового сектору) запроваджуються часові затримки вмикання для деяких пристроїв, для яких введення таких затримок не призводить до дискомфорту споживача чи збоїв у функціональності виконання системою своїх задач.

Схематично структура Smart Grid представлена на рис.1.1.

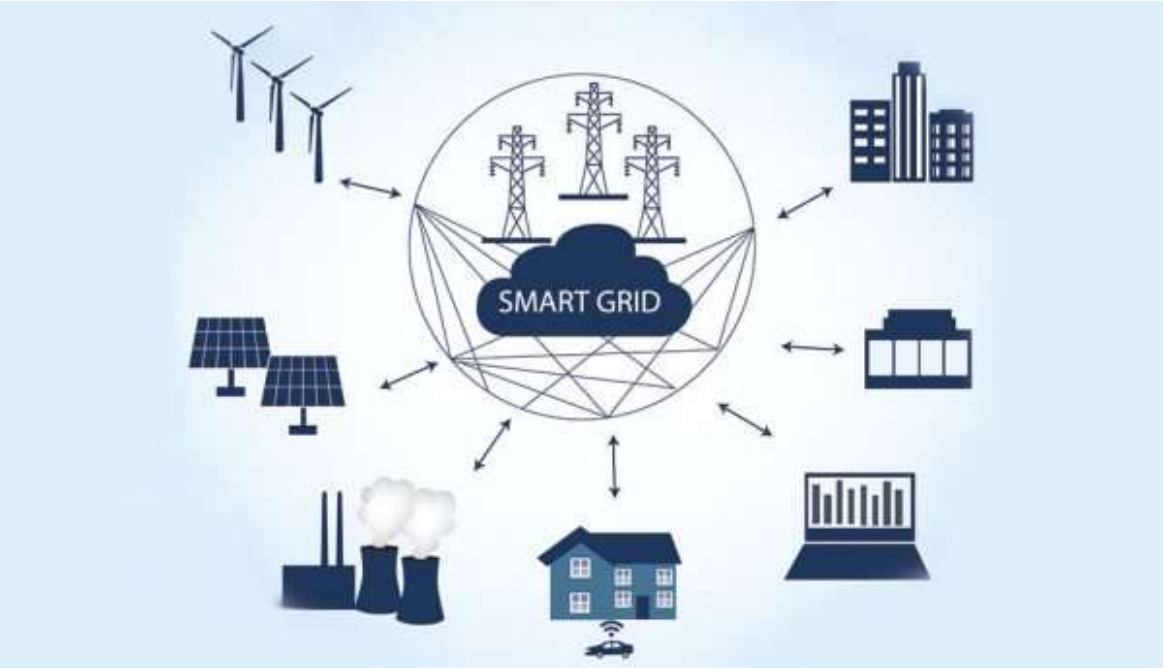


Рис.1.1. Система Smart Grid

Великою перевагою даної системи є перехід споживачів до зеленої електроенергії, що має набагато більше переваг від традиційного способу отримання електроенергії.

Розвиток Smart Grid та Micro Grid супроводжувався появою і подальшим широким впровадженням концепції Інтернету речей (Internet of Things, IoT). IoT описує сукупність об’єднаних між собою фізичних об'єктів, які обмінюються даними та взаємодіють через Інтернет без прямого втручання людини за передустановленими алгоритмами. Передбачається, що об'єкти оснащені різноманітними датчиками (температури, вологості, рівня освітлення, звуку, руху, вібрації, газів, рівня води, диму та ін.) для забезпечення збору, передачі та обробки даних. IoT передбачає підключення фізичних об'єктів до загальної мережі за допомогою різних технологій, таких як Wi-Fi, Bluetooth, Zigbee, RFID, LTE, LoRaWAN[6].

Концепція “Інтернет речей” (Internet of Things, IoT), що розвивалася для Smart Grid та Micro Grid протягом тривалого часу , розвинулася у більш глобальну "Інтернет енергії” (IoE) [7]. Internet of Energy — це технологічний термін, який стосується модернізації та автоматизації електроенергетичної

інфраструктури для виробників і споживачів енергії. В рамках IoE процеси виробництва електроенергії розвиваються в напрямку більшої ефективності та екологічності, зменшуючи обсяги відходів на виробництво. Концепція енергетики нового покоління передбачає, що кожне нове підключення до загальної мережі здійснюватиметься так само легко і швидко, як підключення користувачів до мережі Інтернет – за принципом “plug&play”.

1.2. Способи відстеження електровитрат

Зважаючи на особливості Smart Grid можна виділити основні способи за допомогою яких можна відстежувати електровитрати та робити аналіз системи:

- системи вимірювання споживання електроенергії;
- датчики розподільчих мереж;
- системи дистанційного моніторингу та керування.

Першим способом є використання розумних лічильників, які забезпечують більш точне вимірювання споживання елекроенергії та мають можливість передавати дані в режим реального часу через мережу. Дані пристрої дозволяються користувачам або кваліфікованим операторам мережі моніторити електроспоживання.

Для того щоб розрахувати споживання за домогою лічильників потрібно знати два параметри, а саме потужність пристрою який використовується в мережі, і час використоання пристрою (1.1).

$$E = P \cdot t, \tag{1.1}$$

де: E – енергія, витрачена на споживання електроенергії(в кітоват-годинах,

кВт· год), P – потужність пристрою(в ватах, Вт), t – час роботи(в годинах, год).

Підхід який використовує датчики розподільчих мереж передбачає собою встановлення таких пристроїв на різних елементах розподільчих мереж, а саме

трансформатори, стовпи, кабелі тощо, з метою постійного відстеження параметрів мережі. Таким способом можна відслідковувати рівень напруги, струм, температуру, частоту, споживання енергії і так далі. Такі датчики можуть бути підключені до центральної системи моніторингу, яка збиратиме дані в режимі реального часу. За допомогою цього можна отримати досить точну інформацію про стан мережі. Основними перевагами такого способу є швидке виявлення проблем, на різних ділянках, моніторинг стану мережі, вдосконалення мережі за зібраними данимим, збільшення надійності мережі.

Дистанційний спосіб дозволяє віддалено контролювати та спостерігати за параметрами обладнання та мережами електропостачання. За допомогою даного способу можна швидко вмикати потрібні режими роботи пристроїв, які задіяні в системі, наприклад зміну напруги на певних ділянках системи, вмикати або вимикати обладнання, регулювати загальну потужність, встановлювати сценарії або режими роботи. Ще однією перевагою такого способу є автоматизування більшості процесів, що зменшує вірогідність помилитись при експлуатації пристроїв. При цьому такий спосіб має проблему надійності та безпеки, адже система працює за допомогою інтернет протоколів, які можуть бути уражені, якщо не використовувати шифрування, або захищені протоколи.

Різні системи моніторингу та управління енергоспоживанням можуть аналізувати або навіть і прогнозувати енергоспоживання, реагувати на зміни в навантаженні та зменшувати витрати енергії. За допомогою аналізування та прогнозування можна не тільки ефективно використовувати електроенергію, але й робити історичний аналіз даних про споживання, прогнорзуватии пік навантаження при певних умовах, та оптимізовувати ренераційні та розподільчі системи.

Окрім цього система може використовувати розумні мережі, використовуючи технологію зберігання електроенергії, надаючи доступ до інформації про їхнє енергоспоживання та можливість активної участі у програмах з енергоефективності та управління напантаженням. Найголовнішою

перевагою є збереження енергії та підвищення ефективності системи за допомогою використання акумуляторів енергії, технологій зберігання тепла, або розподілену генерацію енергії. Завдяки цій технології стає можливим швидке реагування на зміну навантаження в мережі, що часто може убезпечити від аварійних ситуацій. При зміні навантаження в системі розумна мереже може перерозподілити енергію з менш навантажених ділянок на ті, де потреба більша, так і навпаки.

Smart Grid також можуть набагато швидше реагувати на аварійні ситуації, та виконувати перші безпекові заходи. Вони можуть виявляти перебої в електропостачанні або відключення і в залежності від ситуаціїї, перемикати джерела енергії, та відновлювати роботу мережі в разі аварій. Окрім цього може ізольовувати певну ділянку, для уникнення пожеж або інших бід.

Ще одною перевагою є керування споживачами, задля ефективного управління споживачами електроенергії. Дана функція може включати в собі програми гнучкого ціноутворення, розумне керування електропристроями та інші інціатими, щоб стимулювати споживачів до енергоефективності.

Використовуючи дані методи можна сутєво зекономити витрати не тільки користувачів, а й в цілому країни, адже використовуючи таку систему можна зменшити електровитрати на 3-4% завдяки економії споживання електроенергії. Також перевагою є зменшення внутрішніх відключень на 7-9%, а також зменшення зносу пристроїв на 3-5% [8]. Україна за січень 2022 року споживала на комунально побутові послуги приблизно 1417.8 млн кВт· год. Від цієї суми 4% економії складатиме 56.712 млн кВт· год, що є дійсно суттєвим зменшенням витрат для суспільства.

1.3. Прогнозування витрат електроенергії

Прогнозування витрат електроенергії є важливим завданням для ефективного керування електромережами та планування енергетичних ресурсів. Цей процес включає аналіз історичних даних про споживання електроенергії,

ідентифікацію тенденцій та факторів, що впливають на споживання, і використання моделей прогнозування для передбачення майбутніх витрат. Далі наведені кілька методів, які можна використовувати для прогнозування витрат електроенергії, а саме:

- експотенційне згладжування [9]: цей метод базується на ваговному середньому, де більший ваговий коефіцієнт надається новішим даним. Він дозволяє прогнозувати майбутні витрати, враховуючи тенденції та сезонні зміни;
- метод ковзного середнього [10]: даний метод використовується для виявлення тенденцій у часових рядах даних. Він обчислює середнє значення попередніх даних за певний період часу і використовує його для прогнозування майбутніх витрат;
- метод ARIMA (autoregressive integrated moving average) [11]: цей метод є розширенням методу ковзного середнього і дозволяє моделювати як тенденції, так і сезонні зміни в часових рядах даних. Він включає авторегресійну, інтегровану та ковзну середню складові;
- метод нейронних мереж [12]: нейронні мережі можуть бути використані для прогнозування витрат електроенергії, особливо в тих випадках, коли є складні нелінійні залежності між вхідними змінними та споживанням електроенергії;
- метод регресії [13]: даний метод може бути використаний для ідентифікації зв'язку між різними факторами, такими як температура, час доби, типи пристроїв тощо, і витратами електроенергії;
- метод гібридних моделей: цей підхід поєднує кілька методів прогнозування для отримання кращих результатів. Наприклад, може бути застосовано комбінацію методів ARIMA та нейронних мереж.

1.3.1. Нейронні мережі

Нейронні мережі - це обчислювальні системи, які моделюють структуру та функціонування нейронних систем у мозку. Вони використовуються для

аналізу складних зв'язків у великих наборах даних, включаючи часові ряди споживання електроенергії. Нейронні мережі можуть бути навчені на історичних даних про споживання електроенергії, враховуючи різноманітні фактори, такі як погода, дата та час, типи пристроїв тощо. Після навчання мережі можна використовувати для прогнозування майбутніх витрат електроенергії на основі нових вхідних даних. Однією з переваг нейронних мереж є їх здатність моделювати складні нелінійні зв'язки між різними вхідними змінними та витратами електроенергії. Вони можуть виявляти складні патерни та взаємозв'язки, які можуть бути важко виявити за допомогою традиційних методів прогнозування. Для даного варіанту прогнозування можна використовувати багатошаровий персептрон. Це тип нейронної мережі, який часто називають «класичною» нейронною мережею. Дану мережу можна представити як трьохшарову, яка складається з вхідного, прихованого та вихідного шару (рис.1.2.).

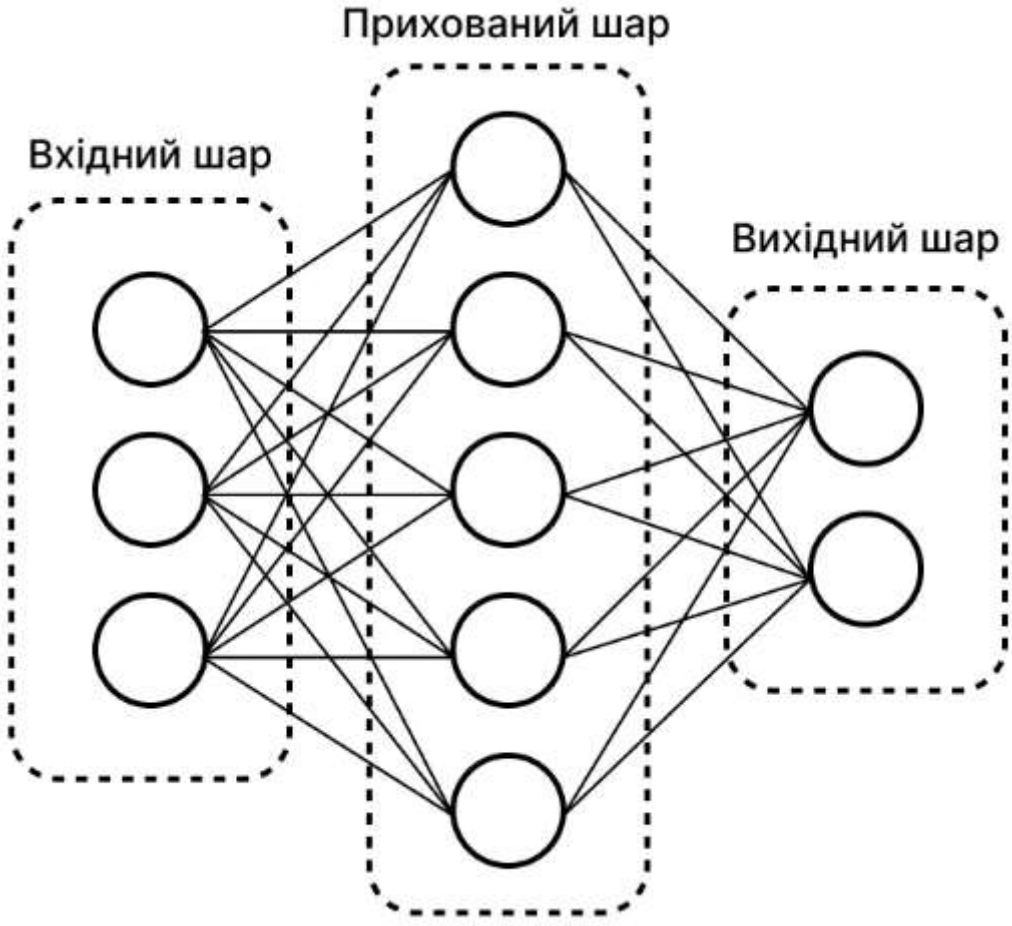


Рис.1.2. Структура багатошарового персептрону

На вхідний шар подається інформація, яка потрібна для розрахунків, для системи Smart Grid це можуть бути дані про споживання окремими пристроями, час світлового дня, навантаження у мережі протягом дня і так далі. В прихованому шарі відбувається основні складні обчислення, первичне прогнозування та ще багато логічних операцій. Таких прихованих шарів можу бути велика кількість, в залежності від складності прогнозування, різних типів вхідних даних. Після цього дані потрапляють до вихідного шару, де обробляються дані після прихованого шару, і підготовлюється фінальний етап прогнозування. Нейрони у шарах поєднані між собою за допомогою синапсів, які визначають наскільки сильно кожен вхід нейрона впливає на вихід. Дана система навчається за допомогою процесу, який називається зворотнім поширеням [14].

1.3.2. Метод регресії

Метод регресії - це статистичний метод, який використовується для визначення зв'язку між залежною змінною (у цьому випадку витратами електроенергії) та однією або декількома незалежними змінними (факторами, які впливають на споживання). Наприклад, можна використовувати метод регресії для аналізу впливу факторів, таких як температура, час доби, день тижня, святкові дні тощо, на споживання електроенергії. Модель регресії може бути побудована на основі історичних даних, і після цього може бути використана для прогнозування майбутніх витрат на основі нових даних. Метод регресії може бути особливо корисним, коли необхідно враховувати прості взаємозв'язки між вхідними факторами та витратами електроенергії, а також для виявлення впливу окремих факторів на споживання.

При прогнозуванні витрат електроенергії важливо враховувати різноманітні фактори, які можуть впливати на споживання, такі як сезонність, кліматичні умови, економічні фактори та інші. Крім того, важливо регулярно

оновлювати моделі прогнозування на основі нових даних та коригувати їх для досягнення кращих результатів.

1.3.3. Метод ARIMA

Метод ARIMA є потужним і широко використовуваним методом для прогнозування часових рядів, включаючи витрати електроенергії. Він дозволяє моделювати складні залежності між значеннями часового ряду та побудовувати прогнози на майбутнє. Основною ідеєю ARIMA є використання попередніх значень часового ряду та їх зсувів для прогнозування майбутніх значень. Модель ARIMA складається з трьох компонент: авторегресії (AR), інтегрованої складової (I) та ковзного середнього (MA).

Основні кроки використання методу ARIMA для прогнозування витрат електроенергії:

- Аналіз часового ряду: починається з аналізу історичних даних щодо витрат електроенергії, включаючи вивчення трендів, сезонності, аномалій та інших характеристик часового ряду.
- Підготовка даних: Перед використанням ARIMA необхідно впевнитися, що часовий ряд стаціонарний, тобто його статистичні властивості, такі як середнє та дисперсія, залишаються сталими з часом. Якщо часовий ряд нестаціонарний, його можна перетворити на стаціонарний шляхом віднімання тренду, диференціювання тощо.
- Вибір параметрів моделі: ARIMA має три параметри: p, d, q, які відповідають порядку авторегресії (AR), ступеню диференціювання (I) та порядку ковзного середнього (MA). Вибір цих параметрів може виконуватися за допомогою аналізу автокореляційних та часткових автокореляційних функцій.
- Побудова моделі та її тренування: Після вибору параметрів створіть та навчіть модель ARIMA за допомогою історичних даних. Використовуйте ці дані для тренування моделі та оптимізації її параметрів.
- Оцінка моделі: Після тренування моделі оцініть її точність за допомогою тестових даних. Використовуйте метрики, такі як

середньоквадратична помилка (RMSE) або середня абсолютна помилка (MAE), для визначення точності прогнозів.

- Прогнозування майбутніх значень: Після успішної оцінки моделі використовуйте її для прогнозування майбутніх значень витрат електроенергії. Зазначте необхідну кількість майбутніх кроків та виведіть прогнозовані значення.

Таким чином, метод ARIMA може бути використаний для ефективного прогнозування витрат електроенергії на основі історичних даних та характеристик часового ряду.

Висновки до першого розділу

На основі даної інформації, можна зробити постановку задачі для дослідження, і основною темою буде керування системою Smart Grid.

Розглянуто вже існуючі методи керування, їхні недоліки та переваги, обрати метод який зможе покращити дану систему. Розглянути алгоритми та стратегії для ефективного розподілу електроенергії в мережі з урахуванням споживачів, виробників електроенергії та завнішніх факторів.

Проаналізувано систему керування споживанням для зменшення пікового навантаження, вирівнювання навантаження та зниження витрат енергії. Окрім цього потрібно дослідити механізми для моніторингу, діагностики мережі для забезпечення її стійкості та надійності.

Розглянуто існуючі методи захисту таких систем, адже вони повинні бути захищеними від кібератак.

2. ОСНОВНІ МЕТОДИ КЕРУВАННЯ СПОЖИВАННЯМ ЕЛЕКТРОЕНЕРГІЇ

2.1. Керування освітленням за допомогою дімерів

В цьому підрозділі буде розглянуто принципи та переваги керуванням освітленняи за допомогою дімерів у системі освітлення Smart Grid. Спочатку розглянемо принцип роботи дімерів, їхні характеристики та можливості використання.

Дімер – це пристрій, що дозволяє плавно або східчасто регулювати потужність, струм або напругу, яку використовую споживач, зменшуючи або збільшуючи яскравість лампи, температуру нагрівання різних пристроїв, таких як праска, паяльник, електроплит і так далі[15].

Розглянемо принципову схему дімера (рис.2.1.), яка виконана на симісторах.

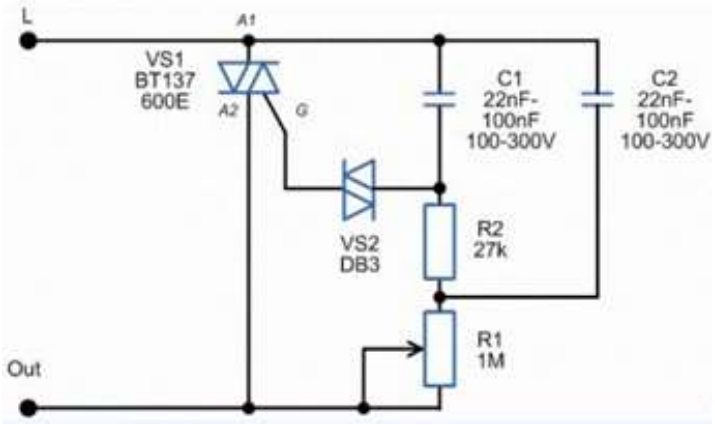


Рис.2.1. Принципова електрична схема дімера

Дані пристрої можуть виконуватись на симісторах, транзисторах або ж на тиристорах. Пристрій працює таким чином, що електронний ключ світлорегулятора не пропускає певну частину хвилі, внаслідок чого зменшується напруга, що йде на джрело. Джерело під час цього отримує синусоїду, яка обрізається, від чого падає яскравість світла.

Дімери класифікують наступним чином:

- аналогові;
- цифрофі;
- програмні.

Аналогові дімери можуть бути потинциометричні та фазові керуючі, і основною відмінністю що в першому використовується потенцометр, а в другому зміну фази вхідної напруги або струму.

Цифрові дімери використовують мікроконтролер для керування стумом або напругою. Дані пристрої мають більше можливостей програмування та контролю, такі як таймери, пам'ять налаштувань, збереження різних сценарціїв.

Програмні в свою чергу мають можливість керування за допомогою Wi-Fi мережі або навіть за допомогою рідіочастоти. Для таких типів дімерів є можливість віддаленого керування, моніторингу за допомогою мобільних або компютерних додатків.

Дімери можна використовувати разом з сесорами руху, відповідно при наближенні рухомого тіла, відбуватиметься освітлення, в залежності від наближення тіла до сенсору. Програмування таких пристроїв не потребує великої кількості ресурсів. Також можливе повязування дімерів разом з голосовими помічниками, серед яких Amazon Alexa, Google Assistant або Apple Siri.

Окрім цього основною метою дімерів є зменшення споживання електроенергії, таким чином зменшуючи яскравість на 25% можна зекономити близько 20% електроенергії, а якщо зменшувати яскравість десь на 50% в темну пору доби, то економія досягає цілий 35 - 40%. Використовуючи дану інформацію можна зробити висновок, що за допомогою керування освітлення дійсно можна менше використовувати електроенергії, зменшити навантаження на систему.

Далі будуть наведені розрахунки, які будуть підтверджувати теорію, яка викладена вище. Отже для того, щоб обчислити значення енергоефективності використання дімерів для освітлення, потрібно визначити енергоспоживання

лампочок на протязі одного місяця без дімера, а потім з дімером. На початку обислень потрібно визначити, наскільки довго за день лампочка може бути увімкнена. Взявши в розгляд лютий місяць, середня тривалість світлового дня складатиме близько 10 годин 15 хвилин, таким чином темна пора доби буде дорівнювати 13 годин 45 хвилин. Від цих годин потрібно відняти час сну, це 8 гоин, коли людина не використовує світло, таким чином отримуємо значення в 5 годин 45 хвилин. За формулою використання електроенергії (2.1.) можна розрахувати яку кількість енергії використовує led лампа:

$$Q = P \cdot t_{\square}, \tag{2.1.}$$

де Q – це споживання електроенергії(кВт · год); P – потужність пристрою; t – час використання пристрою.

Виходить так що 1 лампочка потужністю 15 Вт протягом одного дня використовує 0.0086 кВт · год, а за місяць 0.2415 кВт · год.

Тепер за допомогою дімера зменшимо потужність led лампи на 25%, таким чином споживання одної лампочки буде 0.0065 кВт · год, відповідно за місяць буде 0.182 кВт · год. При цих обрахункеах економія електроенергії складає 24.4%, тому викорситання дімерів дійсно може зменшити електроспоживання.

2.2. Керування різними електропристроями за допомогою сценаріїв

Для початку потрібно зрозуміти, чи використовує будинок електроенергію за відсутності людини у приміщенні. Відповідь на дане питання очевидна, адже взяти для прикладу звичайну квартиру, в якій може працювати холодильник, бойлер який завжди тримає воду нагрітою, можливо забуте вимкнене освітлення, забута увімкнена праска і так далі. Для різних приміщень є конкретні пристрої, які можуть працювати без людини, таким чином зявляється потреба в керуванні даними пристріями.

Керування електропристроями за допомогою сценаріїв це спосіб автоматизувати роботу пристроїв в приміщенні, створюючи передзадані

сценарії для різних ситуацій. Даний тип керування може бути використаний як для підвищення комфорту, так і для енергоефективності та безпеки[16].

Розглянемо декілька сценаріїв, для розуміння що ж можна робити за допомогою такого типу керування. Першим прикладом буде вечірнє освітлення. При заданні сценарію, освітлення буде автоматично переходити з більш яскравого до більш приємного та приглушеного світла, адже давно доведено, що яскравість освітлення впливає на велику кількість чинників, таких як ефективність роботи людини, відпочинок, сон і так далі. Другим прикладом будуть сценарій вихідних днів. Часто буває що у вихідні дні потрібно зробити дуже багато затньої роботи, але це також дні коли хочеться відпочити. За допомогою сценаріїв можна налаштувати увімкнення пральної машинки, увімкнення робота пилосмоку, або інших пристроїв. Третім сценарієм може бути довгострокова відсутність людини в приміщенні. За допомогою такого сценарію можна повністю заблокувати подачу електроенергії в приміщення, що підвищить безпеку у декілька разів за відсутності людини. Можна налаштувати момент прибуття людини і напередодні приходу нагріти воду у бойлері, прибрати квартиру, або навіть нагріти квартиру у випадку викорситання електричних батарей.

Для кращого розуміння енергоефективності при використанні сценаріїв, розглянемо приклад нагрівання води в бойлері. З 24 годин в добі, людина витрачає приблизно 8-10 годин на роботу, ще годину – дві на дорогу додому, що в сумі може досягати 12 годин відсутності в будинку. В цей час постійно увімкнений пристрій який споживає немалу кількість електроенергії. Для прикладу бойлер має потужність 2000 Вт, час роботи бойлера коли вода не використовується 12 годин. За одну годину такий бойлер використовує $2\ кВт \cdot год$ електроенергії. Відповідно за 12 годин використання сягатиме $24\ кВт \cdot год$. Але це занадто приблизне значення, так як пристрій не буде використовувати енергію в $2\ кВт \cdot год$, якщо вода буде постійно теплою. Тому потрібно дослідити яку ж кількість електроенергії використовує бойлер за 12 годин, і чи буде електроефективніше вимикати бойлер і вмикати його тільки тоді коли

потрібно. Отже потірбно розпочати з того, який час потрібен для того, щоб нагріти 80 літрів води до температури 75-80 ℃. Встановивши на розетку до якої підключений пристрій електрометр абюо електролічильник, можна виміряти середній час повного нагрівання води. Елктрометр це пристрій, який приєднується між розеткою(живленням) та електропристрієм(споживанням) (Рис.2.2.). Він вимірює кількість спожитої електроенергії в кіловат-годинах, а також може вимірювати потужність пристрою.

Рис.2.2. Потративний електролічильник

Для початку можемо розрахувати приблизний час нагрівання води. Використовуючи формулу для обчислення теплового підйому:

$$Q = mc\triangle T\,, \tag{2.2}$$

де Q – теплова енергія, що потрібна для нагрівання води; m – маса води; c – теплоємність води; ΔT – зміна температури води, можна визначити теплову енергію, яка потрібна для нагрівання.

Взявши масу води 80 кг, її теплоємність 4.186 кДж/кг℃ та зміну температури 55℃ (75℃ - 20℃ = 55℃), можна обчислити теплову енергію (2.2), яка буде дорівнювати 18418.4 кДж.

Далі використовуючи формулу для розрахунку часу:

$$t = \frac{Q}{P_{пристрою}}, \tag{2.3.}$$

де t – час нагрівання; Q – теплова енергія; $P_{\text{пристрою}}$ – потужність пристрою, можна обчислити час (2.3.), який буде дорівнювати 9 годин і 13 хвилин.

Наступним етапом буде вимірювання реального часу нагрівання, для порівняння розрахункових даних та практичних. При дослідженні було проведено 4 виміри часу нагрівання, які можна побачити в табл. 2.1.

Таблиця 2.1.

Таблиця нагрівання води в бойлері

Номер спроби	Початкова температура	Кінцева температура	Час нагрівання
1	21℃	75℃	4 год 21 хв
2	25℃	72℃	4 год 2 хв
3	22℃	74℃	4 год 20 хв
4	17℃	75℃	4 год 39 хв

З таблиці реальних значень можемо побачити різницю майже в два рази порівняно з теоретичними, так як не було враховано багато параметрів, такі як ефективність бойлера, ізоляція і так далі.

Отже, в середньому, бойлеру з характеристиками, які описані вище потрібно витрати в середньому 4 год 20 хвилин, а це 8.666 кВт · год енергії. Реальним же використання енергії з електрометру було біля 8 кВт · год.

Наступні вимірювання були проведені на 12-ти годинному відрізку, і на наступній таблиці можна побачити використання електроенергії бойлером за 12 годин, включаючи повне нагрівання та подальша підтримка температури.

Таблиця 2.2.

Таблиця нагрівання води в бойлері та підтримки її на рівні 75℃

Номер спроби	Час роботи	Використання електроенергії, кВт · год
1	12 год 3 хв	11.247
2	11 год 54 хв	11.196
3	12 год 15 хв	11.301
4	12 год 7 хв	11.284

Якщо проаналізувати отирмані результати, то можна зробити певні висновки, а саме, що під час підтримування 80-ти літрів води на рівні 75℃, електроспоживання буде приблизно 0.45 – 0.55 кВт · год. Відштовхуючись від цих даних, можна зробити висновок, що якщо людині потрібно часто гаряча вода, то бойлер можна не вимикати, а якщо це довготривала робота, або певні поїздки, то бойлером набагато краще керувати, так як буде можливість не використовувати електроенергії тоді коли вона не потрібна. Керування використовуючи сценарії забезпечить увімкнення таких пристроїв тільки при потребі, підвищуючи безпековий фактор, а також підвищить економію. Якщо брати до уваги що такий електропристрій, як бойлер має велике навантаження на мережу, і є досить популярним серед користувачів, то його керування є важливою складовою. Адже якщо будинок з 100 квартир буде одночасно використовувати такий електропристрій постійно, то навантаженя на мережу буде величезним, що може призводити до аварійних відключень.

Таких прикладів, можна наводити дуже багато, адже сценаріїв можна придумати скільки завгодно, під кожну ситуацію, систему можна підлаштувати для свого комфорту, тому керування системи цим способом, є важливою складовою Smart Grid.

2.3. Керування загальною потужністю споживання

Керування загальною потужністю споживання в системі Smart Grid є важливою складовою для забезпечення ефективності та стабільності

електромережі. Така система дозволяє зменшити використання електроенергії шляхом розподілу навантаження та зниження пікових значень споживання. Є багато методів, як саме можна зменшити ці параметри і для початку розглянемо основні з них.

Першим методом є метод розумного розподілу навантаження, який передбачає розподіл споживання електроенергії впродовж доби, особливо зсув споживання в електромережі на години, коли навантаження в мережі є меншим. Для прикладу розподілені системи енергоспоживання можуть давати можливість вмикати електроприбори, які мають велике навантаження на мережу, а саме посудомийні машини або пральні машини. Одним із методів такого розподілення є диференційовані тарифи, коли компанії можуть виставляти різні тарифи за електроенергію в залежності від годин використання. Такий спосіб розподілить споживання електроенергії не тільки на вечірній та ранішній час але і на нічний та обідній. Для цього потрібно розглянути графік, який пояснить, яка кількість користувачів і коли використовують електроенергію (рис. 2.3.)[17].

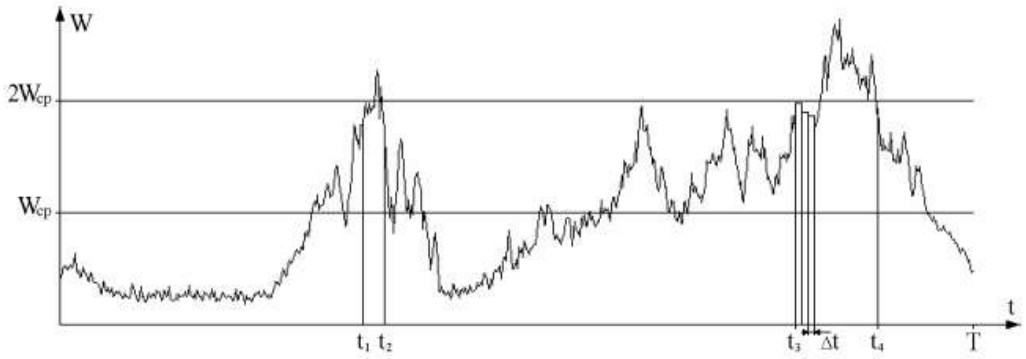


Рис.2.3. Графік енергонавантаження на мережу протягом дня

Дивлячись на графік споживання мона виділити три зони споживання, відповідно низка, середня та висока, які відповідають за навантаження в мережі. Період споживання $t_1 - t_2$ відповідає ранковому споживанню, а $t_3 - t_4$ вечірньому відпровідно. Диференційовані тарифи дозволять розподілити електроенергію таким чином, що зменшать використання

електроенергії в пікові години, розподіливши користувачів по графіку вліво або вправо. Це важливо тому, що при пікових навантаженнях система може спрацювати в аварійне завершення, що відключить користувачів від мережі, поки мережу не буде перезапущено. Бувають також випадки коли мережа не витримує пікових навантажень і стаються аварії на певних участках, що вже може призвести до набагато тривалішого відключення.

У випадку користування диференційованими системами, важливу роль відіграють автоматизовані системи управляння, за допомогою яких можна налаштувати роботу пристроїв в залежності від цінового тарифу на електроенергію. Крім цього багато пристроїв мають енергозберігаючі режими, внаслідок чого можна зменшити навантаженняі на мережу в пікові години, та перенести основний тип роботи на години коли мережа менше завантажена.

Другим методом розподілення є гнучке регулювання потужностей, яке передбачає використання різних систем автоматичного регулювання , які можуть динамічно змінювати потужність споживання електроенергіх в залежності від змін зовнішніх умов або ж потреб електромережі. Для прикладу система Smart Grid може отримувати інформацію про зміну цін на електроенергію в режимі реального часу. В залежності від цін, система може або збільшувати або зменшувати потужність електропристроїв, які підлючені до системи. Окрім автоматичного регулювання потужністю в залежності від цини, система Smart Grid може керувати пристроями в залежності від пікового навантаженняі в мережі, таким чином зменшуючи використання пристроїв, адже часто в мережі напруга може бути стрибкоподівною, і може бути небезпечно використовувати певні пристрої, які чутливі до стрибків напруги. Також гарним доповненням є використання резервних потужностей, за допомогою яких можна використовувати електропристрої навіть у пікові години. За допомогою резервних потужностей, основна мережа не буде отримувати від користувача Smart Grid навантаження, але електроприклади можуть працювати на повну потужність. Завдяки цим властивостям можна

зменшити кількість аварійних випадків, а також продовжити експлуатаційний термін як мережі так і електропристрої споживачів.

Третім методом є керування за допомогою прогнозування, яке передбачає використання разіних прогнозних моделей, для передбачення майбутнього споживання електроенергії та управління потужністю відповідно до цих прогнозів. Основним завданням даного методу є збереження та обробка великої кількості данних, включаючи історичні данні про споживання в певний період, погожні умови, час есплуатації різних підсистем і так далі. На основі цих даних розробляється прогнозована модель, яка може передбачити майбутнє споживання електроенергії. На цьому етапі використовується статистичні методи, машинне навчання, нейронні мережі і тому подібне. Після завершення прогнозування система може моніторити споживання електроенергії та порівнювати значення з прогнозованими і визначати який відсоток прогозу справдився і підлаштовувати систему під ці дані.

Четвертим методом, не менш важливим, є керування за допомогою користувацьких втручань в систему Smart Grid. Зазвичай в даному методі використовуються різні додатки для користувачів, зі зрозумілим інтерфейсом, щоб кожен міг зрозуміти та допомогти налаштувати систему під свої потреби. Зазвичай користувачі мають доступ до таких данихї як моніторинг та аналіз споживання, а також історичні дані про споживання в різний період часу. Окрім цього в таких інтерфейсах передбачений вибір користувачем режимів роботи системи, відповідно до свого розпорядку дня, так як не всі роботи відбуваються під час денного часу. В таких випадках користувач може мати доступ до віддаленого керування власними пристроями, та визначати коли вони повинні відпрцювати. Також додаток може надавати певні рекомендації та поради для людей на основі історичних спосережень за іншими користувачами, їхніми графіками роботи, періодамии використання пристроїв і так далі.

2.4. Використання нейронних мереж та машинного навчання в керуванні Smart Grid

Використання нейронних мереж та машинного навчання в рази збільшує можливості керування та прогнозування системи, забезпечує сталість системи та підвищує ефективність роботи.

Прогнозування енерговитрат за допомогою нейронних мереж – це процес розвитку моделі, яка використовується для передбачання майбутніх витрат, на основі різних вхідних даних[18]. Для цього потрібно зібрати дані про витрати за різні періоди часу, дані про погоду, певні олсоблипості приміщення, кількість людей, робочі години або ж час коли пристуні люди в приміщенні. Після цього етапу потрібно провести обробку даних, включаюси їхнє масштабування, нормалізацію. До вибору типу нейронної мережі підходять відносно завдань для цієї системи, це може бути наприклад feedforward, рекурентна, зворотня, також треба обрати розмірність шарів. Далі іде основна робота, а саме навчання нейронної мережі на основі зібраних даних, налаштування параметрів моделі, збільшення чи зменшення кількості нейронів в кожному шарі, швидкість навчання і тому подібне. Також важливою частиною створення прогнозування на базі нейронних мереж, є оцінка ефективності моделі на даних. Найчастіше використовуються такі метрики оцінки, як середньоквадратична помилка або RMSE, або коефіцієн детермінації (R^2) для оцінки точності створеної моделі. Після всього цього можна доповняти вже нові дані в реальному часі, і можель зможе прогнозувати паарметри які були задані. Такі мережі можуть використовуватись також для прогнорзування погоди, що дуже важливо для генерації електроенергії з відновлювальних джерел енергії. Прогнозування захмарености неба для соняних панелей, швидкість вітру для вітряків і так далі.

Керування енергоефективністю також можливе при використанні нейронних мереж абюо машинного навчання. Машинне навчання може бути використане для оптимізації роботи генеруючих станцій з врахуванням

багатьох вимог до виробництва електроенергії. Для прикладу, алгоритми машинного навчання можуть оптимізувати розподіл навантаженнями між різними генеруючими джерелами, в залежності від їхніх технічних характеристик та економічної вимоги. Нейронні мережі в свою чергу можуть доповнювати машинне навчання, адже можуть аналізувати дані про стан обладнання, навантаження в мережі, погодні умови після чого приймати рішеня для оптимального керування розподілом електроенергії.

Для прикладу команія Google DeepMind впроваджує нейронні мережі для прогнозування електроенергії. Це дозволяє компанії ефективно керувати електроспоживанням та істотно знижувати витрати електроенергії. В той же час італійська компанія Enel викорситовує нейронні мережі для прогнозування попиту на електроенергію та відносно цих прогнозів обирає стратегію генерації електроенергії. Tesla використовує дані методи для розробки системи управління споживанняи електроенергії у своїх системах зберігання енергії PowerWall (рис.2.4.).

Рис.2.4. Система зберігання енергії Power Wall з контролером

Такий спосіб дозволяє користувачам ефективно використовувати збережену енергію та знижувати витрати на електроенергію. В свою чергу

General Electric використовують нейронні мережі для аналізу даних з різних датчиків та пристроїв для виявлення проблем та способів запобігання аварійних ситуацій.

Як видно з прикладу всіх цих компаній, керування за допомогою додаткових методів, може не тільки знизити електровикорситання, але й визначати проблему в мережі, а в деяких випадках і повністю запобігати аваріям. Тому важливо розглядати керування системою Smart Grid за допомогою машинного навчання та нейронних мереж.

Завдяки прогнозуванню та керуванню на базі нейронних мереж можна економити біля 12 – 15% електроенергії в містах. Використання машиного навчання в системах розподілу енергії може знизити час відновлення енергопостачання після аварій на 20 – 25%. Зношеність пристроїв та компонентів знижується на 15 – 20% завдяки машинному навчанню.

Висновки до другого розділу

В даному розділі розглянуто можливість керування використанням електроенергії в мережі за допомогою сценаріїв, таких пристроїв як дімери, досліджено методи розподілення навантаження в мережі протягом дня. Наведені приклади економії при використанні методів керування використання електроенергії.

Розглянуто метод керування системою Smart Grid за допомогою нейронної мережі та машинного навчання в загальному.

3. РОЗРОБКА АЛГОРИТМУ ВІДСТЕЖЕННЯ ТА ПРОГНОЗУВАННЯ ВИКОРИСТАННЯ ЕЛЕКТРОЕНЕРГІЇ

3.1. Наповнення бази даних та їхня обробка

Для початку потрібно визначити які саме дані потрібно зібрати в ході дослідження. Є різні варіанти, починаючи від використання електроенергії освітленням, закінчуючи такимим приладами як варильні поверхіні, бойлери і так далі.

Для даної роботи вдалось зібрати данні використання електроенергії одного з ресторанів, що включає в себе кондиціонери великої потужності, морозильні камери, холодильники, бойлери, конвеєрні печі та фретюрниці.

Для початку потрібно дані перенести до бази даних, яка буде використовуватись в подальшій розробці алгоритму, і відповідно до завдань потрібно вибрати тип бази даних. Більшість баз даних поділяють на реляційні та нереляційні, реляційними називають бази даних які засновані на жорсткій структуризації та типізації відомостей про обєкти, а нереляційні бази зберігають дані без чітко видимих звязків між собою, та чікої структури.

Найпростіший вид реляційної бази даних це таблиці з рядків та стовпців, де в рядках наводяться відомості про обєкт, а в стовпцях – властивості обєктів. Використання реляційних баз даних дозволяє мінімізувати обсяг бази даних, підвищити цілісність системи, спростити масштабування, підвищити стійкість від відмов. Запити до даних типів баз даних виконуються за допомогою структурованої мови SQL [19]. Вона дозволяє робити вибірки, проводити різні маніпуляції, такі як ігрегацію, угруповування, зміна та видалення даних, керувати доступом користувачів і так далі.

Прикладом реляційних баз є:

- MySQL;
- MariaDB;
- PostgreSQL;

- SQLite.

Щодо нереляційної бази даних, то найпростішим проявом є так званий словник, коли дані зберігаються за принципом «ключ-значення», депокажчиком є ключ. В таких базах можна зберігати та обробляти різні типи даних, що дозволяє щоб в одному сховищі під різними ключами були данні не тільки в текстовому форматі, але і в числами, файлами, JSON обєктами та іншими типами даних. Такі бази даних відносно легко масштабувати, тому вони дуже часто використовуються розробниками.

Прикладом нереляційних баз за принципом «ключ-значення» є наступні БД:

- DynamoDb;
- Redis;
- Riak;
- LevelDb.

Відповідно до задач дипломного дослідження в роботі буде використовуватись база даних MySQL. Така база даних дозволяє підтримувати необмежену кількість користувачів, які одночасно працюватимуть за базою, вона проста у використанні адже є Workbench з власним інтерфейсом, кількість рядків в таблицях може досягати 50 млн, висока швидкість виконання команд, наявна проста і ефективна система безпеки.

Для початку створюється база даних через MySQL Workbench [20], що на початку можливо тільки локально. На рис.3.1. продемонстровано інтерфейс створення БД (бази даних).

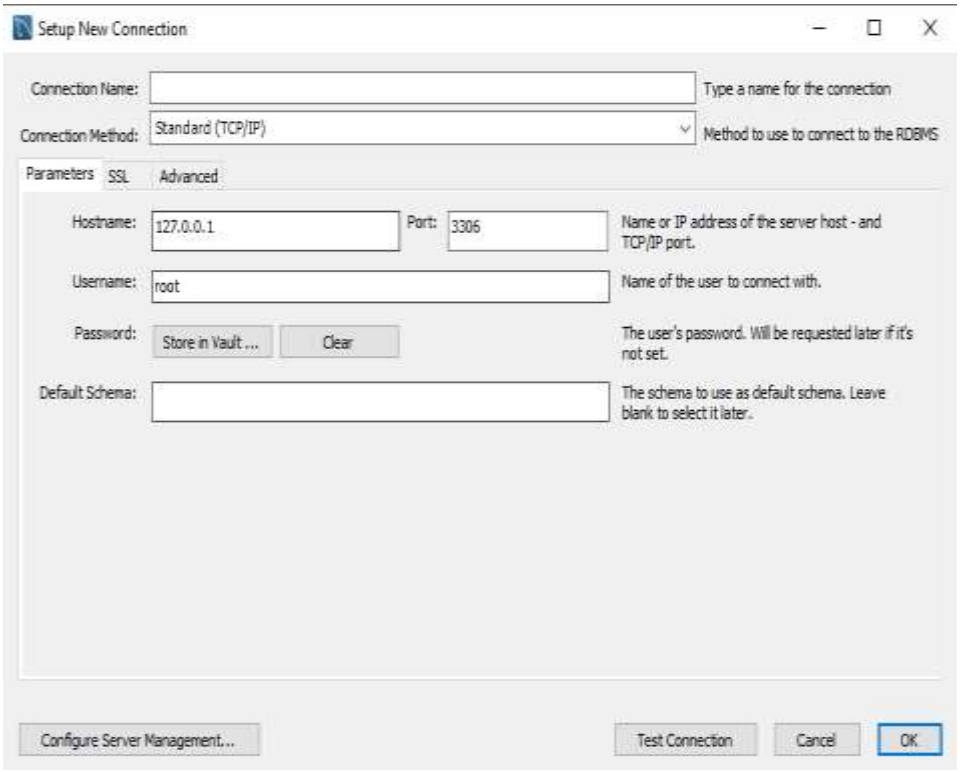


Рис.3.1. Створення БД за допомогою MySQL Workbench

Після створення БД, ми можемо повністю взаємодіяти з нею, створювати таблиці, повязувати їх між собою, наповнювати даними і так далі. Першим завданям майбутнього алгоритму буде передбачення використання електроенергії освітленням, використовуючи різні типи ламп, а саме лампи накалюваннння, флюорисцентні, LED та галогенові. На початку було створено одну таблицю month, яка була наповнена стовпчиками idmonth, яка буде мати рядки від 1 до 12 відповідно до місяців та light_time, який буде містити в собі інформацію про світловий день на кожен день відповідно до місяця. В MySQL Workbench це буде виглядати так як на рис.3.2.

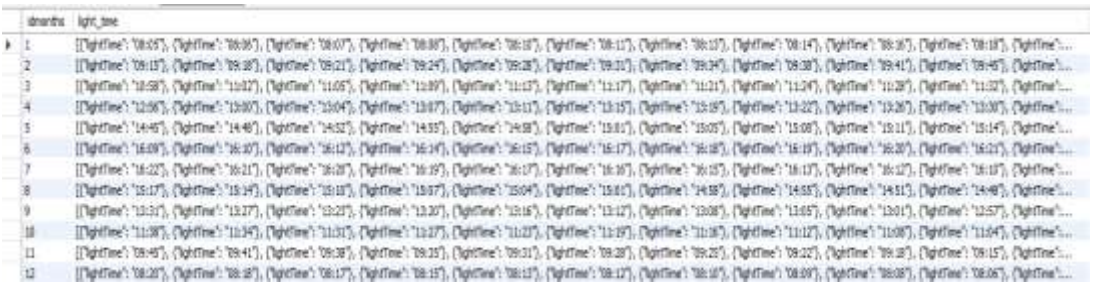


Рис.3.2. Таблиця “month”

Наступним етапом буде заповнення даних які вдалось зібрати з ресторану, для цього для кожного типу приладу була створена відповідна табиця з даними, які потрібні для подальшої роботи. На табл. 3.1. наведений повний перелік пристроїв, які використовуються в даній роботі.

Таблиця 3.1.

Пристрої та їх характеристики

Назва	Кількість	Потужність (кВт)
CLIMA LOBBY (Air/air heat pump HITECSA RXCBZ-2002)	1	13.3
CLIMA KITCHEN (Air/air heat pump HITECSA RMXCA-1402.2)	1	18
FREEZING (RIVACOLD RC325-33ED)	1	3
CHILLING (RIVACOLD RC225-25ED)	1	2.5
NIECO AUTOMATIC NIECO-JF74-ELECTRIC	1	5
FRYMASTER (OCF30™ ELECTRIC FRYER FPEL414CA)	4	0.5

HITECSA RXCBZ-2002 та HITECSA RMXCA-1402.2 – це кондиціонери які встановлюється зазвичай з виходом на вулицю [21], RIVACOLD RC325-33ED та RIVACOLD RC225-25ED – це охолоджувачі повітря [22], які встановлюються для створення холодильних приміщень. NIECO-JF74-ELECTRIC – це подвійний стрічковий бройлер [23], і фретюр OCF30™ ELECTRIC FRYER FPEL414CA [24].

Під час дослідження було виміряні такі параметри як температура, вологість, потужність, напругу, витрату потужності, струм для кожного з приладів. Для подальшої роботи такі параметри як температура та вологість не потрібні для розразхункув, тому їх не буде враховано.

Для початку потрібно всі зібрані дані перемістити в базу MySQL, для цього існує такий інструмент як TDIW (Table Data Import Wizard), за допомогою якого можна імпортувати таблиці з формату Excel таблиць. На рисунку 3.3. зображено процес додавання виміряних ресультатів до БД.

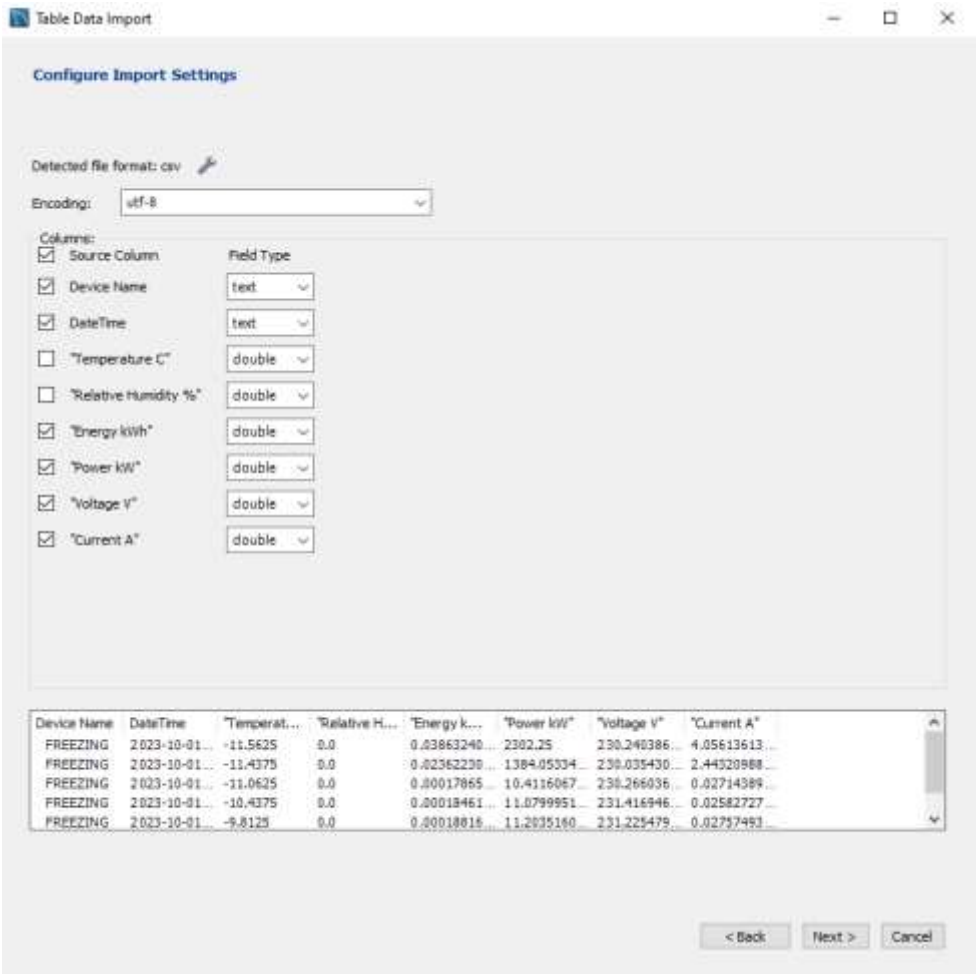
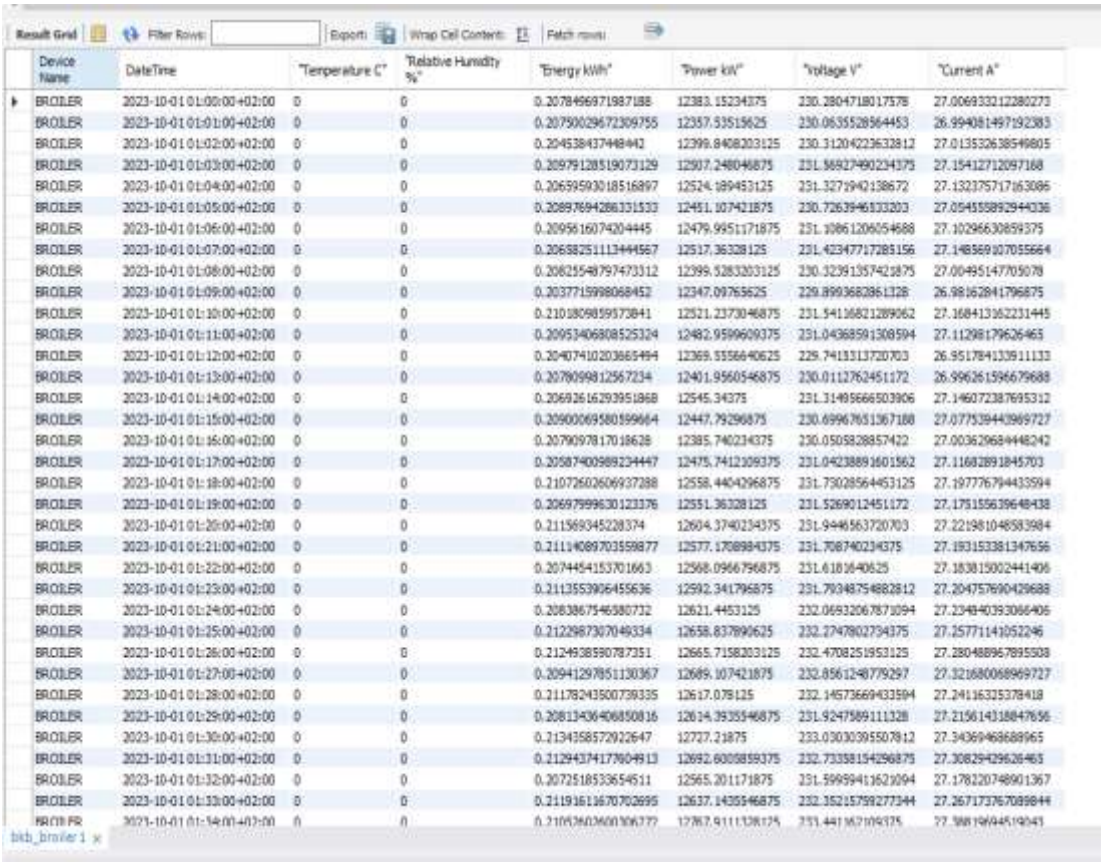


Рис.3.3. Конвертація Excel таблиці в MySQL за допомогою TDIW

При конвертації є можливість змінити тип даних який буде відображатись в таблицях, а також обрати метод енкодування. Окрім конвертування потрібно додати такий параметр до таблиць як primary_key, для швидкого пошуку

значень по таблиці, і легкої обробки даних. Після конвертування отримаємо таблицю, яка зображена на рис.3.4.



Device Name	DateTime	"Temperature C"	"Relative Humidity %"	"Energy kWh"	"Power kW"	"Voltage V"	"Current A"
BROILER	2023-10-01 01:00:00+02:00	0	0	0.2078496971987188	12383.15234375	230.2804718017578	27.006933212280273
BROILER	2023-10-01 01:01:00+02:00	0	0	0.20790029672309755	12357.53515625	230.0635528564453	26.994081497192383
BROILER	2023-10-01 01:02:00+02:00	0	0	0.20453843748442	12399.8408203125	230.31204223632812	27.013532638549805
BROILER	2023-10-01 01:03:00+02:00	0	0	0.20979128519073129	12907.248046875	231.56927490234375	27.15412712097168
BROILER	2023-10-01 01:04:00+02:00	0	0	0.20659930185168967	12524.189453125	231.3271942138672	27.132375717163086
BROILER	2023-10-01 01:05:00+02:00	0	0	0.20897694286331533	12451.107421875	230.7263946533203	27.094555892944336
BROILER	2023-10-01 01:06:00+02:00	0	0	0.209561607420445	12479.9951171875	231.1086120605468	27.10296630859375
BROILER	2023-10-01 01:07:00+02:00	0	0	0.20658251113444567	12517.36328125	231.42347717285156	27.148569107055664
BROILER	2023-10-01 01:08:00+02:00	0	0	0.20825548797473312	12399.5283203125	230.32391357421875	27.00495147705078
BROILER	2023-10-01 01:09:00+02:00	0	0	0.2037715998068452	12347.09765625	229.8993682861328	26.981628417968754
BROILER	2023-10-01 01:10:00+02:00	0	0	0.2101808959573841	12521.2373046875	231.54116821289062	27.168413162231445
BROILER	2023-10-01 01:11:00+02:00	0	0	0.20953406808525324	12482.9599609375	231.04368591308594	27.112981796264648
BROILER	2023-10-01 01:12:00+02:00	0	0	0.20407410203665494	12369.5556640625	229.74131317520703	26.951784133911133
BROILER	2023-10-01 01:13:00+02:00	0	0	0.20780998125672344	12401.9560546875	230.0112762451172	26.996261596679688
BROILER	2023-10-01 01:14:00+02:00	0	0	0.20692616293951868	12545.34375	231.31495666503906	27.146072387695312
BROILER	2023-10-01 01:15:00+02:00	0	0	0.20900069580599664	12447.79296875	230.69967651367188	27.077539443969727
BROILER	2023-10-01 01:16:00+02:00	0	0	0.20790978170186285	12385.740234375	230.0505828857422	27.003629684448242
BROILER	2023-10-01 01:17:00+02:00	0	0	0.20587400988234447	12475.7412109375	231.04238891601562	27.116828918457033
BROILER	2023-10-01 01:18:00+02:00	0	0	0.21072602606937288	12558.440429687	231.73028564453125	27.197767944335938
BROILER	2023-10-01 01:19:00+02:00	0	0	0.20697999630123376	12551.36328125	231.5269012451172	27.175155639648438
BROILER	2023-10-01 01:20:00+02:00	0	0	0.211569345228374	12604.3740234375	231.9446563720703	27.221981048583984
BROILER	2023-10-01 01:21:00+02:00	0	0	0.21114089703559877	12577.1708984375	231.708740234375	27.193153381347656
BROILER	2023-10-01 01:22:00+02:00	0	0	0.2074454153701663	12568.0966796875	231.618164062 5	27.183815002441406
BROILER	2023-10-01 01:23:00+02:00	0	0	0.211355390645563 6	12592.341796875	231.79348754882812	27.204757690429688
BROILER	2023-10-01 01:24:00+02:00	0	0	0.20838675465807 32	12621.4453125	232.06932067871094	27.234840393066406
BROILER	2023-10-01 01:25:00+02:00	0	0	0.21229873070493 34	12658.837890625	232.2747802734375	27.25771141052246
BROILER	2023-10-01 01:26:00+02:00	0	0	0.21249385907873 51	12665.7158203125	232.4708251953125	27.280488967895508
BROILER	2023-10-01 01:27:00+02:00	0	0	0.20941297851130367	12689.107421875	232.8561248779297	27.321680068969727
BROILER	2023-10-01 01:28:00+02:00	0	0	0.21178243500739335	12617.078125	232.14573669433594	27.24116325378418
BROILER	2023-10-01 01:29:00+02:00	0	0	0.20813436406850816	12614.3935546875	231.9247589111328	27.215614318847656
BROILER	2023-10-01 01:30:00+02:00	0	0	0.2134358572022647	12727.21875	233.03030395507812	27.34369468688965
BROILER	2023-10-01 01:31:00+02:00	0	0	0.21294374177604913	12692.6005859375	232.73358154296875	27.308294296264648
BROILER	2023-10-01 01:32:00+02:00	0	0	0.207251853365451 1	12565.201171875	231.5999941162109 4	27.178220748901367
BROILER	2023-10-01 01:33:00+02:00	0	0	0.21191611670702695	12637.1435546875	232.35215759277344	27.267173767089844
BROILER	2023-10-01 01:34:00+02:00	0	0	0.21063630600306777	12767.911132812 5	233.44196224 3125	27.388196945190 43

Рис.3.4. Виміряні дані з NIECO-JF74-ELECTRIC в БД

Кроки описані вище потрібно зробити для всіх виміряних даних, створити для всіх приладів таблиці з даних після чого можна буде працювати з ними. В даній роботі використано технологію JDBC (Java Database Connectivity) для оперування даними в таблиці. JDBC – це прикладний програмний інтерфейс Java, який визначає методи, з допомогою яких програмне забезпечення на Java здійснює доступ до бази даних [25]. Головними перевагами є легкість в розробці, якщо перейти на іншу базу даних, то код не зміниться, так як JDBC використовує мову SQL.

3.2. Побудова архітектури алгоритму

3.2.1. Створення інтерфейсу Android додатку

Для побудови інтерфейсу використовується інтрефейс розробника Android Studio, та такі мови програмування як Java та XML. Серед технологій які використовуються для побудови інтерфейсу та первинної логіки програми використовуються авторизація за допомогою Google Firebase, логування написаного коду використовуючи бібліотеку Lombok, для тестування використовується Junit, MaterialDesign для побудови інтерфейсу, а також OkHttp бібліотеку для побудови запитів на сервер.

Спочатку було розроблено інтерфейс для обчислення приблизного використання електроенергії використовуючи освітлення. Додано 4 типи ламп з можливістю ввести їхню кількість, а також обрати потужність лампи (рис.3.5).

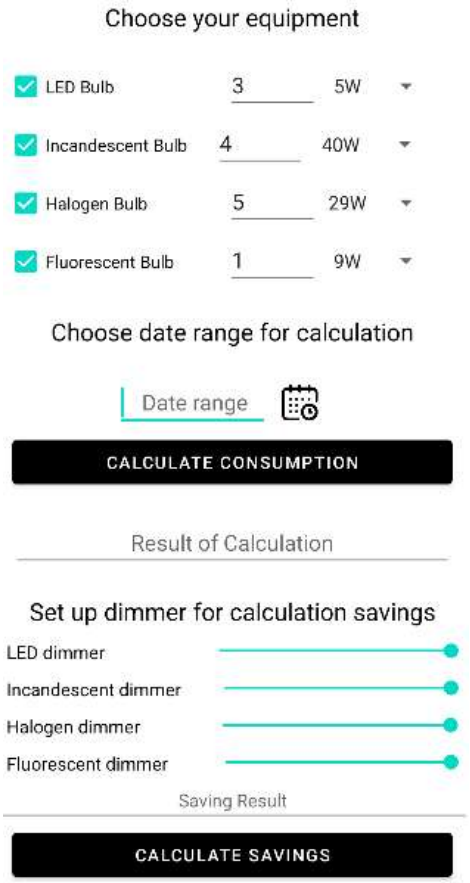


Рис.3.5. Інтерфейс користувача

Для того щоб обрахувати приблизне використання електроенергії, потрібен період використання ламп, але так як кожного місяця навіть кожного

дня темна пора доби різна, то потрібні дані скільки ж годин в день може використовуватись освітлення. Для цього було створено можливість за допомогою бібліотеки MaterialDatePicker обирати діапазон дати, за яку потрібно вирахувати використання електроенергії обраними лампами. Після натискання на кнопку “Calculate result” додаток за допомогою OkHttp бібліотеки звертається до серверу з відповідними даними, які передаються, і повертається результат обчислень.

Розділ додавання дімерів для різних типів ламп додано для того щоб наглядно побачити наскільки сильно можна заощадити, якщо використовувати один зі способів керування освітленням. Біля кожного типу ламп додано елемент інтерфейсу, скроллер, який віддалено імітує дімер. По суті даний елемент схожий на потенціометр, за допомогою якого працює дімер. Аналогічно до обчислення використання електроенергії, обчислення заощедження виконується на серверній частині додатку, яка взаємодіє з БД. Весь код даного інтерфейсу продемонстровано у Додатку 1, де можна детально з ним ознайомитись.

3.2.2. Створення серверної частини

Серверна частина додатку – це код який, який працює або локально або віддалено, і виконує різну обчислювальну частину. Для побудови даної частини було використано фреймфорк Spring [26] за допомогою мови програмування Java.

Для початку потрібно створити проект в середовищі IntelliJ IDE Ultimate з усіма конфігураціями для Spring проекту. Основними налаштуваннями є вибір версії Spring Boot, вибір “білдера” для додатку, мову програмуваня. Далі потрібно обрати тип пакування, jar файл або war. Після чого версію мови програмування. Створення проекту можна зробити не тільки в середовищі розробки, але також і з веб сторінки і обрати всі налаштування для подальшої роботи (рис.3.6).

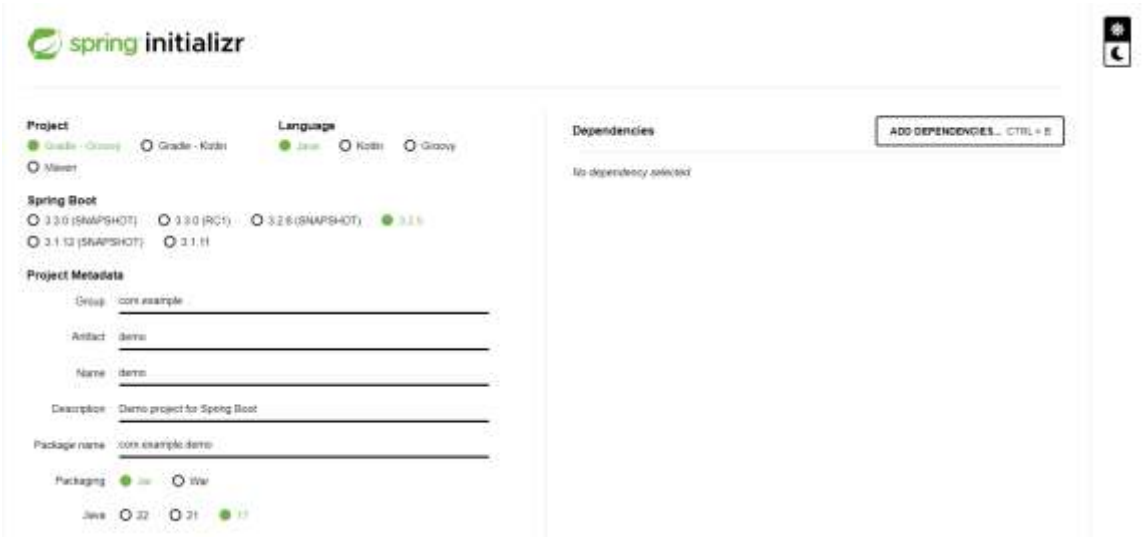


Рис.3.6. Створення Spring Application

Після усіх налаштувань веб версія або середовище розробки автоматично створить первинні налаштування які потірбні будуть для подальшої роботи. Є суттєва відмінність між простим написанням Java коду, та написанням коду за допомогою Spring, адже основною відміннісю є велика кількість анотацій, що спрощує велику кількість робочих моментів з Java.

На серверній частині додатку спочатку потрібно звязатись з БД в якій збережені дані про тривалість світлового дня, а також реальні дані з обєкту дослідження. Це робиться за допомогою бібліотеки java.sql, які потрібні такі параметри, як url за яким знаходиться база даних, user та pass відповідно логін та пароль з якими відбуватиметься підключення до БД. Окрім цього середовище розробки надає доволі зручну функцію, як підключення до БД, використовуючи плагіни, де можна протестувати підключення, взяти дані, написати SQL команди та протестувати їх на якомусь етапі роботи. Так як піднімати реальний сервер є не дуже доцільно на етапі розробки та тесування, вся робота була виконана локально.

На даному рисунку можна побачити як відбувається підключення до БД та плагін підключення в середовищі розробки(рис.3.7.).

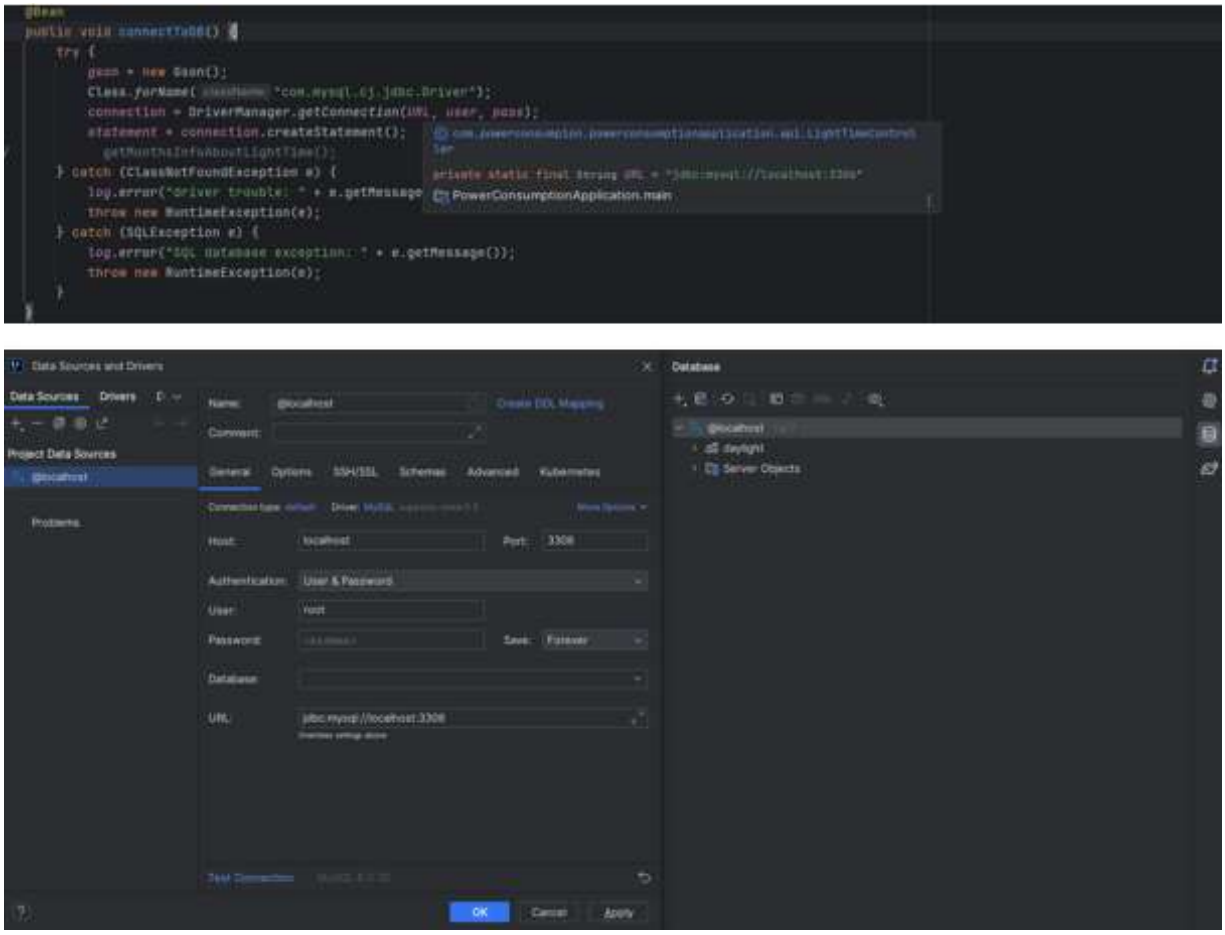


Рис.3.7. Підключення до БД

Далі потрібно продумати основний алгоритм роботи та правильну взаємодію між клієнтом та сервером. Так як раніше було визначено що клієнт буде робити запити до серверу, то сервер повинен приймати ці запити та обробляти, після чого віддавати віповідь. Найпростішим варіантом буде побудувати REST API (Representational State Transfer Application Programming Interface) на серверній стороні, і працювати з ним. REST API – це це інтерфейс прикладного програмування, який відповідає принципам проектування архітектурного стилю передачі стану [27]. Завдяки REST API, сервер зможе приймати HTTP(HyperText Transfer Protocol) запити, і давати відповідь клієнту по тому ж підключенню.

Для початку потірбно створити клас який матиме в собі назву Controller, в якому на рівні класу додати анотацію @RestController, після чого в тілі класу буде можливість анотувати методи класу відповідно до завдань. Основними

HTTP запитами, є GET та POST. Відповідно GET це запит на отримання інформації, POST – надсилання інформації на сервер. На наступному рисунку показаний принцип “клієнт-сервер” архітектури (рис.3.8.).

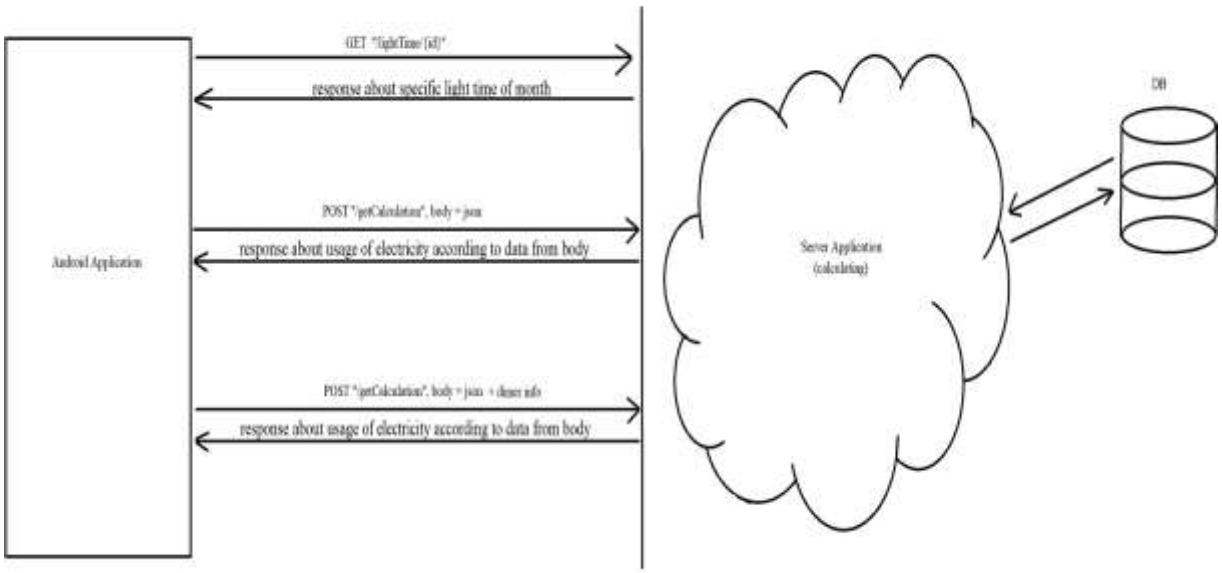


Рис.3.8. Принцип роботи “клієнт-сервер” архітектури

Android Додаток надсилає HTTP запит на сервер з відповідними параметрами, наприклад метод запиту POST, з кінцевою точкою “/getCalculation”. Сервер приймає даний запит і починається обробка даних які прийшли в тілі запиту. Для прикладу з клієнтської частини прийшла інформація, що потрібно зпрогнозувати яку кількість електроенергії буде використовувати 2 LED лампочки, 1 лампа накалювання та 2 флюоресцентені. Також було надано параметри потужності цих ламп і період за який треба спрогнозувати використання електроенергії. Сервер робить запит до БД на отримання інформації про тривалість світлового дня для цілого місяця. Після чого відбувається обробка даних, адже частіше за все лампи використовуються використовуються в темну пору доби, тову відбувається розрахунок темної пори доби для кожного дня в місяці. Ці дані перенаправляються на наступний етап, а саме фільтр по додатковим параметрам, як наприклад час сну. Від часу темної пори доби віднімається час сну, приблизно 7-8 годин, коли освітлення не

використовується. В кінцевому етапі відбувається прорахунок кількості використаної енергії в залежності від кількості ламп, їхнього типу, потужності і повертається результат клієнту. Розрахунок використаної енергії для окремого типу лампи відбувається за наступною формулою:

$$E_u = h \cdot \sum_{i=1}^{n} P \cdot n, \quad (3.1.)$$

де, $E_u -$ використана енергія в кВт-год, $h -$ час протягом якого використовуються лампи, $\sum_{i=1}^{n} P \cdot n -$ сума номінальної потужності лампи, $n -$ кількість ламп.

Повертаючись до прикладу розрахуємо використану електроенергію за один день (3.1.). Час використання лампи для 5ого квітня буде дорівнювати, часу темної пори доби відняти час сну, тобто h = 10 год 49 хв – 7 год = 3 год 49 хв.

Нехай LED лампи мають потужність 15 Вт, тоді використання енергії за один день буде $E_u =$ 3 год 49 хв·(15Вт + 15Вт) = 114.5 Вт-год, або 0.00145 кВт-год.

За таким самим алгоритмом відбувається прогнозування використання електроенергії для декількох місяців наперед.

3.2.3. Результати роботи алгоритму

Для прикладу розглянемо використання 2 LED ламп потужністю 5 Вт, 3 лампи накалу потужністю 40 Вт, 4 галогенні лампи потужністю 29 Вт та 5 флюоресцентних ламп потужністю 9 Вт протягом 3ьох місяців (11.2024 – 01.2024). На рисунку 3.9 можна побачити результат виконання програми.

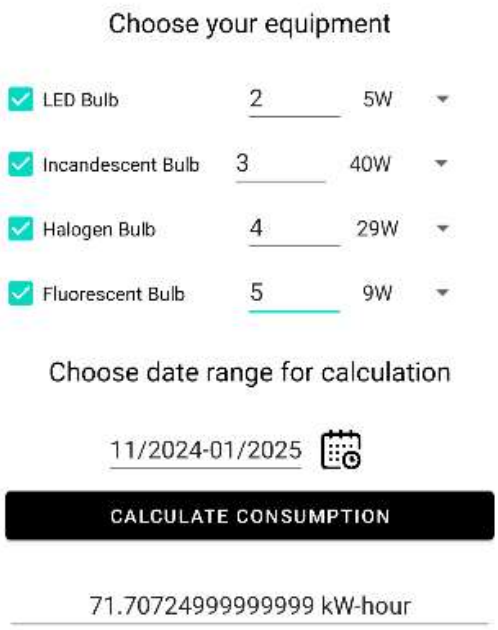


Рис.3.9. Результат прогнозування використання електроенергії освітленням за 11.2024 – 01.2024

Бачимо, що за 3 місяці приблизне використання електроенергії складатиме 71.7 кВт-год. Для наступного порівняльного прикладу візьмемо також три місяці, коли темна пора доби менша, ніж в попередньому прикладі (рис.3.10.).

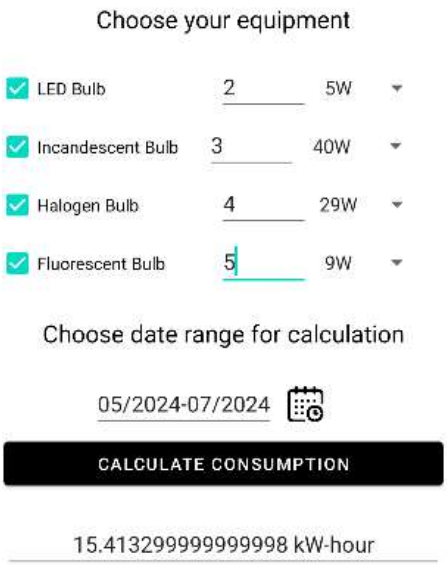


Рис.3.10. Результат прогнозування використання електроенергії освітленням за 05.2024 – 07.2024

Порівнюючи дані рисунки можна помітити велику різницю у використанні електроенергії за місяці коли світловий день великий, і навпаки малий. В результаті цього постає задача економії, і на рис. 3.11. продемонстровані результати економії електроенергії, якщо використовувати дімери, які налаштувані на зменшення яскравості на 25%.

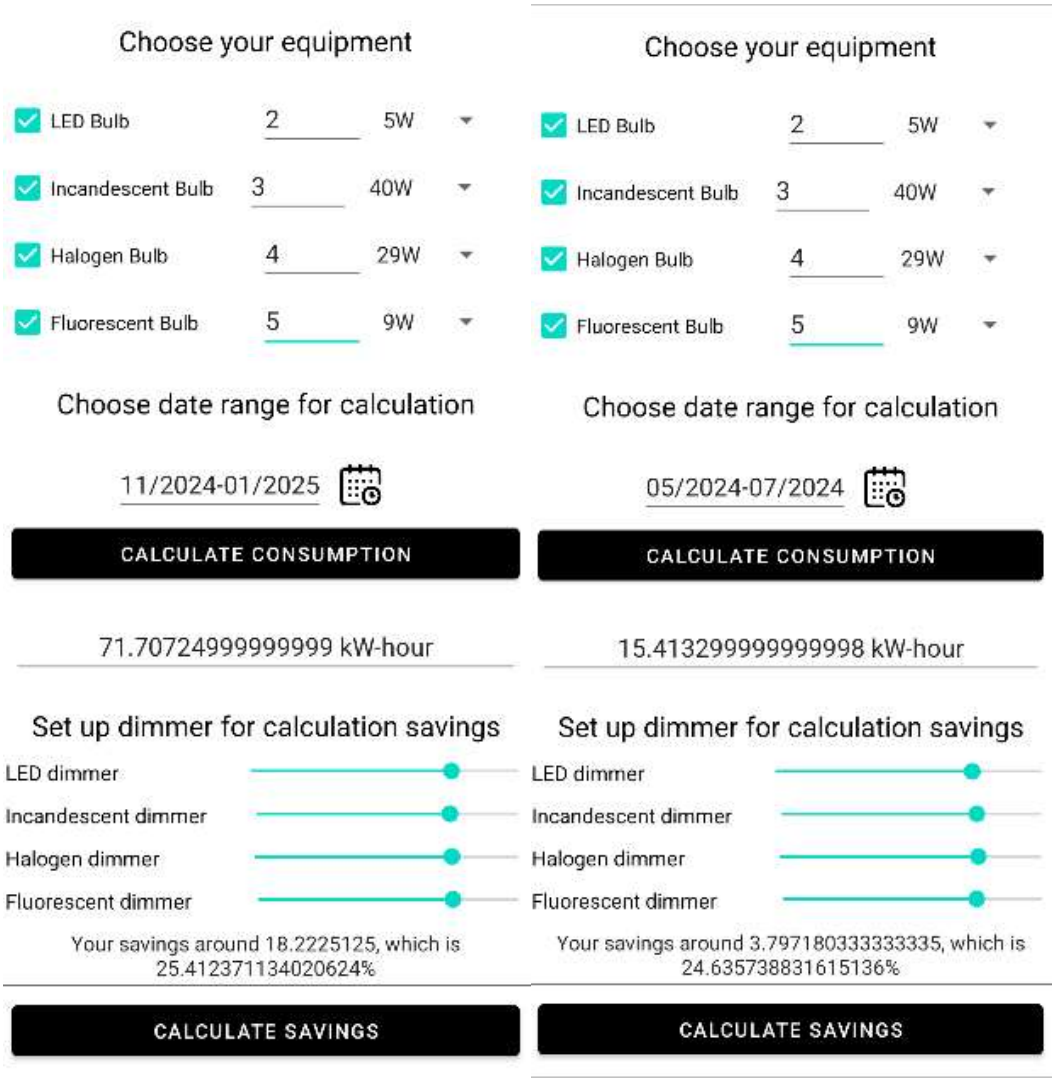


Рис.3.11. Порівняння економії використання електроенергії використовуючи дімери

Завдяки цим прогнозам можна визначати яка кількість електроенергії може бути використана в муйбутніх місяцях, і яку кількість можна заощадити використовуючи такий пристрій як дімер.

3.3. Використання нейронної мережі для прогнозування витрат

3.3.1. Нормалізація даних

Нормалізація зібраних даних це важливий крок для навчання нейронної мережі. В мові програмування Java можна використовувати бібліотеку DeepLearning4j, яка надає багато інструментів для нормалізації даних і подальшої роботи з ними. Нормалізація даних – це процес масштабування вхідних значень до певного діапазону, зазвичай це 0 та 1, або -1 та 1. Даний процес допомогає прискорити процес навчання та підвищити стабільність мережі [28].

Основними перевагами навчання моделі за нормалізованими данними є такі параметри, як швидкість навчання, стабільність навчання та покращення продуктивності моделі.

Нормалізацію розподіляють за типами:

- min-max нормалізація;
- z-score нормалазація;
- нормалізація на основі відсотилів.

Нормалізація min-max означає саме масштабування значень в діапазоні від 0 до 1. Z-score нормалазація – це масштабування значень значень таким чином, щоб вони мали середнє значення 0 та стандартне відхилення 1. І нормалізація на основі відсотилів, виконується масштабуваням даних з використанням медіани та міжквартильного діапазону.

В бібліотеці DL4J(DeepLearning4j) використовуються класи NormalizerMinMaxScaler та NormilizerStandardize. На наступному прикладі приведено як саме працює нормалізація min-max.

Беруться виміряні дані, де перший стовпець це дата та час проведеного заміру витрати електроенергії, а другий стовпець це витрата електроенергії. Для початку потрібно привести дату та час до якогось стандартизованого типу значень, і таким є UnixTime. Unix Time визначається як кількість секунд, які

пройшли з півночі 1-ого січня 1970 року. Саме 01.01.1970 рік вважають початком епохи Unix. На табл. 3.2. представлений приклад виміряних даних.

Таблиця 3.2.

Дані про використання електроенергії

Дата і час	Витрата електроенергії, кВт · год
13.10.2023 02:30	0.17
13.10.2023 02:31	0.18
13.10.2023 02:32	0.20
13.10.2023 02:33	0.25
13.10.2023 02:34	0.30

З таблиці можна зрозуміти, що мінімальне значення використання електроенергії для даного прикладу складає 0.17 кВт · год, а максимальне значення 0.30 кВт · год. Після цього можна сформувати таблицю вже з нормалізованими значеннями в діапазоні від 0 до 1, а також переведеною датою та часом в epoch seconds (табл.3.3.).

Таблиця 3.3.

Нормалізовані дані використання електроенергії

Дата та час(epoch seconds)	Нормалізована витрата електроенергії
1697171400	0.00
1697171460	0.07
1697171520	0.21
1697171580	0.71
1697171640	1.00

Таким чином найменше значення в нормалізованому вигляді буде дорівнювати 0.00, а найбільше значення 1.00.

3.3.2. Шари та методи активації нейронної мережі

Для того, щоб правильно обрати тип шару та активації, потірбно правильно сформувати завдання для навчання мережі. В даному випадку це прогнозування використання електроенергії тому найкращим вибором буде RNN з шарами LSTM.

Метод активації – це математична функція, яка визначає вихід нейрона в мережі на основі вхідних даних. Такі функції додають нейлінійності моделі, що дозволяє нейронній мережі навчатись складним функціям, які відповідатимуть реальним данним [29].

Першою функцією активації є Sigmoid. Значення виходу такої функції, обмежуються в діапазоні 0 – 1. Така функція найчастіше використовується, коли потірбна ймовірнісь виходу.

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{3.2.}$$

де, x – значення входу або виходом з попереднього шару.

Наступною функцією активації є Tanh(Hyperbolic Tangent). Значення виходу даної функції, знаходяться в діапазоні від -1 до 1. Така функція активації використовується в RNN моделях, де потрібна симетрія.

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \tag{3.3.}$$

де, x – значення входу або виходом з попереднього шару.

Третьою функцією активації є ReLU(Rectified Linear Unit). Вихідний результат буде нульовим, якщо вхідне значення менше нуля і буде орівнювати вхідному значенню, якщо воно блыіше або дорівнює 0. Даний метод простий та дійсно ефективний.

$$\mathrm{Re}LU(x) = \max(0, x), \tag{3.4.}$$

де, x – значення входу або виходом з попереднього шару.

Функція активації Softmax перетворює вектор значень в ймовірності, які в сумі дають 1.

Дана функція використовується в останньбому шарі нейронної мережі для класифікації з кількома класами.

$$Soft\max(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}, \quad (3.5.)$$

де, x – значення входу або виходом з попереднього шару.

Для завдання прогнозування енерговитрат, було підібрано наступні налаштування:

- updater(new Adam()), є оптимізатором для навчання мережі. За допомогою якого можна зменшити час навчання мережі;
- weightInit(WeightInit.RELU), встановлюється метод ініціалізації ваг з вагою WeightInit.RELU;
- LSTM.Builder(), визначає шар LSTM;
- nIn(1), це кількість входів в шарі, а саме часовий ряд з однією ознакою;
- nOut(70), кількість виходів з шару, тобто кількістьнейронів у шарі LSTM;
- activation(Activation.TANH), метод активаційної функції, яка потребує нормалізації даних в діапазоні від -1 до 1;
- RnnOutputLayer.Builder(), визначає вихідний шар, для рекурентної мережі;
- nIn(70), кількість входів вихідного шару;
- nOut(1), кількість виходів вихідного шару, тобто 1 прогнозоване значення.

Дані налаштування навчають нейронну мережу прогнозувати наступні дані на основі аналізування попередніх значень. Окрім цього можна додавати додаткові шари, що зробить більш точними прогнози, але потрібно памятати, що мережу не можна перенавчувати, бо це буде призводити до помилок прогнозування. Крім цього збільшення шарів, або нейронів потребує більшої

кількості оперативної памяті для навчання мережі, що може ускладнювати даних процес, тому треба використовувати можливі оптимізатори.

3.3.3. Навчання мережі та оцінка моделі

В даному підпункті розглядається декілька конфігурацій, та порівняння оцінок навченої мережі на основі нормалізованих даних. Зазвичай для навчання мережі потрібно велика кількість даних, так як зустрічається більше варіацій, і в подальшому можна краще виконувати прогнозування [30].

Перша конфігурація мережі має в собі один шар, RRN з методом активації IDENTITY, в якому 1 нейрон, дані формуються в пакети розміром 32, та кількість епох навчання дорівнює 300 (табл.3.4.).

Таблиця 3.4.

Оцінка мережі RRN

Кількість шарів та характеристики	Кількість нейронів	Пакет даних	Кількість епох	Результат
1.RRN, activ.=IDENTITY	1	32	300	• MSE - $3.47 \cdot 10^{15}$ • MAE - $5.03 \cdot 10^{7}$ • RMSE - $5.89 \cdot 10^{7}$ • RSE - $5.13 \cdot 10^{17}$ • PC - - 0.988 • R^2 - - $5.13 \cdot 10^{17}$

Результат показує ефективність мережі, і має наступні показники:

- MSE(Mean Square Error) – середній квадрат помилок моделі;
- MAE(Mean Absolute Error) – оцінка середньої абсолютної різниці між прогнозованим значенням та фактичним;
- RMSE(Root Mean Square Error) – квадратний корінь з MSE;
- RSE(Residual Standart Error) – оцінка стандартного відхилення помилок моделі;

- PC(Pearson Correlation) – кореляція Пірсона, показує лінійну залежність між прогнозованим та фактичним значенням.
- R^2 – оцінка частки дисперсії залежної змінної.

За результатами першого дослідження можна зробити висновки, що MSE та RMSE дуже великі значення, це вказує на те, що модель робить великі помилки у передбаченнях. MAE також має велике значення, яке означає що модель сильно помиляється. RSE вказує на те що можель не ефективна у порівнянні з простою моделлю передбачення середнього значення. Від'ємні значення PC та R^2 вказують на велику різницю між передбачуваними значеннями та фактичними, що є поганою ознакою.

Наступною конфігурацією для моделі буде два шари, LSTM та RNN, функція активації Tanh для шару LSTM, та IDENTITY для шару RNN. Кількість нейронів для даного випадку збільшена до 100, дані формуються у пакети розмірністю 64, та кількість епох навчання 200 (табл.3.5.).

Таблиця 3.5.

Оцінка мережі на основі шарів LSTM та RNN

Кількість шарів та характеристики	Кількість нейронів	Пакет даних	Кількість епох	Результат
1.LSTM, activ. = TAHN 2.RRN,activ.=IDENTITY	100	64	200	• MSE - $1.55 \cdot 10^{-4}$ • MAE - $7{,}37 \cdot 10^{-3}$ • RMSE - $1.24 \cdot 10^{-2}$ • RSE - $2.3 \cdot 10^{-2}$ • PC - $9.88 \cdot 10^{-1}$ • R^2 - $9.77 \cdot 10^{-1}$

Оцінка даної моделі значно покращилась в порівнянні з попередньою, тому потрібно проаналізувати. MSE та RMSE параметри вказують, що модель має низьке значення середньої квадратичної помилки та квадрату середньої квадратичної помилки, отже модель добре може перебачати значення. Параметр MAE означає, що модель помиляється менш ніж на 0.01, що є добрим

показником. Коефіцієнт Пірсона близький до 1, що говорить про сильну позитивну кореляцію між передбаченнями моделі та фактичними значеннями. R^2 вказує на точність моделі, і чим значення ближче до 1, тим краще модель пояснює варіацію даних.

Третьою дослідницькою конфігурацією мережі є застосування трьох шарів, з яких два шари LSTM та один RNN, збільшена кількість нейронів до 40, пакет даних розмірністю 16, та кількість епох зменшена до 100. В даному випадку кількість нейронів була зменшена через те, що мережа навчалась на восьмидесяти тисячах пар даних. В такому випадку потрібно було або знизити конфігурацію мережі, або зменшити кількість вхідних даних (табл.3.6.).

Таблиця 3.6.

Оцінка мережі з трьома шарами (два LSTM та один RNN)

Кількість шарів та характеристики	Кількість нейронів	Пакет даних	Кількість епох	Результат
1.LSTM, activ. = THAN 2.LSTM, activ. = TAHN 3.RRN,activ.=IDENTITY	40	16	100	• MSE – $1.55 \cdot 10^{-4}$ • MAE - $7{,}62 \cdot 10^{-3}$ • RMSE - $1.24 \cdot 10^{-2}$ • RSE - $2.2 \cdot 10^{-2}$ • PC - $9.78 \cdot 10^{-1}$ • R^2 - $9.6 \cdot 10^{-1}$

В третьому випадку мережа дуже схожа на попередню, з одним шаром LSTM, але трошки з гіршими показниками, що може вказувати на невелику переначеність мережі, чого слід уникати.

Ще одним дослідним випадком є навчання мережі з трьома шарами LSTM та вихідним шаром RNN. Було збільшено загальну кількість нейронів до 400 нейронів, кількість епох навчання збільшена до 600, також змінений метод активації для LSTM шарів, з Tanh на Sigmoid і доданий параметр до цих шарів DropOut з коефіцієнтом 0.2 для запобігання перенавченості моделі. В підсумку отримано наступні результати (табл. 3.7.).

Таблиця 3.7.

Оцінка мережі з чотирма шарами(три LSTM та один RNN)

Кількість шарів та характеристики	Кількість нейронів	Пакет даних	Кількість епох	Результат
1.LSTM,activ.= Sigmoid 2.LSTM,activ.= Sigmoid 3.LSTM,activ.= Sigmoid 4.RRN,activ.=IDENTITY	400	32	600	• MSE - $6{,}15 \cdot 10^{-3}$ • MAE - $6{,}75 \cdot 10^{-2}$ • RMSE - $7{,}84 \cdot 10^{-2}$ • RSE - $9{,}09 \cdot 10^{-2}$ • PC - $9{,}61 \cdot 10^{-1}$ • R^2 - $9{,}07 \cdot 10^{-2}$

Порівнюючи результати дослідження по навчанню нейромережі, можна сказати що не завжди кількість нейронів, збільшення шарів та інших параметрів, гарантуватиме стабільну оцінку мережі. Далі потрібно порівняти результати прогнозування, вже навчених мереж на підготовлених тестових даних.

В першу чергу, для тестування потрібно виділити близька 15000 пар записів, часу вимірювання та використання електроенергії. Після цього ці дані нормалізуються аналогічно до навченої мережі, і в подальшому завантажуються в неї для обробки цих даних та прогнозування.

На рис. 3.12. можна побачити реальні дані про використання електроенергії, та прогнозовані дані.

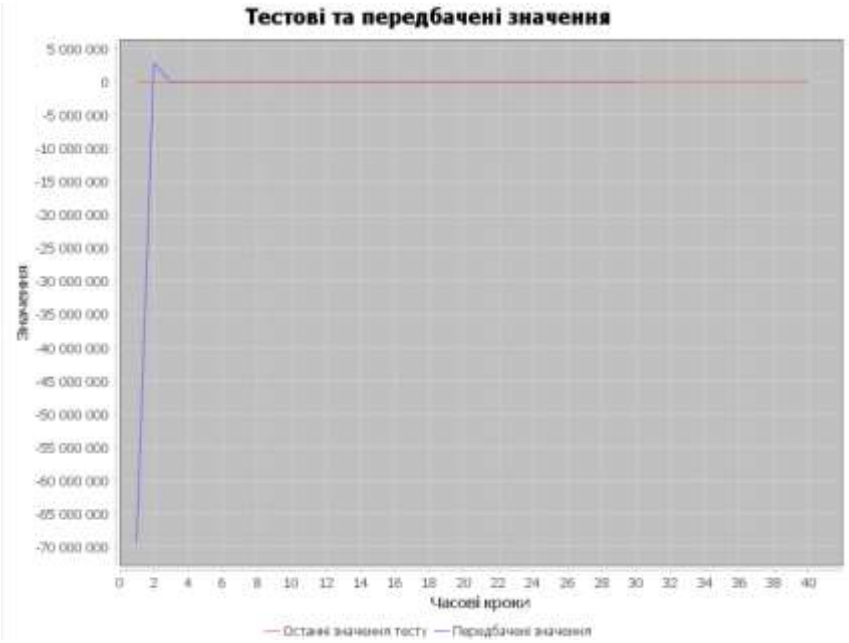


Рис.3.12. Графік тестових та передбачуваних значень для першої конфігурації

З рисунку видно, що на першій хвилині прогнозу, мережа зпрогнозувала відємне значення, і доволі велике, враховуючи, що діапазон використання електроенергії в тестовому наборі даних коливається в діапазоні від 0.12 до 0.45 кВт · год. Дана мережа з одним шаром RNN не підходить для задачі прогнозування, адже має велику похибку.

Наступна конфігурація мережі має два шари, а саме LSTM та RNN. Аналіз MSE показав, що дана мережа непогано підходить для прогнозування використаня електроенергії. На наступному рисунку можемо бачити, що перші 6 ітерацій відбувались у діапазоні тестового набору даних, але після цього напрямок передбачуваних значень змінився у відємну сторону, а це означає що мережа має певні проблеми, які аналіз MSE не бачить. Можливо дана мережа перенавчена, внаслідок чого отримуємо значення прогнозування з відємним знаком (рис.3.13.).

Рис.3.13. Графік тестових та передбачуваних значень для другої конфігурації

Третя конфігурація за оцінкою дуже схожа на першу, але має в собі меншу кількість нейронів, але на один LSTM шар більше. Рисунок 3.14. показує невелику різницю між другою та третьою конфігурацією, але в загальному вона також не підходить для завдання прогнозування.

Рис.3.14. Графік тестових та передбачуваних значень для третьої конфігурації

Четверта конфігурація відрізняється найбільше, так як має більше нейронів, більше шарів, має налаштування для запобігання перенавчанню

нейронної мережі, а також, збільшена кількість епох. На рис.3.15. показана тестові та передбачені значення останньої конфігурації.

Рис.3.15. Графік тестових та передбачуваних значень для четвертої конфігурації

Четверта нейронна мережа, добре справилась з прогнозуванням витрат, адже вона знаходиться в діапазоні тестового набору даних, повторює графік останніх значень тестування, але навіть дану конфігурацію можна покращити. Для цього можна спробувати підвищити кількість шарів, трохи збільшити кількість нейронів. Крім цього можна зменшити або збільшити коефіцієнт перенавченості мережі, збільшити пакет даних, який передається в мережу для прогнозування, або ж зробити ще більше епох навчання.

В наслідок цих досліджень було підібране найкраще налаштування для навчання мережі, а саме три шари LSTM з функцією активації Sigmoid та RNN з функцією активації IDENTITY, кількість нейронів 400, пакет даних розмірнісю 32, та кількість епох дорівнює 600.

Висновки до третього розділу

В даному розділі описане прогнозування використання електроенергії на основі різних типів лампочок, наведені розрахунки, досліджено як дімер впливає на енерговикористання. Розроблений інтерфейс під дану задачу на ОС Android, також написано серверну чатину, де працює алгоритм, повний код в додатку А.

Другим етапом було дослідженння різних нлаштувань навчання мережі та їх можливість до прогнозування ними використання електроенергіїї, наведені графіки з результатами, а також таблиці до кожної конфігурації.

4. РОЗРОБКА СТАРТАП ПРОЕКТУ

Стартапи в останній час доволі часто створюються для заповнення певної ринкової ніші або ж вирішення певної проблеми і відрізняються високим рівнем невизначеності та ризику. Стартапи зазвичай пропонують новіі продукти або послуги, які можуть певним чином допомагати суспільству у вирішення їх проблем.

Дана форма ініціативи або бізнесу часто проходять через багато рівнів тестування та розробки, аналізування ринку, отримання зворотнього відгуку від потенційних клієнтів. Основною перевагою стартапів є швидка зміна вектору розвитку проекту, масштабування його в критично малий проміжок часу. В стартапі є головними ідея, розробка, пошук фінансування, запуск продукту та масштабування.

4.1. Опис ідеї проекту

Опис ідеї стартапу можна побачити в табл. 4.1., на якій можна побачити зміст ідеї, напрямки застосування, а також вигоди для користувача.

Таблиця 4.1.

Опис ідеї стартап - проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Зміст ідеї: розробка алгоритму прогнозування та керуваннння системою SmartGrid	Приватні користувачі	Самостійний моніторинг
	Компанії, які займаються постачанням електроенергії	Можливість аналізувати систему та керувати її складовими на основі прогнозів

Для вдосконалення оцінки ідеї потрібно зробити аналіз потенційних техніко-економічних переваг порівнюючи з пропозиціями конкурентів.

В табл. 4.2. наведено техніко-економічні характеристики та порівняння з конкурентами.

Таблиця 4.2.

Визначення спильних, слабких та нейтральних характеристик ідей проекту

№ п/п	Техніко-економічні характеристики ідеї				
		Мій проект	Конку-рент 1	Конку-рент 2	Конку-рент 3
1.	Інтерфейс для моніторингу	+	+	-	-
2.	Прогнозування витрат використовую-чи нейронну мережу	+	-	-	+
3.	Керування системою за даними прогнозів	+	-	-	-
4.	Підключення резервного живлення в систему	+	-	+	+

4.2. Технологічний аудит ідеї проекту

Технолонічна здійсненість ідеї проекту представлена в табл. 4.3.

Таблиця 4.3.

Технологічна здійсненість ідеї проекту

№ п/п	Ідея проекту	Технології та реалізації	Наявність технологій	Доступність технологій
1	Масштабування	Залучення користувачів до SmartGrid	Наявна	Доступна
2	Підвищення точності прогнозів	Тестування керування на основі прогнозованих даних	Наявна	Доступна

З таблиці здійсненності, можна зробити висновок, що при залученні корситувачів до системи SmartGrid, буде можливість обробляти більше даних,

внаслідок чого навчати нейронну мережу до більш складних маніпуляцій і не стандартних випадків.

4.3. Аналіз ринкових можливостей запуску стартап проекту

Характеристика потенційного ринку стартап проекту в табл. 4.4.

Таблиця 4.4.

Попередня характеристика потенційного ринку стартап проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од.	3
2	Загальний обсяг продаж, грн/ум.од.	-
3	Динаміка ринку(якісна оцінка)	Зростає
4	Наявність обмежень для входу	-
5	Специфічні вимоги до стандартизації та сертифікації	Похибка 5%
6	Середня норма рентабельності в галузі(або по ринку), %	>90%

В таблиці 4.5. відобращена характеристика потенційних клієнтів стартап проекту.

Таблиця 4.5.

Характеристика потенційних клієнтів стартапу

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Підвищення ефективності SmartGrid	1) Компанії, які займаються постачанням електроенергії 2) Споживачі електроенгергії	Керування та моніторинг системи	Точні дані прогнозу, довготривалість пристроїв, зменшення навантаження в системі

Фактори загроз наведені в таблиці 4.6.

Табл.4.6.

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Ймовірність появи нових рішень, які будуть економічно дешевші, прогресивніші	Швидка реакція на зміни тенденції наукового світу та підлаштування під нові аспекти

В таблиці 4.7. вказані фактори можливостей та їхній зміст, а також можливу реакцію компанії.

Таблиця 4.7.

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Розширення кількості клієнтів підключених до системи	Завдяки збільшенню підключень, можна обробляти більше даних, внаслідок чого отримати добре навчену нейромережу	Заохоченя клієнтів долучатись до системи
2	Впровадження автономного керування, яким буде займатись нейронна мережа	Автономне керування системою зможе запобігти більшості типів аварійних ситуацій в SmartGrid	Дослідження додаткових параметрів, детальний аналіз передачі електроенергії

Наступна табл. 4.8. показує ступеневий аналіз пропозиції, в якому визначені загальні риси конкуренції на ринку.

Таблиця 4.8.

Ступеневий аналіз конкуренції на ринку

Особливості конкуретного сережовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції: монополія	Низька кількість виробників і велика кількість користувачів	Тиск з боку “великих” підприємців
2. За рівнем конкурентної боротьби: локальний	Підприємства по різному використовують електроенергію	Пристосування до обробки великих по обєму даних
3. За галузевою ознакою: внутрішньогалузева	Керування за допомогою нейромережі та автоматики	Потрібно постійно покращувати мережу, перевіряти обладнання
4. Конкуренція за видами товарів: між бажаннями	Кожен користувач має право право вибору моніторингу та керування власним споживанням	Забезпечення споживача постійною технічною підтримкою
5. За характером конкурентних переваг: цінова	Навчання системи та її керування не потребує великих затрат на вдосконалення	Збільшення рентабельності
6. За інтенсивністю: марочна	Високий розвиток даної сфери	Розвиток компанії, реклама

Наступним кроком аналізу можливостей для впровадження проекту є складання SWOT-аналізу(матриці аналізу сильних(Strength) та слабких(Weak) сторін, загроз (Troubles), та можливостей (Opportunities)) на основівиділених ринкових загроз та можливостей, та сильних і слабких сторін.

Перелік ринкових загроз і можливостей визначається шляхом оцінки факторів, які впливають на ринкове середовище. Ринкові загрози та можливості є передбачуваними наслідками впливу цих факторів і, на відміну від них, наразі

не виявлені на ринку, але мають певну ймовірність реалізації. Наприклад, зниження доходів потенційних споживачів може бути розглянуто як фактор загрози, що може викликати зростання важливості цінового фактору при виборі товару і, відповідно, створювати ринкову загрозу в області цінової конкуренції.

В табл. 4.9. відображено SWOT аналіз ідеї.

Таблиця 4.9.

SWOT аналіз ідеї

Сильні сторони: • Новизна; • Простота у використанні; • Інтелектуальне керування.	Слабкі сторони: • Невідомий бренд на ринку; • Відсутність досвіду порівняно з великими компаніями.
Можливості: • Підвищененя клієнтоорієнтованості; • Співпраця з великими компаніями.	Загрози: • Велика конкуренція; • Націоналізація

Наступним параметром буде аналіз ринкового впровадження стартап проекту, який описаний в табл. 4.10.

Таблиця 4.10.

Альтернативи ринкового впровадження стартап проекту

№	Альтернатива(орієнтований комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Зменшення загроз завдяки сильним сторонам проекту	50%	2 роки
2	Передача розробки вже існуючим підприємствам	80%	1 рік
3	Підвищення точності результатів, полегшення використання	70%	4-5 років

Після аналізу успіху проекту було обрано третю стратегію як основну, адже завдяки ній можна зберегти ідею та проект.

4.4. Розробка ринкової стратегії проекту

Наступним проводиться опис цільових груп потенційних споживачів для того, щоб визначити яка стратегія охоплення ринку буде найкращою (табл.4.11).

Таблиця 4.11

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Підприємства	Висока	Висока	Середня	Середня
2.	Науково дослідницькі-інститути	Середня	Середня	Висока	Середня
3.	Приватні користувачі	Середня	Середній	Низька	Низька
Які цільові групи обрано:Проаналізувавши дані у таблиці, було прийнято рішення розвивати проєкт у напрямку приватних користувачів.					

Після аналізу груп потенційних споживачів, було обрано стратегію охоплення рингу – стратегія диференційованого маркетингу. В наступній табл. 4.12. показана стратегія розвитку для роботи в обраному сегменті ринку.

Таблиця 4.12.

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Зменшення вартості розробки системи та покращення його зручності у використанні	Диференційо-ваний маркетинг	Якісний продукт із гнучкими налаштуваннями та зручним інтерфейсом користувача	Стратегія диференціації

В табл. 4.13. проводяться визначення базової стратегії конкурентної поведінки.

Таблиця 4.13.

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1.	Ні	Так	Ні	Позиціонування категорії

Підсумувавши, визначається стратегія позиціонування для ідентифікації власного продукту споживачем. В табл. 4.14. наведені результати стратегії позиціонування.

Таблиця 4.14.

Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Висока якість продукту Широкий спектр можлиостей використання	Стратегія диферен-ціації	Корисний продукт, який зможе економити на оновленні приладів керування, та дасть змогу використати передбачення для запобіганя аварійних ситуацій	Надійність Якість Необхідність

4.5. Розробка маркетингової програми стартап проекту

В табл. 4.15 показані ключові переваги концепції потенційного товару.

Таблиця 4.15.

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1.	Прогнозування використання електроенергії на основі історичних даних, та керування мережею відповідно прогнозу	Ефективність	Зручний дизайн та інтерфейс користувача, постійне покращення алгоритмів роботи пристрою

В табл. 4.16 показана розроблена трирівнева модель товару для уточнення ідеї.

Таблиця 4.1.

Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Продукт зменшує використання електроенергії, завдяки навченій нейромережі. Алгоритм легко адаптується та налаштовується.</td></tr>
<tr><td rowspan="9">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Доступність</td><td>Нм</td><td>Е</td></tr>
<tr><td>2.Зручність налаштування</td><td>М</td><td>Тх</td></tr>
<tr><td>3.Інтерфейс користувача</td><td>Нм</td><td>Тх</td></tr>
<tr><td>4.Функціонал</td><td>М</td><td>Тх</td></tr>
<tr><td>5. Технічна підтримка</td><td>Нм</td><td>Ор</td></tr>
<tr><td>6. Вартість</td><td>М</td><td>Е</td></tr>
<tr><td colspan="3">Марка: Neural Grid</td></tr>
<tr><td colspan="3">До продажу: Надання інструкції з експлуатації клієнтам, безкоштовні технічні консультації, участь у технічних форумах та реклама товару</td></tr>
<tr><td>III. Товар із підкріпленням</td><td colspan="3">Після продажу: Безкоштовна технічна підтримка, покращення алгоритмів роботи пристрою, збір статистики користування</td></tr>
<tr><td colspan="4">Алгоритм буде захищений за допомогою SSL ключа, тому всі запити між клієнтом та сервером будуть проходить через захищений протокол https.</td></tr>
</table>

*М/Нм – монотонні/немонотонні; Вр/Тх/Тл/Е/Ор – вартісні/ технічні/ технологічні/ ергономічні/ органолептичні;

Висновки до четвертого розділу

Підсумовуючи все вище описане, можна сказати, що проект є дійсно перспективним, якщо притримуватись маркетингової стратегії. Проект має непогані переваги та можливості на ринку, порівняно з іншими компаніями. Основною перевагою є можливість автоматизовувати складні часитини системи

і тільки спостерігати за роботою. Також не має обмежень по масштабуванню, адже до такої системи можна додавати безліч контролерів, датчиків, які будуть знімати показники і так далі. Тому проект є дійсно конкурентноспроможним, не дивлячись на складність виходу на ринок.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

Під час виконання магістерської дисертації, було проаналізовано велику кількість літератури за темою дослідження. Було знвято велику кількість даних на реальному приміщення, для того щоб промоделювати прогнозування нейронної мережі.

Розроблений алгоритм для прогнозування використання електроенергії різними липами ламп, а саме галогенними, накалювання, флюоресцентними та LED в залежності від конкретного дня в місяці, використовуючи таку змінну як тривалість темної пори доби. Окрім цього досліджено як дімер може заощадити електроенергії, на прикладі ламп. Для цього створений інтерфейс в ОС Android, серверну частину, де відбувається обчислення, доступ до БД, запити за допомогою REST API.

В другій частині дослідження було порівввняно різні типи нейронних мереж, а саме LSTM та RNN мережі. Проаналізовані їхні властивості, методи активації, досліджено залежності між памяттю, яка потрібна для навчання мережі, і кількістю шарів, нейронів, обєднання в пакет даних і так далі. Розглянуто різні методи активації шарів нейроннох мережі а саме Sigmoid, TANH, RELU, SOFTMAX.

Зроблено оцінку чотирьох конфігурацій мережі за допомогою метрик MSE, проаналізовано результати навчання цих нейронних мереж. І останнім етомпом зроблено тестування навчених мереж на реальних даниї, і порівняно наскільки точно вони можуть прогнозувати використання електроенергії в залежності від історичних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tracking Smart Grids [Електронний ресурс] – URL: https://www.iea.org/energy-system/electricity/smart-grids#tracking
2. ДСТУ IEC 61850-5:2019 Комунікаційні мережі та системи для автоматизації електроенергетичних підприємств. Частина 5. Технічні вимоги до функцій і моделей приладів (IEC 61850-5:2013, IDT)
3. ДСТУ IEC/TR 61850-1:2018 Комунікаційні мережі та системи для автоматизації електроенергетичних підприємств. [Електронний ресурс] – URL: https://zakon.rada.gov.ua/rada/show/v1423731-13#Text
4. Environmental Initiatives In Baldwin Park[Електронний ресурс] – URL: https://facts.net/world/cities/10-facts-about-environmental-initiatives-in-baldwin-park-california/
5. Amsterdam Smart City [Електронний ресурс] – URL: https://amsterdamsmartcity.com/
6. Особливості технології LoRaWAN [Електронний ресурс] – URL: https://iotji.io/osoblyvosti-lorawan/
7. Internet of Energy (IoE): What it is, How it Works [Електронний ресурс] – URL: https://www.investopedia.com/terms/i/internet-energy-ioe.asp
8. Saleh M.S., Althaibani A., Esa Y., Mhandi Y., Mohamed A.A. Impact of clustering microgrids on their stability and resilience during blackouts. In Proceedings of the 2015 International Conference on Smart Grid and Clean Energy Technologies. Offenburg, Germany, 2020. 2032 p
9. Грешилов А. А., Стакун В. А., Стакун А. А. Математические методы построения прогнозов. — М.: Радио и связь, 1997.- 112 с. — ISBN 5-256-01352-1
10. Chris Chatfield (1996). *The Analysis of Time Series, an Introduction* (вид. 5-те). Chapman & Hall/CRC. с. 33

11. Notation for ARIMA Models [Електронний ресурс] – URL: https://support.sas.com/documentation/cdl/en/etsug/63939/HTML/default/viewer.htm#etsug_tffordet_sect016.htm

12. Efficient Artificial Neural Network for Smart Grid Stability Prediction [Електронний ресурс] – URL: https://www.hindawi.com/journals/itees/2023/9974409/

13. Chiang, C.L, (2003) *Statistical methods of analysis*, World Scientific. ISBN 981-238-310-7 - page 274 section 9.7.4 "interpolation vs extrapolation"

14. Що таке багатошаровий персептрон (Multilayer Perceptron, MLP) у машинному навчанні? [Електронний ресурс] – URL : https://thetransmitted.com/adlucem/shho-take-mlp-u-mashynnomu-navchanni/

15. Іванов А. О. Теорія автоматичного керування: Підручник. — Дніпропетровськ: Національний гірничий університет. — 2003. — 250 с.

16. Загальна електротехніка і основи електроніки: навчальний посібник / Співак В.М., Гуржий А.М., Нельга А.Т., Ітякін О.С.– Київ: КПІ, 2020. – 266 с., 155 рис., 10 табл., 17 бібл

17. Інтелектуальні системи забезпечення енергозбереження житлових будинків. Навчальний посібник./ Петергеря Ю.С., Жуйков В.Я., Терещенко Т.О. – К.: Медіа-ПРЕС, 2008. – 256 с.

18. Пантєєв Р.Л. «Нейронні мережі у керуванні складними технічними системами» // Вісник Київського інституту бізнесу і технологій, 2019 № 3(37)2019, УДК 621.396.551.553, стор. 27-36.

19. Анісімов А.В., Кулябко П.П. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. - Київ. – 2017. – 110 с.

20. Основи роботи з MySQL Workbench [Електронний ресурс] – URL: http://kimt.uad.edu.ua/static/media/files/inzheneriia-danikh-ta-znan-fvpit-5/robota-v-mysql-workbench.pdf

21. Hitecsa [Електронний ресурс] – URL: https://www.hitecsa.com/

22. Refrigeration equipment [Електронний ресурс] – URL: https://www.rivacold.com/ww/en/

23. Flexible broiler [Електронний ресурс] – URL: https://nieco.com/wp-content/uploads/2015/07/JF74-6-30-Email.pdf

24. FryMaster [Електронний ресурс] – URL: https://www.frymaster.com/Products/Fryers/Oil-Conserving-Fryers/OFC30

25. Oracle JDBC Driver [Електронний ресурс] – URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

26. Spring [Електронний ресурс] – URL: https://spring.io/

27. Richardson L. RESTful Web APIs: Services for a Changing World 1st Edition. – RESTful, 2013. - 406p. – ISBN 978-1449358068

28. Основи штучних нейронних мереж [Електронний ресурс] – URL: https://learn.ztu.edu.ua/pluginfile.php/290224/mod_resource/content/1/СШІ_Л-5_Нейронні_мережі.pdf

29. Штучні нейронні мережі в задачах групування та аналізу інформації [Електроенний ресурс] – URL: https://csc.knu.ua/media/study/asp/art_net_group_inf_akimenko/lecture/lec1.pdf

30. Методи та технології обчислювального інтелекту: Навчальний посібник [Електронний ресурс] : навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки» / І. В. Федорін; КПІ ім. Ігоря Сікорського. – Електронні текстові дані. – Київ: КПІ ім. Ігоря Сікорського, 2022. – 314 с

ABSTRACT

Relevance. In the context of the current development of the electric power industry, the issue of efficient use of electricity is extremely acute. One of the most effective areas of development involves the widespread use of SmartGrid systems, which has a number of advantages for individual consumers as well as at the level of household associations, cottage communities, districts, and even entire cities. In this context, electricity management is crucial for managing and balancing electricity supply and demand. Research in the field of electricity demand management is essential for the further development of the energy industry and ensuring a sustainable energy future.

One of the key aspects of SmartGrid is electricity consumption management, which implies the ability to actively respond to changes in electricity consumption, ensure efficient energy distribution, and reduce losses.

Thus, the topic of the master's thesis, which is devoted to solving the problem of management and forecasting, is relevant. The solution of the tasks set out in the thesis will help to improve the energy efficiency of the SmartGrid, reduce energy and money costs.

Relationship of the work to scientific programs, plans, topics. The thesis was prepared in accordance with the research plan of the Department of Electronic Devices and Systems of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

Purpose and objectives of the study. The purpose of the study is to develop effective control algorithms, study and analyze control strategies in the SmartGrid system, model the smart grid system, and develop appropriate software with the ability to predict using neural networks. The objectives are to study modern solutions for system management and forecasting, analyze existing infrastructures, identify factors that may affect efficiency, and develop new algorithms and evaluate their effectiveness.

Object and subject of research. The object of research is the process of managing and forecasting electricity consumption in the SmartGrid, and the subject of research is forecasting electricity consumption in the SmartGrid system.

Research methods. To achieve the results, the following research methods were used, namely literature analysis, mathematical modeling, and statistical analysis. Literary analysis was used to analyze scientific sources related to the management of electricity consumption in the SmartGrid system. Mathematical modeling is used to develop algorithms for managing electricity consumption in the system. Statistical analysis is used to process and analyze data, and it also allows for quantitative evaluation of the research results and drawing conclusions about the results obtained.

The scientific novelty of the results obtained is the development of a new approach to power management in the system using programming and electronics. The study improves the calculation of power losses in different scenarios, as well as under the influence of various factors. The obtained results allow to improve the efficiency of the system in the field of power consumption and increase its reliability.

Practical significance of the results:

- An algorithm has been developed that calculates electricity consumption depending on different types of consumers and the amount of daylight hours;

- The data obtained was analyzed to reduce electricity consumption.

This algorithm can be used both for predicting electricity consumption and for consumer management, which greatly facilitates the monitoring and control of the system.

Publications. The system of detection of hazardous substances in the air based on the Internet of Things for the Bulletin of the Mykhailo Ostrohradskyi National University. Participation in the conference of young scientists “Electronics - 2024” with the topic: “Total Power Management in the Smart Grid System”.

Structure and scope of the thesis. The work consists of 4 chapters. The first chapter discusses energy consumption in the Smart Grid system, namely, ways to track energy consumption and possible ways to forecast it; the second chapter discusses methods of energy consumption control using dimmers, scenario control,

and neural networks; the third chapter describes the development of an algorithm for predicting energy consumption by lamps of different types during the dark time of day, as well as predicting electricity use using a neural network; the fourth chapter describes the startup project. The work contains 73 pages, 21 figures, 25 tables, a list of 30 references on 3 pages.

ДОДАТОК А. ЛІСТИНГ СЕРВЕРНОЇ ЧАСТИНИ ДОДАТКУ

```
package com.powerconsumpion.powerconsumptionapplication;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PowerConsumptionApplication {

    public static void main(String[] args) {
       SpringApplication.run(PowerConsumptionApplication.class, args);
    }

}
```

```
package com.powerconsumpion.powerconsumptionapplication.model;

import com.google.gson.Gson;
import com.google.gson.internal.LinkedTreeMap;
import com.powerconsumpion.powerconsumptionapplication.addData.dto.LightTime;
import com.powerconsumpion.powerconsumptionapplication.addData.dto.Months;
import com.powerconsumpion.powerconsumptionapplication.api.LightTimeController;
import lombok.extern.slf4j.Slf4j;

import java.util.ArrayList;
import java.util.List;

@Slf4j
public class ElEffLamp {
    private String months;
    private ArrayList<String> arrayList;
    private Gson gson = new Gson();

    private int elementsInStream = 0;

    public ElEffLamp(String months) {
        this.months = months;
        this.arrayList = new ArrayList<>();
        getLightTime();
    }

    private ArrayList<String> getLightTime() {
        if (months.contains("-")) {
            int length = months.length();
            int firstMonth = Integer.parseInt(String.valueOf(months.charAt(0)));
            int lastMonth;
            int range = firstMonth;
            if (length == 3) {
                lastMonth = Integer.parseInt(String.valueOf(months.charAt(length
- 1)));
            } else {
                lastMonth = Integer.parseInt(new String(new char[] {
months.charAt(length - 2), months.charAt(length - 1)}));
            }
            while (range - 1 != lastMonth) {

arrayList.add(gson.toJson(LightTimeController.getMonthsInfoAboutLightTime(range)
));
                range++;
            }

        } else {
            log.info("here");
```

```
            if (months.length() == 2) {

arrayList.add(gson.toJson(LightTimeController.getMonthsInfoAboutLightTime(Intege
r.parseInt(new String(new char[] { months.charAt(0), months.charAt(1)}))))));
            } else {

arrayList.add(gson.toJson(LightTimeController.getMonthsInfoAboutLightTime(Intege
r.parseInt(String.valueOf(months.charAt(0))))));
            }

        }
        return arrayList;
    }


    public double calculate(int quantity, double power) {
        log.info("Q: " + String.valueOf(quantity));
        int sizeOfArrayList  = arrayList.size();
        log.info(String.valueOf(sizeOfArrayList));
        int s = 0;
        double avg = 0.0;
        double eConsumptionFull = 0.0;
        while (s != sizeOfArrayList) {
            int size = arrayList.get(s).length();
            log.info("size: " + size);
            avg = getAvgWorkTime(arrayList.get(s));


            double avgTimeInOneDay = avg/60; // in hours
            log.info("avgTimeInOneDay: " + avgTimeInOneDay);
//            double avgTimeInHours = avgTimeInOneDay/60;  // 13.8166667
//            log.info(String.valueOf(avgTimeInHours));
            avgTimeInOneDay = avgTimeInOneDay - (double) 7.5;
            log.info(String.valueOf(avgTimeInOneDay));
//
            double eConsumePerDay = avgTimeInOneDay * power * 0.001;
            log.info("eConsumePerDay: " + eConsumePerDay + "\nwith power: " +
power);
            double eConsumePerAllTime = eConsumePerDay * elementsInStream;
            log.info("eConsumePerAllTime: " + eConsumePerAllTime);
            eConsumptionFull += eConsumePerAllTime * (double) quantity;
            log.info("eConsumptionFull :" + eConsumptionFull);
            log.info("avg: " + avg);
            s++;
        }

        return eConsumptionFull;
    }

    private double getAvgWorkTime(String info) {
        log.info("info:" + info);
        Months months = gson.fromJson(info, Months.class);

        int fullDay = 1440; //minutes in full day
        int fullLightTime = 0;
        log.info("info2: " + months.getDays());
        ArrayList<LinkedTreeMap> list = gson.fromJson(months.getDays(),
ArrayList.class);
        List<LightTime> lightTimes = list.stream()
                .map(s -> gson.fromJson(gson.toJson(s), LightTime.class))
                .toList();
        log.info("LightTimes: " + String.valueOf(list));
```

```
         elementsInStream = 0;
        for (LightTime lightTime1 : lightTimes) {
            String time = lightTime1.getLightTime();
            String hours = new String(new char[] { time.charAt(0),
time.charAt(1)});
            String min = new String(new char[] { time.charAt(3),
time.charAt(4)});
            int lt = Integer.parseInt(hours) * 60 +  Integer.parseInt(min);
            fullLightTime += lt;
            elementsInStream ++;
        }
        log.info("fullLightTime: " + String.valueOf(fullLightTime));
        int fullTime =  (elementsInStream * 24 * 60);
        double fullNightTime = (double) (fullTime - fullLightTime) /
elementsInStream;

        return fullNightTime;
    }
}
```

```
package com.powerconsumpion.powerconsumptionapplication.dto;


public class Led {
    private String quantity;

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    private String power;
    private String type;
}
```

```
package com.powerconsumpion.powerconsumptionapplication.dto;

import lombok.Data;

//@Data
public class Incandescent {
    private String quantity;
    private String power;
    private String type;

    public String getQuantity() {
```

```
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

```
package com.powerconsumpion.powerconsumptionapplication.dto;

public class Halogen {
    private String quantity;
    private String power;
    private String type;

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

```
package com.powerconsumpion.powerconsumptionapplication.dto;

public class Fluorescent {
    private String quantity;

    public String getQuantity() {
        return quantity;
    }
```

```
    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    private String power;
    private String type;
}
```

```
package com.powerconsumpion.powerconsumptionapplication.dto;

public class ElectricalDevice {
    private String months;
    private Led led;
    private Incandescent incandescent;
    private Fluorescent fluorescent;
    private Halogen halogen;
    private String nameDev;

    public String getMonths() {
        return months;
    }

    public void setMonths(String months) {
        this.months = months;
    }

    public Led getLed() {
        return led;
    }

    public void setLed(Led led) {
        this.led = led;
    }

    public Incandescent getIncandescent() {
        return incandescent;
    }

    public void setIncandescent(Incandescent incandescent) {
        this.incandescent = incandescent;
    }

    public Fluorescent getFluorescent() {
        return fluorescent;
    }

    public void setFluorescent(Fluorescent fluorescent) {
```

```
        this.fluorescent = fluorescent;
    }

    public Halogen getHalogen() {
        return halogen;
    }

    public void setHalogen(Halogen halogen) {
        this.halogen = halogen;
    }

    public String getNameDev() {
        return nameDev;
    }

    public void setNameDev(String nameDev) {
        this.nameDev = nameDev;
    }
}
```

```
package com.powerconsumpion.powerconsumptionapplication.api;

import com.google.gson.Gson;
import com.powerconsumpion.powerconsumptionapplication.addData.dto.Months;
import
com.powerconsumpion.powerconsumptionapplication.api.dto.CalculationResult;
import com.powerconsumpion.powerconsumptionapplication.dto.ElectricalDevice;
import com.powerconsumpion.powerconsumptionapplication.model.ElEffLamp;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import java.sql.*;
import java.util.ArrayList;

@Slf4j
@RestController
@EnableWebMvc
public class LightTimeController {
    private Connection connection;
    private static Statement statement;
    private final static String URL = "jdbc:mysql://localhost:3306";
    private final static String user = "root";
    private final static String pass = "admin";
    private static Gson gson;

    @Bean
    public void connectToDB() {
        try {
            gson = new Gson();
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(URL, user, pass);
            statement = connection.createStatement();
//            getMonthsInfoAboutLightTime();
        } catch (ClassNotFoundException e) {
            log.error("driver trouble: " + e.getMessage());
            throw new RuntimeException(e);
        } catch (SQLException e) {
            log.error("SQL database exception: " + e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    @GetMapping("/home")
    public String helloWorld() {
        return "HelloWorld";
    }

    @GetMapping("/lightTime/{id}")
    public static Months getMonthsInfoAboutLightTime(@PathVariable int id) {
        String Q = "SELECT light_time FROM daylight.months where idmonths =" +
id ;
        Months months = new Months();

        try {
            ResultSet rs = statement.executeQuery(Q);
            ArrayList<String> array = new ArrayList<String>();
            while(rs.next()){
                log.info("add");
                array.add(rs.getString("light_time"));
                log.info(rs.getString("light_time"));
            }

            months.setMonth(String.valueOf(id));

//            Days days = new Days();
//            days.setLightTime(array.iterator().next());
//            log.info("hre: " + days);
            months.setDays(array.iterator().next());
            log.info("1: " +gson.toJson(months));

        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return months;
    }

    @PostMapping(value = "/getCalculation", consumes = "application/json")
    public CalculationResult makeCalculations(@RequestBody ElectricalDevice
electricalDevice) {
        System.out.println(electricalDevice.getNameDev());
        log.info("makeCalculations: " + gson.toJson(electricalDevice));
        String months = electricalDevice.getMonths();
        switch (electricalDevice.getNameDev()) {
            case "calculation":{
                double rs = 0.0;
                ElEffLamp effLamp = new ElEffLamp(months);
                CalculationResult calculationResult = new CalculationResult();
                System.out.println("calc: " +gson.toJson(electricalDevice));

System.out.println(Integer.parseInt(electricalDevice.getIncandescent().getQuanti
ty()));

System.out.println(Double.parseDouble(electricalDevice.getIncandescent().getPowe
r()));
                if (electricalDevice.getIncandescent() != null) {
                    rs +=
effLamp.calculate(Integer.parseInt(electricalDevice.getIncandescent().getQuantit
y()), Double.parseDouble(electricalDevice.getIncandescent().getPower()));
                    log.info("rs: " + rs);
                }
                if (electricalDevice.getLed() != null) {
                    rs +=
effLamp.calculate(Integer.parseInt(electricalDevice.getLed().getQuantity()),
Double.parseDouble(electricalDevice.getLed().getPower()));
                    log.info("rs: " + rs);
```

```
                }
                if (electricalDevice.getHalogen() != null) {
                    rs +=
effLamp.calculate(Integer.parseInt(electricalDevice.getHalogen().getQuantity()),
Double.parseDouble(electricalDevice.getHalogen().getPower()));
                    log.info("rs: " + rs);
                }
                if (electricalDevice.getFluorescent() != null) {
                    rs +=
effLamp.calculate(Integer.parseInt(electricalDevice.getFluorescent().getQuantity
()), Double.parseDouble(electricalDevice.getFluorescent().getPower()));
                    log.info("rs: " + rs);
                }
                calculationResult.setRs(rs);
                return calculationResult;
            }
        }
        return new CalculationResult();
    }

    @GetMapping("/predictNeuralNetwork/{device}")
    public void predictRealDataWithNeuralNetwork(@PathVariable String device) {

        NeuralNetworkController nnc = new NeuralNetworkController(device);
        switch (device) {
            case "bkb_broiler": {
            }
        }
    }

}
```

```
package com.powerconsumpion.powerconsumptionapplication.api.dto;

import lombok.Data;

@Data
public class CalculationResult {
    private double rs;
}
```

```
package com.powerconsumpion.powerconsumptionapplication.addData.dto;

import lombok.Data;

@Data
public class Months {
    private String month;
    private String days;
}
```

```
package com.powerconsumpion.powerconsumptionapplication.addData.dto;

import lombok.Data;

@Data
public class LightTime {
    private String lightTime;
}
```

```
package com.powerconsumpion.powerconsumptionapplication.addData.dto;

import lombok.Data;

@Data
public class Days {
```

```
//    private String lightTime;
    private LightTime lightTime;
}
```

ДОДАТОК Б. ЛІСТИНГ ANDROID ДОДАТКУ

```
package com.example.myapplication;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.SeekBar;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.util.Pair;

import com.google.android.material.datepicker.MaterialDatePicker;

import org.json.JSONException;
import org.json.JSONObject;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;

public class CalculationActivity extends AppCompatActivity {

    private ImageButton calendarBtn;
    private Button calculateConsumptionBtn, calculateSavings;
    private EditText  consumptionResult, dateRange, quantityLedBulb,
quantityIncandescentBulb, quantityHalogenBulb, quantityFluorescentBulb,
savingResult;
    private CheckBox ledCheckBox, incandescentCheckBox, halogenCheckBox,
fluorescentCheckBox;
    private Spinner ledBulbPower, incandescentBulbPower, halogenBulbPower,
fluorescentBulbPower;
    private String rs, rsSavings;
    private SeekBar ledDimmer, incandescentDimmer, halogenDimmer,
fluorescentDimmer;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.calculation_layout);
        initComponents();
    }

    private void initComponents() {
        // initializing all fields in layout
        calendarBtn = findViewById(R.id.calendarBtn);
        calculateConsumptionBtn = findViewById(R.id.calculateConsumptionBtn);
        consumptionResult = findViewById(R.id.consumptionResult);
        savingResult = findViewById(R.id.savingResult);
        dateRange = findViewById(R.id.dateRange);
        quantityLedBulb = findViewById(R.id.quantityLedBulb);
        quantityIncandescentBulb = findViewById(R.id.quantityIncandescentBulb);
        quantityHalogenBulb = findViewById(R.id.quantityHalogenBulb);
```

```
        quantityFluorescentBulb = findViewById(R.id.quantityFluorescentBulb);
        ledCheckBox = findViewById(R.id.ledBulb);
        incandescentCheckBox = findViewById(R.id.incandescentBulb);
        halogenCheckBox = findViewById(R.id.halogenBulb);
        fluorescentCheckBox = findViewById(R.id.fluorescentBulb);
        ledBulbPower = findViewById(R.id.ledBulbPower);
        incandescentBulbPower = findViewById(R.id.incandescentBulbPower);
        halogenBulbPower = findViewById(R.id.halogenBulbPower);
        fluorescentBulbPower = findViewById(R.id.fluorescentBulbPower);
        ledDimmer = findViewById(R.id.ledDimmer);
        incandescentDimmer = findViewById(R.id.incandescentDimmer);
        halogenDimmer = findViewById(R.id.halogenDimmer);
        fluorescentDimmer = findViewById(R.id.fluorescentDimmer);
        calculateSavings = findViewById(R.id.calculateSavings);

        // adding spinner data for selection
        String[] ledBulbAdapterStrings = {"5W", "7W", "9W", "12W", "15W"};
        String[] incandescentBulbAdapterStrings = {"40W", "60W", "75W", "100W"};
        String[] fluorescentBulbAdapterStrings = {"9W", "14W", "19W", "29W"};
        String[] halogenBulbAdapterStrings = {"29W", "43W", "53W", "72W"};

        ArrayAdapter<String> ledAdapter = new ArrayAdapter<>

(this,android.R.layout.simple_spinner_item,ledBulbAdapterStrings);
        ArrayAdapter<String> incandescentAdapter = new ArrayAdapter<>

(this,android.R.layout.simple_spinner_item,incandescentBulbAdapterStrings);
        ArrayAdapter<String> fluorescentAdapter = new ArrayAdapter<>

(this,android.R.layout.simple_spinner_item,fluorescentBulbAdapterStrings);
        ArrayAdapter<String> halogenAdapter = new ArrayAdapter<>

(this,android.R.layout.simple_spinner_item,halogenBulbAdapterStrings);


        ledBulbPower.setAdapter(ledAdapter);
        ledBulbPower.setDropDownHorizontalOffset(400);
        incandescentBulbPower.setAdapter(incandescentAdapter);
        incandescentBulbPower.setDropDownHorizontalOffset(400);
        fluorescentBulbPower.setAdapter(fluorescentAdapter);
        fluorescentBulbPower.setDropDownHorizontalOffset(400);
        halogenBulbPower.setAdapter(halogenAdapter);
        halogenBulbPower.setDropDownHorizontalOffset(400);

        ledBulbPower.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {

            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        calendarBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openCalendarAndSelectDate();
            }
```

```
        });

        calculateConsumptionBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                JSONObject json = prepareDataBeforeSending();

                ConsumptionCalculator consumptionCalculator = new
ConsumptionCalculator(json);
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        rs = consumptionCalculator.calculate();
                        consumptionResult.setText(rs + " kW-hour");
                    }
                }).start();
            }
        });

        calculateSavings.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                JSONObject jsonObject = prepareDataBeforeSending();

                ConsumptionCalculator consumptionCalculator = new
ConsumptionCalculator(jsonObject);

                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        rsSavings = consumptionCalculator.calculate();
                        rsSavings = String.valueOf(Double.parseDouble(rs)-
Double.parseDouble(rsSavings));

                        double percentOfSave =
(Double.parseDouble(rsSavings)/Double.parseDouble(rs));

                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                savingResult.setText("Your savings around " +
rsSavings + ", which is " + percentOfSave * (double) 100 + "%");
                            }
                        });

                    }
                }).start();

            }
        });

        ledDimmer.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            int progressChangedValue = 0;
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
                progressChangedValue = progress;
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
```

```
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                Toast.makeText(CalculationActivity.this, "Seek bar progress is
:" + progressChangedValue,
                        Toast.LENGTH_SHORT).show();
            }
        });

        incandescentDimmer.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            int progressChangedValue = 0;
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
                progressChangedValue = progress;
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                Toast.makeText(CalculationActivity.this, "Seek bar progress is
:" + progressChangedValue,
                        Toast.LENGTH_SHORT).show();
            }
        });

        halogenDimmer.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            int progressChangedValue = 0;
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
                progressChangedValue = progress;
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                Toast.makeText(CalculationActivity.this, "Seek bar progress is
:" + progressChangedValue,
                        Toast.LENGTH_SHORT).show();
            }
        });

        fluorescentDimmer.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            int progressChangedValue = 0;
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
                progressChangedValue = progress;
            }
```

```
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                Toast.makeText(CalculationActivity.this, "Seek bar progress is
:" + progressChangedValue,
                        Toast.LENGTH_SHORT).show();
            }
        });
    }

    private JSONObject prepareDataBeforeSending() {
        ArrayList<String> listOfBulbs = new ArrayList<>();
        if (!checkLedParams()) {
            System.out.println("there is no led lamp");
        } else {
            listOfBulbs.add("led");
        }
        if (!checkIncandescentParams()){
            System.out.println("there is no led lamp");
        } else {
            listOfBulbs.add("incandescent");
        }
        if (!checkFluorescentParams()) {
            System.out.println("there is no led lamp");
        } else {
            listOfBulbs.add("fluorescent");
        }
        if (!halogenLedParams()) {
            System.out.println("there is no led lamp");
        } else {
            listOfBulbs.add("halogen");
        }
        JSONObject json = new JSONObject();
        try {
            json.put("months", dateRange.getText().toString().replaceAll("/",
".."));
            json.put("nameDev", "calculation");
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        for (String type : listOfBulbs) {
            try {
                JSONObject jsonObject = new JSONObject();
                jsonObject.put("type", type);
                switch (type) {
                    case "led": {
                        jsonObject.put("quantity", quantityLedBulb.getText());
                        if (ledDimmer.getProgress() == 100) {
                            jsonObject.put("power",
String.valueOf(Double.parseDouble(ledBulbPower.getSelectedItem().toString().repl
ace("W", ""))));
                        } else {
                            double power =
Double.parseDouble(ledBulbPower.getSelectedItem().toString().replace("W", ""));
                            double savePower = power * (double)
ledDimmer.getProgress() / 100.0;
                            System.out.println("savingPower: " + savePower);
                            jsonObject.put("power", String.valueOf(savePower));
```

```
                        }

                        json.put("led", jsonObject);
                        break;
                    }
                    case "incandescent": {
                        jsonObject.put("quantity",
quantityIncandescentBulb.getText());
                        if (incandescentDimmer.getProgress() == 100) {
                            jsonObject.put("power",
String.valueOf(Double.parseDouble(incandescentBulbPower.getSelectedItem().toStri
ng().replace("W", "")));
                        } else {
                            double power =
Double.parseDouble(incandescentBulbPower.getSelectedItem().toString().replace("W
", ""));
                            double savePower = power * (double)
incandescentDimmer.getProgress() / 100.0;
                            System.out.println("savingPower: " + savePower);
                            jsonObject.put("power", String.valueOf(savePower));
                        }
                        json.put("incandescent", jsonObject);
                        break;
                    }
                    case "fluorescent": {
                        jsonObject.put("quantity",
quantityFluorescentBulb.getText());
                        if (fluorescentDimmer.getProgress() == 100) {
                            jsonObject.put("power",
String.valueOf(Double.parseDouble(fluorescentBulbPower.getSelectedItem().toStrin
g().replace("W", "")));
                        } else {
                            double power =
Double.parseDouble(fluorescentBulbPower.getSelectedItem().toString().replace("W"
, ""));
                            double savePower = power * (double)
fluorescentDimmer.getProgress() / 100.0;
                            System.out.println("savingPower: " + savePower);
                            jsonObject.put("power", String.valueOf(savePower));
                        }
                        json.put("fluorescent", jsonObject);
                        break;
                    }
                    case "halogen": {
                        jsonObject.put("quantity",
quantityHalogenBulb.getText());
                        if (halogenDimmer.getProgress() == 100) {
                            jsonObject.put("power",
String.valueOf(Double.parseDouble(halogenBulbPower.getSelectedItem().toString().
replace("W", "")));
                        } else {
                            double power =
Double.parseDouble(halogenBulbPower.getSelectedItem().toString().replace("W",
""));
                            double savePower = power * (double)
halogenDimmer.getProgress() / 100.0;
                            System.out.println("savingPower: " + savePower);
                            jsonObject.put("power", String.valueOf(savePower));
                        }
                        json.put("halogen", jsonObject);
                        break;
                    }
                }
```

```
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
        }
        return json;
    }

    private boolean halogenLedParams() {
        if (halogenCheckBox.isChecked() &&
!quantityHalogenBulb.getText().equals("") &&
!halogenBulbPower.getSelectedItem().equals("")) {
            return true;
        } else {
            return false;
        }
    }

    private boolean checkFluorescentParams() {
        if (fluorescentCheckBox.isChecked() &&
!quantityFluorescentBulb.getText().equals("") &&
!fluorescentBulbPower.getSelectedItem().equals("")) {
            return true;
        } else {
            return false;
        }
    }

    private boolean checkIncandescentParams() {
        if (incandescentCheckBox.isChecked() &&
!quantityIncandescentBulb.getText().equals("") &&
!incandescentBulbPower.getSelectedItem().equals("")) {
            return true;
        } else {
            return false;
        }
    }

    private boolean checkLedParams() {
        if (ledCheckBox.isChecked() && !quantityLedBulb.getText().equals("") &&
!ledBulbPower.getSelectedItem().equals("")) {
            return true;
        } else {
            return false;
        }
    }

    private void openCalendarAndSelectDate() {
        // Creating a MaterialDatePicker builder for selecting a date range
        MaterialDatePicker.Builder<androidx.core.util.Pair<Long, Long>> builder
= MaterialDatePicker.Builder.dateRangePicker();
        builder.setTitleText("Select a date range");

        // Building the date picker dialog
        MaterialDatePicker<Pair<Long, Long>> datePicker = builder.build();
        datePicker.addOnPositiveButtonClickListener(selection -> {

            // Retrieving the selected start and end dates
            Long startDate = selection.first;
            Long endDate = selection.second;

            // Formating the selected dates as strings
            SimpleDateFormat sdf = new SimpleDateFormat("MM/yyyy",
Locale.getDefault());
```

```
            String startDateString = sdf.format(new Date(startDate));
            String endDateString = sdf.format(new Date(endDate));

            // Creating the date range string
            String selectedDateRange = startDateString + "-" + endDateString;

            // Displaying the selected date range in the TextView
            dateRange.setText(selectedDateRange);
        });

        // Showing the date picker dialog
        datePicker.show(getSupportFragmentManager(), "DATE_PICKER");
    }

    @Override
    protected void onStart() {
        super.onStart();
    }

    @Override
    protected void onStop() {
        super.onStop();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

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

```
package com.example.myapplication;

import com.example.myapplication.dto.ElectricalDevice;
import com.google.gson.Gson;
import com.squareup.okhttp.MediaType;
import com.squareup.okhttp.OkHttpClient;
import com.squareup.okhttp.Request;
import com.squareup.okhttp.RequestBody;
import com.squareup.okhttp.Response;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.logging.Logger;

import lombok.extern.slf4j.Slf4j;

public class ConsumptionCalculator {
    private JSONObject jsonArray;
    private Gson gson = new Gson();

    public static final MediaType JSON = MediaType.parse("application/json;
charset=utf-8");
```

```
    public ConsumptionCalculator(JSONObject jsonArray) {
        this.jsonArray = jsonArray;
        System.out.println(jsonArray.toString());
    }


    public String calculate() {
        System.out.println("calculate");
        try {
            String months = jsonArray.get("months").toString();
            String startMonth = months.substring(0, months.indexOf("-"));
            String endMonth = months.substring(months.indexOf("-") +1);
            System.out.println("startMonth: " + startMonth);
            System.out.println("endMonth: " + endMonth);

            if ((startMonth.charAt(0) == endMonth.charAt(0)) &&
(startMonth.charAt(1) == endMonth.charAt(1))) {
                System.out.println("we need to calculate for 3rd month");
                jsonArray.put("months", startMonth.charAt(1));
                System.out.println(jsonArray.toString());

                // make an http request to server
            } else {
                System.out.println("we need to calculate for more than 1
month");
                jsonArray.put("months", startMonth.charAt(1) + "-" +
endMonth.charAt(1));
                System.out.println(jsonArray.toString());
            }

            ElectricalDevice electricalDevice =
gson.fromJson(String.valueOf(jsonArray), ElectricalDevice.class);
            System.out.println(gson.toJson(electricalDevice));


            // make an http request to server

            OkHttpClient ohc = new OkHttpClient();
            String url = "http://10.0.2.2:8181/getCalculation";

            RequestBody requestBody = RequestBody.create(JSON,
gson.toJson(electricalDevice));

            Request request = new Request.Builder()
                    .url(url)
                    .addHeader("Content-Type", "application/json")
//                    .addHeader("Accept", "application/json")
                    .post(requestBody)
                    .build();

            System.out.println("make http request with requestBody: " +
requestBody.contentType() + "\n" + request.body().toString());
            Response response = ohc.newCall(request).execute();
            JSONObject responseObj = new JSONObject(response.body().string());

            if (response.code() == 200) {
                return responseObj.get("rs").toString();
            }

        } catch (Exception e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
```

```
            System.out.println(e.getLocalizedMessage());
        }
        return "something went wrong";
    }
}
```

```
package com.example.myapplication.dto;

public class ElectricalDevice {
    private String months;
    private Led led;
    private Incandescent incandescent;
    private Fluorescent fluorescent;
    private Halogen halogen;
    private String nameDev;

    public String getMonths() {
        return months;
    }

    public void setMonths(String months) {
        this.months = months;
    }

    public Led getLed() {
        return led;
    }

    public void setLed(Led led) {
        this.led = led;
    }

    public Incandescent getIncandescent() {
        return incandescent;
    }

    public void setIncandescent(Incandescent incandescent) {
        this.incandescent = incandescent;
    }

    public Fluorescent getFluorescent() {
        return fluorescent;
    }

    public void setFluorescent(Fluorescent fluorescent) {
        this.fluorescent = fluorescent;
    }

    public Halogen getHalogen() {
        return halogen;
    }

    public void setHalogen(Halogen halogen) {
        this.halogen = halogen;
    }

    public String getNameDev() {
        return nameDev;
    }

    public void setNameDev(String nameDev) {
        this.nameDev = nameDev;
    }
}
```

```
package com.example.myapplication.dto;

import lombok.Data;

//@Data
public class Fluorescent {
    private String quantity;

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    private String power;
    private String type;
}
```

```
package com.example.myapplication.dto;

import lombok.Data;

//@Data
public class Halogen {
    private String quantity;
    private String power;
    private String type;

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    public String getType() {
        return type;
```

```
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

```
package com.example.myapplication.dto;

import lombok.Data;

//@Data
public class Incandescent {
    private String quantity;
    private String power;
    private String type;

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

```
package com.example.myapplication.dto;

import lombok.Data;

//@Data
public class Led {
    private String quantity;

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPower() {
        return power;
    }

    public void setPower(String power) {
        this.power = power;
    }
```

```
    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    private String power;
    private String type;
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:layout_marginTop="20dp"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="20dp"
                android:textColor="@color/black"
                android:layout_gravity="center"
                android:text="Choose your equipment"
                android:layout_marginBottom="20dp"/>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:layout_marginStart="10dp"
                android:layout_marginEnd="10dp"
                android:weightSum="0.9">

                <CheckBox
                    android:id="@+id/ledBulb"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:text="LED Bulb">

                </CheckBox>

                <EditText
                    android:id="@+id/quantityLedBulb"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:inputType="number"
                    android:hint="Quantity">

                </EditText>
```

```
    <Spinner
        android:id="@+id/ledBulbPower"
        android:layout_width="wrap_content"
        android:gravity="center"
        android:layout_height="wrap_content"
        android:hint="Power">

    </Spinner>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginStart="10dp"
    android:layout_marginEnd="10dp"
    android:weightSum="0.9">

    <CheckBox
        android:id="@+id/incandescentBulb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Incandescent Bulb">

    </CheckBox>

    <EditText
        android:id="@+id/quantityIncandescentBulb"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:hint="Quantity">
    </EditText>

    <Spinner
        android:id="@+id/incandescentBulbPower"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:hint="Power">

    </Spinner>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginStart="10dp"
    android:layout_marginEnd="10dp"
    android:weightSum="0.9">

    <CheckBox
        android:id="@+id/halogenBulb"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Halogen Bulb">
```

```
        </CheckBox>

        <EditText
            android:id="@+id/quantityHalogenBulb"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="number"
            android:hint="Quantity">

        </EditText>

        <Spinner
            android:id="@+id/halogenBulbPower"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:hint="Power">

        </Spinner>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginStart="10dp"
        android:layout_marginEnd="10dp"
        android:weightSum="0.9">

        <CheckBox
            android:id="@+id/fluorescentBulb"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Fluorescent Bulb">

        </CheckBox>

        <EditText
            android:id="@+id/quantityFluorescentBulb"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="number"
            android:hint="Quantity">

        </EditText>

        <Spinner
            android:id="@+id/fluorescentBulbPower"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:hint="Power">

        </Spinner>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
```

```
    android:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Choose date range for calculation"
        android:textSize="20dp"
        android:textColor="@color/black">

    </TextView>


</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:gravity="center"
    android:orientation="horizontal">

    <EditText
        android:id="@+id/dateRange"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="    Date range   ">

    </EditText>

    <ImageButton
        android:id="@+id/calendarBtn"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:background="@color/white"
        android:padding="10dp"
        android:layout_marginEnd="15dp"
        android:scaleType="centerInside"
        android:src="@drawable/calendar">
    </ImageButton>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="15dp"
    android:layout_marginEnd="15dp">

    <Button
        android:id="@+id/calculateConsumptionBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/black"
        android:text="@string/CalculateConsumption">

    </Button>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout height="wrap content"
    android:layout_marginStart="15dp"
    android:layout_marginEnd="15dp"
    android:layout_marginTop="20dp">
```

```
    <EditText
        android:id="@+id/consumptionResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:hint="Result of Calculation">
    </EditText>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="15dp"
    android:layout_marginEnd="15dp"
    android:layout_marginTop="20dp">

    <TextView
        android:layout_width="match_parent"
        android:layout height="wrap content"
        android:text="Set up dimmer for calculation savings"
        android:textSize="20dp"
        android:textColor="@color/black"
        android:gravity="center">
    </TextView>
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="horizontal"
        android:weightSum="1">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="15dp"
            android:layout_marginStart="10dp"
            android:textColor="@color/black"
            android:layout_weight="0.3"
            android:text="LED dimmer">

        </TextView>


        <SeekBar
            android:id="@+id/ledDimmer"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:progress="100"
            android:layout_weight="0.7" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
```

```
    android:orientation="horizontal"
    android:weightSum="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15dp"
        android:layout_marginStart="10dp"
        android:textColor="@color/black"
        android:layout_weight="0.1"
        android:text="Incandescent dimmer">

    </TextView>


    <SeekBar
        android:id="@+id/incandescentDimmer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:progress="100"
        android:layout_weight="0.9" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:orientation="horizontal"
    android:weightSum="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15dp"
        android:layout_marginStart="10dp"
        android:textColor="@color/black"
        android:text="Halogen dimmer"
        android:layout_weight="0.225">

    </TextView>


    <SeekBar
        android:id="@+id/halogenDimmer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:progress="100"
        android:layout_weight="0.775" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:orientation="horizontal"
    android:weightSum="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="15dp"
        android:layout_marginStart="10dp"
        android:textColor="@color/black"
```

```
                android:text="Fluorescent dimmer"
                android:layout_weight="0.15">

            </TextView>

            <SeekBar
                android:id="@+id/fluorescentDimmer"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:progress="100"
                android:layout_weight="0.85" />

        </LinearLayout>

        <EditText
            android:id="@+id/savingResult"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:inputType="none"
            android:textSize="15dp"
            android:hint="Saving Result">

        </EditText>

        <Button
            android:id="@+id/calculateSavings"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:layout_marginEnd="15dp"
            android:text="calculate savings"
            android:backgroundTint="@color/black">

        </Button>

    </LinearLayout>
</LinearLayout>
</ScrollView>
</LinearLayout>
```

ДОДАТОК В. КОД НЕЙРОМЕРЕЖІ

```
@Test
public void main2() throws Exception {

        File file = new File("C:\\Users\\Vasia\\Desktop\\маг\\data\\test.csv");
        SequenceRecordReader reader = new CSVSequenceRecordReader(0, ";"); //
Перший аргумент - індекс стовпця з мітками, другий - роздільник
        reader.initialize(new FileSplit(file));

        // Читання даних та їхнє перетворення в нормалізовані DataSet об'єкти
        List<List<Writable>> sequence;
        List<DataSet> data = new ArrayList<>();

        while (reader.hasNext()) {
            sequence = reader.sequenceRecord();
            INDArray features = Nd4j.create(sequence.size(), 1,1); // Кількість
ознак (тут: 1)
            INDArray labels = Nd4j.create(sequence.size(), 1,1); // Кількість
міток (тут: 1)
            for (int i = 0; i < sequence.size(); i++) {
                DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy H:mm");
                LocalDateTime dateTime =
LocalDateTime.parse(sequence.get(i).get(0).toString(), formatter);
                if (Double.parseDouble(sequence.get(i).get(1).toString()) !=
0.0) {
                    features.putScalar(new int[]{i, 0, 0},
dateTime.toEpochSecond(ZoneOffset.UTC)); // Перший стовпець - це час
                    labels.putScalar(new int[]{i, 0, 0},
Double.parseDouble(sequence.get(i).get(1).toString())); // Другий стовпець - це
електроспоживання
                }
            }
            data.add(new DataSet(features, labels));
        }
//        System.out.println(data);
        int miniBatchSize = 32;
        DataSetIterator iterator = new ListDataSetIterator<>(data,
miniBatchSize);

        // Нормалізація даних
//        DataNormalization normalizer = new NormalizerMinMaxScaler(0, 1);
//        normalizer.fit(iterator);
//        for (DataSet dataSet : data) {
//            normalizer.transform(dataSet);
//        }

        // Конфігурація моделі LSTM

        // Створення та налаштування моделі
        MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
                .seed(123)
                .updater(new Adam())
                .weightInit(WeightInit.RELU)
                .list()
                .layer(new LSTM.Builder()
                        .nIn(1)
                        .nOut(50)
                        .dropOut(0.2)
                        .activation(Activation.SIGMOID)
                        .build())
                .layer(new LSTM.Builder()
```

```
                        .nIn(50)
                        .nOut(100)
                        .dropOut(0.2)
                        .activation(Activation.SIGMOID)
                        .build())
                .layer(new LSTM.Builder()
                        .nIn(100)
                        .nOut(50)
                        .dropOut(0.2)
                        .activation(Activation.SIGMOID)
                        .build())
                .layer(new
RnnOutputLayer.Builder(LossFunctions.LossFunction.MSE)
                        .activation(Activation.IDENTITY)
                        .nIn(50)
                        .nOut(1)
                        .build())
                .build();

        MultiLayerNetwork net = new MultiLayerNetwork(conf);
        net.init();
        net.setListeners(new ScoreIterationListener(50));

        // Навчання моделі
        int numEpochs = 500;
        for (int i = 0; i < numEpochs; i++) {
            net.fit(iterator);
            iterator.reset();
            System.gc();
        }

        DataSetIterator testIterator = new ListDataSetIterator<>(data, 10); //
Міні-пакет розміром 1
        RegressionEvaluation eval = new RegressionEvaluation();
        while (testIterator.hasNext()) {
            DataSet next = testIterator.next();
            INDArray features = next.getFeatures();
            INDArray labels = next.getLabels();
            INDArray predicted = net.output(features);
            eval.eval(labels, predicted);
        }
        System.out.println(eval.stats());

        File locationToSave = new
File("C:\\Users\\Vasia\\Desktop\\маг\\data\\testNeuralModel.zip");
        ModelSerializer.writeModel(net, locationToSave, true);

    }
```

```
@Test
public void cheq() throws Exception  {
    // Шлях до моделі
    File modelFile = new
File("C:\\Users\\Vasia\\Desktop\\маг\\data\\testNeuralModel.zip");
    MultiLayerNetwork model =
ModelSerializer.restoreMultiLayerNetwork(modelFile);

    // Шлях до нових тестових даних
    File testFile = new
File("C:\\Users\\Vasia\\Desktop\\маг\\data\\predictiontest.csv");
    SequenceRecordReader testReader = new CSVSequenceRecordReader(0, ";");
    testReader.initialize(new FileSplit(testFile));

    // Підготовка тестових даних
```

```
    List<List<Writable>> sequence;
    Set<LocalDateTime> uniqueDateTimes = new HashSet<>(); // Множина для
зберігання унікальних дат
    List<DataSet> testData = new ArrayList<>();
    List<Double> predictedValues = new ArrayList<>(); // Список для зберігання
передбачених значень



    while (testReader.hasNext()) {
        sequence = testReader.sequenceRecord();
        List<LocalDateTime> dateTimes = new ArrayList<>();
        List<Double> consumptions = new ArrayList<>();
        for (List<Writable> record : sequence) {
            DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy H:mm");
            LocalDateTime dateTime =
LocalDateTime.parse(record.get(0).toString(), formatter);
            double consumption = Double.parseDouble(record.get(1).toString());
            if (consumption != 0.0 && uniqueDateTimes.add(dateTime)) { //
Додавання унікальних дат до множини
                dateTimes.add(dateTime);
                consumptions.add(consumption);
            }
        }

        if (!dateTimes.isEmpty()) {
            INDArray features = Nd4j.create(dateTimes.size(), 1, 1);
            INDArray labels = Nd4j.create(dateTimes.size(), 1, 1);
            for (int i = 0; i < dateTimes.size(); i++) {
                features.putScalar(new int[]{i, 0, 0},
dateTimes.get(i).toEpochSecond(ZoneOffset.UTC));
                labels.putScalar(new int[]{i, 0, 0}, consumptions.get(i));
            }
            testData.add(new DataSet(features, labels));
        }
    }

    if (testData.isEmpty()) {
        System.out.println("No valid data found for testing.");
        return;
    }

    // Прогнозування на основі тестових даних
    for (DataSet dataSet : testData) {
        INDArray features = dataSet.getFeatures();
        INDArray lastTimeStep =
features.get(NDArrayIndex.interval(features.size(0) - 1, features.size(0)),
NDArrayIndex.all(), NDArrayIndex.all());
        long lastTimestamp = (long) lastTimeStep.getDouble(0, 0); // Останній
часовий крок
        for (int i = 0; i < 30; i++) { // Прогнози після закінчення файлу для
наступних 10 хвилин
            INDArray predicted = model.rnnTimeStep(lastTimeStep);

            // Отримання часу для наступного прогнозу (час останнього + 1
хвилина)
            LocalDateTime nextTime = LocalDateTime.ofEpochSecond(lastTimestamp +
60 * (i + 1), 0, ZoneOffset.UTC);

            // Оновлення вхідних даних для наступного прогнозу
            lastTimeStep = predicted; // Використовуємо передбачене значення як
```

```
вхід для наступного кроку

            double predictedConsumption = predicted.getDouble(0, 0, 0);
            predictedValues.add(predictedConsumption);
            System.out.println("Predicted at " +
nextTime.format(DateTimeFormatter.ofPattern("dd.MM.yyyy H:mm")) + ": " +
predictedConsumption);
        }
        model.rnnClearPreviousState();
    }

    // Прогнозування на основі тестових даних
    RegressionEvaluation evaluation = new RegressionEvaluation();
    for (DataSet dataSet : testData) {
        INDArray features = dataSet.getFeatures();
        INDArray labels = dataSet.getLabels();
        INDArray predicted = model.output(features);

        // Оновлення оцінки на основі передбачених і істинних значень
        evaluation.eval(labels, predicted);

        // Друкуємо передбачені та істинні значення
        System.out.println("True Values: " + labels);
        System.out.println("Predicted Values: " + predicted);
    }

    // Вивід статистики оцінки
    System.out.println("Evaluation stats: " + evaluation.stats());

    // Створення графіка
    XYSeries testSeries = new XYSeries("Останні значення тесту");
    for (int i = 0; i < 30; i++) {
        testSeries.add(i + 1, testData.get(testData.size() -
1).getLabels().getDouble(i, 0, 0));
        System.out.println();
    }

    XYSeries predictedSeries = new XYSeries("Передбачені значення");
    for (int i = 0; i < predictedValues.size(); i++) {
        predictedSeries.add(i + 1, predictedValues.get(i));
    }

    XYSeriesCollection dataset = new XYSeriesCollection();
    dataset.addSeries(testSeries);
    dataset.addSeries(predictedSeries);

    JFreeChart chart = ChartFactory.createXYLineChart(
            "Тестові та передбачені значення", // Заголовок графіку
            "Часові кроки", // Підпис осі X
            "Значення", // Підпис осі Y
            dataset, // Дані для графіку
            PlotOrientation.VERTICAL,
            true, // Легенда
            true, // Включення взаємодії
            false // Включення URL
    );

    // Збереження графіку в PNG
    saveChartAsPNG(chart, "C:\\Users\\Vasia\\Desktop\\маг\\data\\graph.png");

    // Створюємо панель для відображення графіку
    ChartPanel chartPanel = new ChartPanel(chart);
```

```
    JFrame frame = new JFrame("Графік");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLayout(new BorderLayout());
    frame.add(chartPanel, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
}

    // Метод для збереження графіку в PNG
    private static void saveChartAsPNG(JFreeChart chart, String filePath) {
        int width = 800;
        int height = 600;
        File file = new File(filePath);
        try {
            ChartUtils.saveChartAsPNG(file,chart,width,height);
            System.out.println("Графік збережено в " + filePath);
        } catch (IOException e) {
            System.err.println("Помилка при збереженні графіку: " +
e.getMessage());
        }
    }
```