МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій

Кафедра комп’ютерної інженерії та електромеханіки

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему:

Система збору та аналізу даних споживання електроенергії у приміщенні

Рівень вищої освіти другий (магістерський)

Спеціальності 123 «Комп’ютерна інженерія»

Освітня програма «Комп’ютерна інженерія»

Виконав: студент групи МгКІ-23.

Бугаєнко Б. А.

Керівник: к.т.н., доц. Стаценко Д.В.

Рецензент.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(науковий ступінь, вчене звання, прізвище та ініціали)

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій .

Кафедра комп’ютерної інженерії та електромеханіки .

Спеціальність 123 «Комп’ютерна Інженерія».

Освітня програма «Комп’ютерна Інженерія»

**ЗАТВЕРДЖУЮ**

Завідувач кафедри КІЕМ

\_\_\_\_\_\_\_\_\_\_\_\_ Дмитро СТАЦЕНКО

“\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 року

**ЗАВДАННЯ**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Бугаєнко Богдан Андрійович.

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи: Система збору та аналізу даних споживання електроенергії у приміщенні

.

Науковий керівник роботи \_Стаценко Дмитро Володимирович, к.т.н., доцент . (прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

.

затверджені наказом вищого навчального закладу від 03.09.2024 № 188-уч.

2. Вихідні дані до кваліфікаційної роботи: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

3. Зміст кваліфікаційної роботи (перелік питань, які потрібно опрацювати): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_2024

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної роботи | Терміни виконання етапів | Примітка про виконання |
| 1 | Вступ | 09.09.2024 |  |
| 2 | РОЗДІЛ 1. [Огляд існуючих технологій моніторингу енергоспоживання](#_20kjqz8j3zjb) | 20.09.2024 |  |
| 3 | РОЗДІЛ 2. [Типи сенсорів та засоби інтеграції з telegram ботом](#_nsiv3gniv12u) | 05.10. 2024 |  |
| 4 | РОЗДІЛ 3. [Розробка програмної частини для аналізу та візуалізації даних](#_mqzv94yy517j) | 25.10.2024 |  |
| 5 | Висновки | 28.10.2024 |  |
| 6 | Оформлення (чистовий варіант) | 31.10.2024 |  |
| 7 | Подача кваліфікаційної роботи науковому керівнику для відгуку | 01.11.2024 |  |
| 8 | Подача кваліфікаційної роботи для рецензування (за 14 днів до захисту) | 09.11.2024 |  |
| 9 | Перевірка кваліфікаційної роботи на наявність ознак плагіату та текстових співпадінь (за 10 днів до захисту) | 11.11.2024 |  |
| 10 | Подання кваліфікаційної роботи на завідувачу кафедри (за 7днів до захисту) | 18.11.2024 |  |

З завданням ознайомлений:

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_** Богдан Бугаєнко.

( підпис ) (Власне ім’я та ПРІЗВИЩЕ)

Науковий керівник \_\_\_\_\_\_\_\_\_\_\_\_ Дмитро СТАЦЕНКО.

( підпис ) (Власне ім’я та ПРІЗВИЩЕ)

### АНОТАЦІЯ

**Бугаєнко Б. А. Система збору та аналізу даних споживання електроенергії у приміщенні. - Рукопис.**

Кваліфікаційна робота за спеціальністю 123 “Комп’ютерна інженерія”, освітньою програмою “Комп’ютерна Інженерія”. - Київський національний університет технологій та дизайну, Київ, 2024 рік.

Кваліфікаційна робота присвячена розробці інноваційної системи збору та аналізу даних споживання електроенергії у приміщенні з інтеграцією в Telegram бот. Метою дослідження є забезпечення ефективного моніторингу енергоспоживання та надання користувачу зручного інструменту для управління та аналізу даних безпосередньо через месенджер. У роботі досліджуються сучасні технології збору даних, зокрема, використання смарт-сенсорів, мікроконтролерів і технологій передачі даних в реальному часі. Telegram бот розробляється як ключовий компонент, що забезпечує взаємодію між системою збору даних та користувачем, пропонуючи функції для аналізу, прогнозування та порівняння енергоспоживання за допомогою команд “*/analyze”*, “*/forecast”*, “*/compare”*. Окрім того, система підтримує інтерактивні елементи управління через клавіатуру бота та автоматичні сповіщення для надання користувачу рекомендацій щодо ефективного використання енергії. В експериментальній частині проведено тестування системи у реальних умовах приміщення, що дозволило оцінити її точність, ефективність та зручність використання для кінцевого користувача.

*Ключові слова: споживання електроенергії, Telegram бот, моніторинг, сенсори, аналіз даних.*

**ABSTRACT**

**B. A. Bugayenko. System of collection and analysis of indoor electricity consumption data. - Manuscript.**

Qualification work in the specialty 123 "Computer engineering", educational program "Computer Engineering". - Kyiv National University of Technology and Design, Kyiv, 2024.

This qualification thesis is dedicated to the development of an innovative system for collecting and analyzing electricity consumption data in a room with integration into a Telegram bot. The aim of the research is to provide efficient energy consumption monitoring and offer the user a convenient tool for data management and analysis directly through a messenger. The study explores modern data collection technologies, including the use of smart sensors, microcontrollers, and real-time data transmission techniques. The Telegram bot is designed as a core component that ensures interaction between the data collection system and the user, offering functions for consumption analysis, forecasting, and comparison through commands such as “*/analyze”*, “*/forecast”*, and “*/compare”*. Additionally, the system supports interactive control elements via the bot's keyboard and automatic notifications to provide the user with recommendations for optimizing energy usage. The experimental part includes system testing in real room conditions, evaluating its accuracy, efficiency, and user-friendliness for the end-user.

*Keywords: electricity consumption, Telegram bot, monitoring, sensors, data analysis.*

[ВСТУП 1](#_210r5v53wl70)

[РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ МОНІТОРИНГУ ЕНЕРГОСПОЖИВАННЯ 5](#_20kjqz8j3zjb)

[1.1. Класифікація побутових та комерційних приладів для збору даних про споживання енергії 5](#_huivmo7vpf4v)

[1.2. Сучасні підходи до інтеграції систем моніторингу з месенджерами для оповіщення та взаємодії з користувачем 15](#_i0du4kag4r7s)

[Висновки до розділу 1 19](#_5n0uiflvsagl)

[РОЗДІЛ 2. ТИПИ СЕНСОРІВ ТА ЗАСОБИ ІНТЕГРАЦІЇ З TELEGRAM БОТОМ 20](#_nsiv3gniv12u)

[2.1. Типи сенсорів та засобів контролю енергоспоживання 20](#_fkomr5ac2ijq)

[2.1.1. Лічильники електроенергії 20](#_ut526qkosxdp)

[2.1.2. Датчики споживання електроенергії 21](#_3y6c8ai0am8l)

[2.1.3. Смарт-розетки 23](#_spdlis52y57x)

[2.2. Інтеграція з Telegram API для передачі даних користувачу 24](#_lxnkoqap36y6)

[2.3. Модуль збору та обробки даних із використанням мікроконтролерів 29](#_u3hwpa3k3vf0)

[2.3.1. Використання мікроконтролерів для моніторингу 29](#_ieonlmq6s33o)

2[.3.2. Архітектура підключення 31](#_1yr21puoq2n5)

[Висновок до розділу 2 38](#_9ejzljh9w3ht)

[РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ДЛЯ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ 40](#_mqzv94yy517j)

[3.1. Зберігання, обробка та скидання даних 40](#_xfxjzbmrgchq)

[3.1.1. Зберігання та обробка даних 40](#_8n4zvqfrc0x8)

[3.1.2. Скидання даних 40](#_7oxs1jeaoccs)

[3.2. Аналіз, візуалізація та звітність 41](#_6gk5xrukufty)

[3.2.1. Команда “/analyze”: Аналіз даних та візуалізація 41](#_uj7lsmyj3of)

[3.2.2. Команда “/report”. Звіт даних 43](#_tr8e0j33tahq)

[3.2.3. Команда “/statistics”. Статистика даних 45](#_suyblyjc7f3t)

[3.3 Команди прогнозування та аналізу 47](#_2n4soae11467)

[3.3.1. Команда “/forecast”. Прогнозування 47](#_880bnyvfqik0)

[3.3.2. Команда “/period\_analysis”. Аналіз споживання 49](#_y0tg83p9qp7w)

[3.3.3. Порівняння даних за місяцями 51](#_5nuezskaia7l)

[3.4 Інтерактивна взаємодія з користувачем та експорт даних 52](#_i4i9ccpelhcv)

[3.4.1. Обробка команд та нагадування через Telegram. Реалізація нагадувань з використанням asyncio 52](#_c8ae4s1yky7o)

[3.4.2. Експорт CSV файлу 53](#_xzw28027lcwi)

[3.4.3 Команда “/help” 54](#_x9iqt8j8cyqq)

[3.5. Аналіз і розрахунок споживання 55](#_jfghog7icxgf)

[Висновки до розділу 3 57](#_g3gyghxcrlr1)

[ВИСНОВКИ 58](#_t12wkhtz9mma)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 59](#_69lugb6hfz6m)

[ДОДАТОК А 62](#_w9nyq2253cim)

### ВСТУП

В умовах стрімкого зростання споживання електроенергії та підвищення вимог до енергоефективності у побутовому та комерційному секторах постає гостра потреба у впровадженні інноваційних систем для контролю та оптимізації енергоспоживання. Енергетичні ресурси стають дедалі дорожчими, що спонукає до розробки нових рішень для моніторингу використання електроенергії та скорочення втрат. Однією з найважливіших задач є створення систем, які дозволяють не тільки відслідковувати споживання електроенергії, але й аналізувати дані для прийняття ефективних управлінських рішень.

Одним із перспективних напрямків вирішення цієї задачі є використання технологій Інтернету речей (IoT), зокрема інтеграція смарт-пристроїв для моніторингу енергоспоживання з сучасними засобами комунікації. Однією з найбільш доступних і зручних платформ для інтерактивної взаємодії з користувачем є Telegram, що дозволяє впроваджувати технології автоматизації та моніторингу на рівні звичайного месенджера, забезпечуючи зручний інтерфейс для взаємодії з користувачем у режимі реального часу.

**Актуальність теми** пов’язана з потребою підвищення енергоефективності у приміщеннях шляхом впровадження систем автоматизованого моніторингу та аналізу даних споживання електроенергії. Невід'ємною складовою таких систем є забезпечення можливості швидкого реагування на зміну рівнів енергоспоживання та надання користувачу оперативної інформації щодо споживання ресурсів. Telegram бот у цій системі виконує роль посередника між користувачем та системою збору даних, надаючи зручний інтерфейс для отримання аналітики, прогнозів та рекомендацій.

Сучасні системи моніторингу енергоспоживання зазвичай включають в себе великий набір функцій: від контролю за споживанням окремих приладів до комплексних систем керування будинком або офісом. Проте більшість таких рішень орієнтовані на обмежене коло спеціалістів і не завжди зручні для пересічних користувачів. Впровадження простих у використанні засобів контролю та аналізу через такі платформи, як Telegram, дозволяє значно спростити взаємодію з системою і підвищити ефективність прийняття рішень щодо економії електроенергії.

**Метою роботи** є розробка інтегрованої системи збору, аналізу та візуалізації даних про споживання електроенергії у приміщенні з використанням Telegram бота. Така система повинна забезпечити автоматизований моніторинг, швидкий доступ до даних та можливість керування параметрами енергоспоживання через зручний інтерфейс.

Для досягнення цієї мети у роботі ставляться наступні задачі:

* Провести аналіз сучасних технологій для збору даних про споживання електроенергії, їхнього функціонування та інтеграції з комунікаційними платформами.
* Розробити архітектуру системи збору даних на основі смарт-сенсорів для моніторингу енергоспоживання.
* Інтегрувати систему збору даних з Telegram ботом для взаємодії користувача із системою в режимі реального часу.
* Реалізувати функціонал бота для аналізу даних про споживання електроенергії: прогнозування, порівняння, візуалізація та інші аналітичні інструменти.
* Провести експериментальні дослідження ефективності роботи системи у реальних умовах для оцінки її точності та зручності використання.

**Об’єктом дослідження** є процес збору, обробки та аналізу даних споживання електроенергії у приміщенні.

**Предметом дослідження** є система збору даних на основі смарт-сенсорів із можливістю інтеграції з Telegram ботом для надання аналітичної інформації користувачу у реальному часі.

**Методи дослідження.** Для вирішення поставлених задач у роботі використано такі методи:

* Теоретичний аналіз літератури та сучасних рішень у галузі моніторингу та контролю за споживанням електроенергії, а також методів інтеграції IoT пристроїв з комунікаційними платформами.
* Моделювання процесу збору та обробки даних на основі смарт-сенсорів та мікроконтролерів з використанням програмного забезпечення для створення Telegram бота.
* Експериментальні дослідження, спрямовані на тестування розробленої системи у реальних умовах та оцінку її точності, ефективності та зручності використання.

**Наукова новизна** роботи полягає у розробці інтегрованої системи збору та аналізу даних споживання електроенергії з використанням Telegram бота як засобу для взаємодії з користувачем. Система поєднує в собі технології Інтернету речей (IoT), автоматизації та сучасних методів аналізу даних для забезпечення максимальної енергоефективності. Telegram бот дозволяє користувачу керувати системою через зручний інтерфейс і отримувати інформацію у реальному часі.

**Практичне значення** роботи полягає в тому, що розроблена система може бути використана як у побутових, так і в комерційних приміщеннях для оптимізації споживання електроенергії. Telegram бот дозволяє спростити процес управління енергоспоживанням, надаючи користувачу доступ до аналітичних інструментів, прогнозування та рекомендацій щодо економії електроенергії. Крім того, система може бути легко адаптована для впровадження у системи смарт-будинків або великі енергетичні мережі.

**Структура роботи.** Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглядаються сучасні технології моніторингу енергоспоживання та інтеграція з комунікаційними платформами. У другому розділі розроблено архітектуру системи збору даних, описано сенсори та засоби інтеграції з Telegram ботом. Третій розділ присвячений створенню Telegram бота, тестуванню системи та аналізу її роботи.[8]

### РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ МОНІТОРИНГУ ЕНЕРГОСПОЖИВАННЯ

#### **1.1. Класифікація побутових та комерційних приладів для збору даних про споживання енергії**

Сучасні системи енергоменеджменту стають все більш інтегрованими і доступними для кінцевих користувачів. Завдяки розвитку технологій Інтернету речей (IoT), з'явилися численні смарт-рішення, які дозволяють здійснювати точний контроль за споживанням електроенергії, як на рівні окремих приладів, так і на рівні всього будинку або офісу.

Ці пристрої можуть використовуватися як у побутових, так і в комерційних умовах, для забезпечення зручності користувачів та ефективного контролю над енергоспоживанням. Основні класи пристроїв:

**Смарт-лічильники** – це інноваційні пристрої, що дозволяють автоматизувати процес збору даних про споживання енергії та передавати ці дані у реальному часі до централізованих систем контролю. Основна перевага цих лічильників полягає в усуненні необхідності ручного зчитування показників. Завдяки використанню різних протоколів передачі даних, таких як Wi-Fi, Zigbee, GSM або LoRa, смарт-лічильники можуть інтегруватися у домашні та комерційні мережі, забезпечуючи простий і ефективний контроль над енергоспоживанням. Ці пристрої дають можливість споживачам отримувати детальну інформацію про поточний стан мережі, відстежувати пікові навантаження та ідентифікувати прилади, що споживають найбільше енергії. Дані від таких лічильників можуть використовуватися не лише для аналізу, а й для автоматичного керування пристроями, як-от вимкнення обладнання у позапіковий час для зниження витрат.

Смарт-лічильники також підтримують можливість інтеграції з додатками та платформами, що працюють на основі IoT. Це означає, що користувач може керувати споживанням електроенергії дистанційно через мобільний додаток або месенджери, наприклад, Telegram. Подібні рішення дозволяють оперативно отримувати звіти, сповіщення про аномальне споживання та порівнювати витрати між різними періодами. Такий підхід сприяє оптимізації енергетичних витрат та підвищує обізнаність користувачів про їхні енергетичні звички. Деякі системи, як-от Sense або Emporia Vue, навіть здатні використовувати штучний інтелект для розпізнавання підписів конкретних пристроїв у загальному енергоспоживанні, що дає можливість краще контролювати споживання.



Рис. 1.1 – смарт-лічильник

Однією з важливих особливостей смарт-лічильників є можливість тарифікації за часом доби, коли ціна на електроенергію залежить від навантаження на мережу. Наприклад, електроенергія може бути дешевшою вночі, що стимулює користувачів переносити частину свого споживання на ці періоди. Окрім того, смарт-лічильники допомагають енергетичним компаніям виявляти та усувати втрати енергії чи крадіжки. Завдяки функціям моніторингу у реальному часі та передбаченню навантажень, постачальники можуть ефективніше управляти енергетичними ресурсами.

Для впровадження таких рішень часто використовуються IoT-платформи на зразок **Kaa IoT** або **EcoStruxure** від Schneider Electric, які дозволяють створювати індивідуальні рішення для бізнесу чи побутових потреб. Проте, незважаючи на переваги, впровадження смарт-лічильників може стикатися з труднощами, такими як висока вартість обладнання та залежність від стабільного інтернет-з'єднання. Деякі користувачі також висловлюють занепокоєння щодо захисту своїх персональних даних через постійний моніторинг. Тим не менш, з огляду на загальну тенденцію до діджиталізації та зростання попиту на енергоефективні рішення, очікується, що смарт-лічильники будуть ставати все більш популярними як у побутових, так і в комерційних умовах.

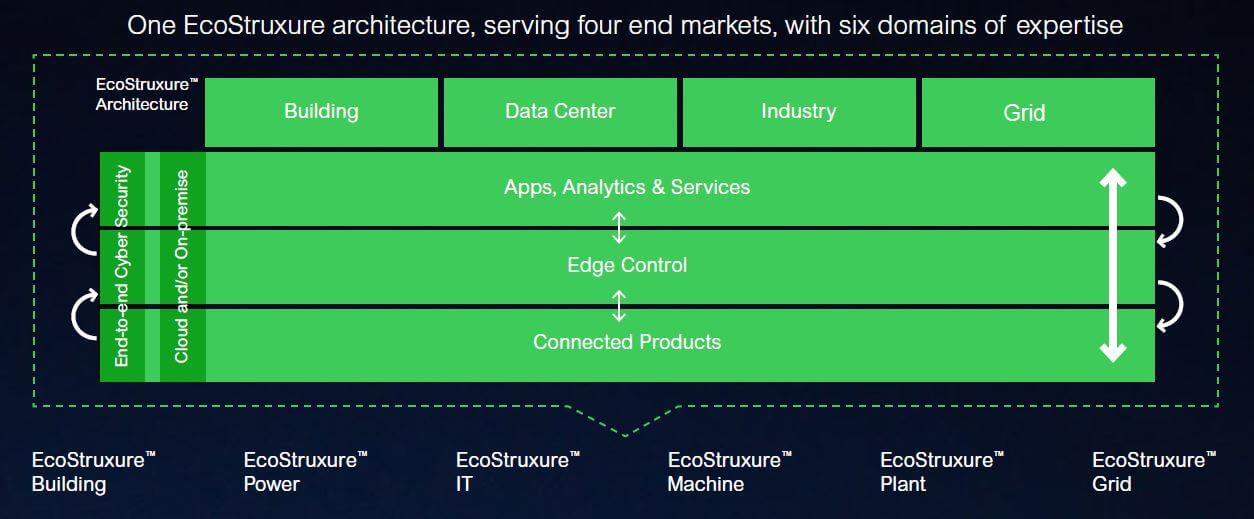


Рис. 1.2 – ІоТ-платформа

У промислових об'єктах ці лічильники використовуються у складі систем енергоменеджменту (EMS) для централізованого контролю та оптимізації роботи обладнання. Вони дозволяють аналізувати ефективність використання енергії та розробляти сценарії для автоматичного вимкнення непотрібних приладів, знижуючи таким чином загальні витрати. Важливо, що сучасні лічильники здатні працювати з декількома протоколами передачі даних одночасно, що спрощує інтеграцію у вже існуючі мережі.

**Смарт-розетки** є важливим елементом сучасних систем моніторингу та управління енергоспоживанням. Вони забезпечують гнучкий контроль над приладами, які підключаються до електромережі, дозволяючи дистанційно керувати їхньою роботою та відстежувати споживання електроенергії в режимі реального часу. Це особливо корисно для оптимізації енергоспоживання як у побутових, так і в комерційних умовах. Смарт-розетки можуть інтегруватися з мобільними додатками або платформами на базі IoT, такими як Google Home, Alexa чи Telegram боти, надаючи користувачам можливість керувати пристроями з будь-якої точки світу.

Ці пристрої надають низку переваг, серед яких миттєвий контроль над окремими приладами. Наприклад, розумна розетка дозволяє користувачу дистанційно увімкнути або вимкнути прилад, що підключений до неї, через мобільний додаток або голосову команду. Завдяки цьому можна автоматизувати роботу приладів за розкладом, зменшуючи непотрібне споживання енергії вночі чи під час відсутності вдома. Наприклад, холодильник або водонагрівач можуть працювати лише в години низького тарифу на електроенергію, що дозволяє заощаджувати кошти. Крім того, користувачі можуть отримувати звіти про споживання електроенергії за певний період і аналізувати, які саме пристрої споживають найбільше.

Смарт-розетки також відіграють важливу роль у підвищенні безпеки. Наприклад, вони можуть автоматично вимикати прилади у разі перевищення встановлених лімітів споживання або при виявленні аномалій, що може запобігти перегріву та пожежам. Це особливо актуально для побутових приладів із високим рівнем енергоспоживання, таких як обігрівачі або електродуховки. Окрім цього, розетки дозволяють встановлювати обмеження на час роботи приладів, що допомагає уникати нераціонального використання електроенергії.

З точки зору інтеграції, сучасні смарт-розетки можуть працювати з популярними протоколами, такими як Wi-Fi, Zigbee та Z-Wave. Це забезпечує їхню сумісність з різними системами розумного будинку, а також з централізованими платформами для енергоменеджменту (EMS). Деякі моделі навіть підтримують віддалене оновлення програмного забезпечення, що дозволяє додавати нові функції без необхідності заміни обладнання. Наприклад, у комерційних умовах такі розетки можуть бути корисними для автоматизації роботи офісної техніки чи освітлення, що знижує витрати на електроенергію та покращує загальну енергоефективність.

Ще однією цікавою функцією є інтеграція з месенджерами, такими як Telegram. Завдяки цьому можна налаштувати бота, який буде надсилати повідомлення про стан підключених приладів або нагадування про необхідність вимкнути певний пристрій. Це підвищує зручність управління та дозволяє вчасно реагувати на проблеми або перевищення лімітів споживання.



Рис. 1.3 – смарт-розетка Belkin

Популярні моделі смарт-розеток, як-от TP-Link Kasa Smart та Belkin WeMo Insight, забезпечують інтуїтивно зрозумілий інтерфейс та просту інтеграцію з іншими компонентами розумного будинку. Окрім цього, такі пристрої часто мають підтримку голосових асистентів, як-от Amazon Alexa або Google Assistant, що додатково підвищує зручність їх використання.

**Системи енергоменеджменту (EMS)** є комплексними рішеннями, що забезпечують централізований моніторинг та управління енергоспоживанням у будівлях, підприємствах або виробничих об'єктах. Вони дозволяють оптимізувати використання енергії шляхом автоматизації та аналізу даних у режимі реального часу, що сприяє зменшенню витрат та підвищенню енергоефективності.



Рис. 1.4 – система енергоменеджменту

Однією з ключових функцій EMS є збір даних з різних джерел енергоспоживання (лічильники, датчики, розетки тощо) та їх централізоване оброблення. Використовуючи отримані дані, система здатна відслідковувати ефективність використання енергії, виявляти перевантаження та непотрібне використання обладнання. Аналітика EMS може рекомендувати користувачу найкращі практики з оптимізації, наприклад, перенесення частини споживання на періоди з нижчим тарифом або автоматичне вимкнення приладів під час простою.

Завдяки інтеграції з IoT-платформами EMS здатні працювати з великим обсягом даних у хмарі та надавати користувачам детальні звіти та прогнозування енергоспоживання. Такі платформи, як **Schneider Electric EcoStruxure** або **Siemens Desigo**, пропонують повністю інтегровані рішення, що охоплюють усі аспекти енергетичного менеджменту — від управління електропостачанням до моніторингу роботи вентиляційних, кліматичних і освітлювальних систем.

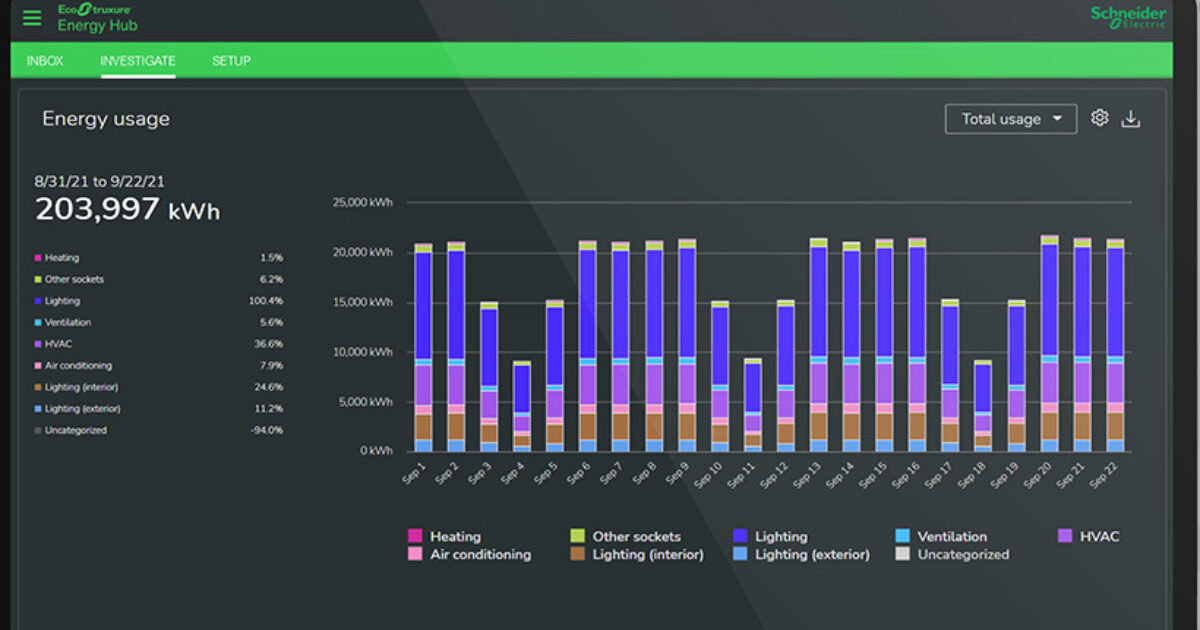


Рис. 1.5 – Schneider Electric EcoStruxure

Інша важлива функція EMS — це планування та оптимізація пікових навантажень. За допомогою прогнозування система може запобігати перевантаженням мережі та забезпечувати рівномірний розподіл енергоресурсів. Це особливо актуально для підприємств, де велике навантаження може призвести до високих штрафів за перевищення договірних лімітів споживання електроенергії.

Сучасні EMS дозволяють не лише аналізувати дані, а й надсилати сповіщення або виконувати автоматизовані дії. Наприклад, при виявленні аномального споживання система може автоматично вимкнути відповідний прилад або надіслати попередження користувачу через Telegram бота. Це забезпечує швидке реагування на проблеми та мінімізує можливі втрати.

Інтеграція з мобільними додатками та месенджерами спрощує контроль за енергоспоживанням навіть у віддаленому режимі. Користувачі можуть отримувати щоденні звіти про споживання або встановлювати ліміти для різних відділів чи обладнання, що сприяє більш раціональному використанню ресурсів.[11]

Попри всі переваги, впровадження EMS може вимагати значних початкових інвестицій, оскільки система включає в себе не лише програмне забезпечення, а й широкий спектр апаратних компонентів, таких як лічильники, контролери та сенсори. Крім того, для ефективної роботи EMS необхідна стабільна інтернет-інфраструктура та регулярні оновлення програмного забезпечення. Однак у довгостроковій перспективі ці інвестиції виправдовуються завдяки значному зниженню енергетичних витрат та підвищенню ефективності роботи обладнання.

**Датчики енергоспоживання** – це невеликі пристрої, які можуть бути встановлені в різних частинах системи електропостачання. Вони відстежують різні параметри електрики, як-от напруга, струм та потужність, і передають дані до системи аналізу.

Датчики енергоспоживання є ключовими елементами систем моніторингу та управління електроенергією, що використовуються як у побутових, так і промислових умовах. Ці пристрої допомагають відстежувати різні параметри електричного потоку, включаючи напругу, силу струму, потужність і частоту. Отримані дані дозволяють аналізувати споживання енергії у реальному часі, виявляти перевантаження та втрати в системі, а також прогнозувати майбутні потреби.

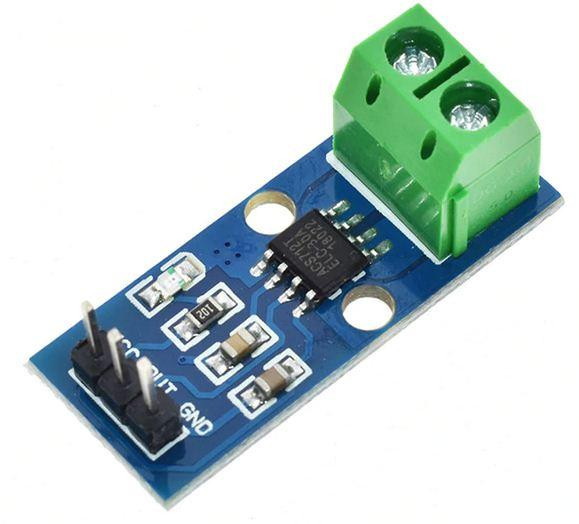


Рис. 1.6 – датчик струму ACS712

Датчики можна поділити на кілька основних типів. Наприклад, сенсори струму (ACS712) вимірюють силу струму, яка проходить через електричну лінію, і передають ці дані на контролер. Ці сенсори часто використовуються для моніторингу окремих пристроїв або ліній живлення, щоб визначити їхню ефективність. Ще одним популярним рішенням є датчики напруги (ZMPT101B), які допомагають відстежувати зміни напруги у мережі та попереджати про небезпечні коливання.

Інтеграція таких датчиків із мікроконтролерами, як-от Arduino або ESP32, дозволяє автоматизувати збір та обробку даних. Через MQTT-протокол або інші мережеві технології ці дані можуть передаватися на сервери або мобільні додатки для подальшого аналізу. Важливою функцією є можливість передачі даних до месенджерів, таких як Telegram, де бот може інформувати користувачів про стан мережі чи споживання окремих приладів.

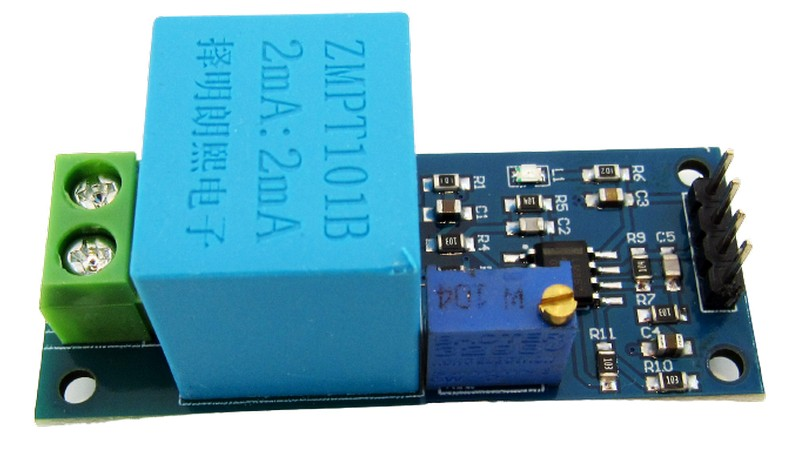


Рис. 1.7 – датчик напруги ZMPT101B

Переваги використання датчиків енергоспоживання включають підвищення точності моніторингу та попередження про несправності у мережі. Наприклад, вони можуть виявляти раптові стрибки струму, які можуть свідчити про зношеність обладнання або можливий короткий замик. Завдяки таким попередженням користувач може своєчасно відреагувати та запобігти аваріям.

Окрім побутового використання, датчики широко застосовуються у промислових умовах. Вони допомагають оптимізувати роботу обладнання та знижувати витрати шляхом автоматичного вимкнення приладів у непотрібний час або регулювання інтенсивності роботи в залежності від поточного навантаження. У великих будівлях та підприємствах вони стають частиною більш комплексних систем енергоменеджменту (EMS), де поєднуються з іншими сенсорами для моніторингу клімату, освітлення та безпеки.

Попри всі переваги, інтеграція таких рішень може стикатися з певними труднощами, як-от вартість обладнання та необхідність належної інфраструктури для передачі даних. Однак у довгостроковій перспективі використання датчиків енергоспоживання виправдовує себе завдяки зниженню експлуатаційних витрат і підвищенню ефективності використання електроенергії.[1]

#### **1.2. Сучасні підходи до інтеграції систем моніторингу з месенджерами для оповіщення та взаємодії з користувачем**

З розвитком технологій автоматизації та збільшенням кількості інтернет-підключених пристроїв, великого значення набуває можливість інтеграції систем збору даних із засобами комунікації, зокрема з месенджерами, які є поширеними платформами для взаємодії з користувачами. Telegram бот у такій системі виступає як інтерфейс для користувача, через який можна взаємодіяти з системою у зручному форматі чату.

1. **Передача даних у режимі реального часу**. Месенджери можуть забезпечити миттєву передачу даних про енергоспоживання користувачу, дозволяючи оперативно реагувати на зміни. Користувач може отримувати регулярні сповіщення про споживання електроенергії або про перевищення лімітів, які були встановлені.
2. **Інтерактивні команди для управління енергоспоживанням**. Telegram бот може підтримувати команди, які дозволяють користувачу керувати системою збору даних. Наприклад, за допомогою команд “*/start”*(початок роботи з ботом), “*/analyze”*(отримання аналітики за певний період), “*/forecast”*(прогноз споживання на основі минулих даних), “*/compare”*(порівняння показників споживання за різні періоди), користувач може отримувати аналітичні дані, прогнози споживання на основі історичних даних та порівнювати показники споживання за різні періоди.
3. **Автоматизація сповіщень**. Завдяки інтеграції з месенджерами система може автоматично повідомляти користувачів про зміни в споживанні енергії, порушення режиму або відхилення від норми. Сповіщення можуть бути налаштовані за допомогою спеціальних команд, як-от “*/set\_reminder\_interval”*, які задають частоту та формат повідомлень.
4. **Аналіз даних та їх візуалізація**. Telegram бот може виступати не тільки як інструмент для отримання інформації, але й як засіб для її аналізу та візуалізації. Дані можуть відображатися у вигляді графіків, діаграм або таблиць, що допомагає користувачу краще зрозуміти тенденції споживання і зробити відповідні висновки для оптимізації використання енергії.

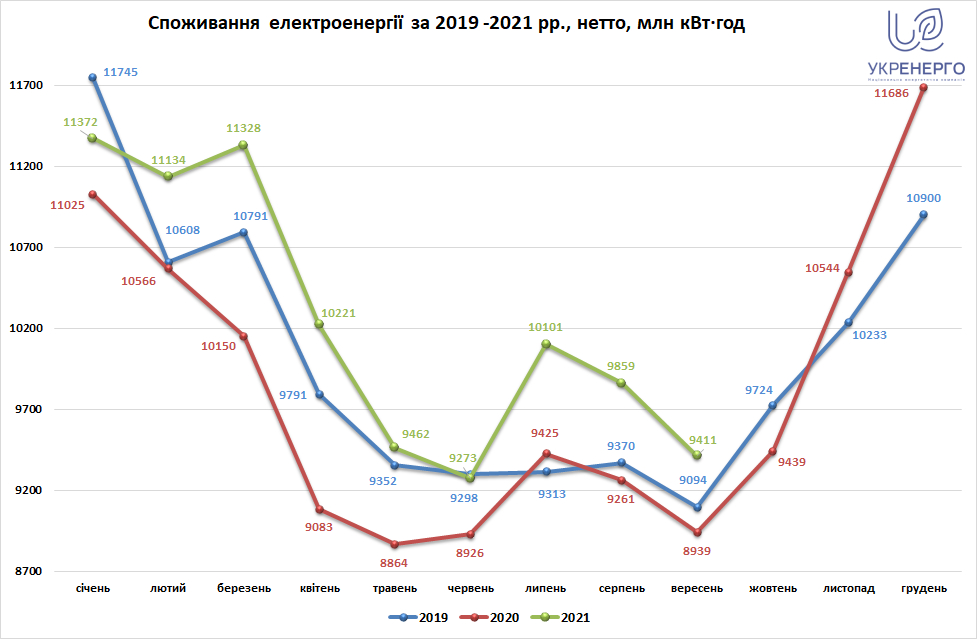


Рис. 1.8 – графік споживання електроенергії

1. **Інтеграція з системами смарт-будинків**. Telegram бот може бути частиною більшого комплексу автоматизації будинку, де збір даних про енергоспоживання інтегрований з іншими системами управління – освітленням, опаленням, кондиціюванням тощо. Це дозволяє користувачу отримувати комплексні дані про стан будинку та керувати ним з одного місця. Наприклад, за допомогою бота можна автоматично вимикати світло або зменшувати потужність обігрівача, якщо датчики показують, що у кімнаті нікого немає.

Сучасні технології Інтернету речей (IoT) та інтелектуальних систем моніторингу дедалі більше орієнтуються на інтеграцію із месенджерами, що дає змогу користувачам здійснювати контроль за енергоспоживанням дистанційно та взаємодіяти з системами у режимі реального часу. Одним з найпопулярніших інструментів для цього є Telegram, який пропонує зручний інтерфейс і потужні можливості для розробки ботів, здатних передавати інформацію, надавати сповіщення та виконувати команди користувача у зручному форматі чату. Важливою перевагою такої інтеграції є можливість автоматичної діагностики: система не лише передає дані про споживання, але й здатна виявляти аномалії у споживанні та надсилати користувачу звіт про можливі причини перевантаження, за потреби ініціюючи діагностику обладнання. Це особливо актуально для складних систем, таких як розумні будинки та офісні комплекси, де одночасно функціонує багато пристроїв.

Інтеграція з Telegram надає широкі можливості для персоналізації інтерфейсу взаємодії. Користувач може налаштовувати команди, які відповідають його потребам, а також створювати сценарії керування енергоспоживанням. Наприклад, для різних зон або пристроїв в будинку можна налаштувати сценарії автоматичного вимкнення в нічний час або встановити окремі сповіщення для конкретних приладів. Це дозволяє системі працювати в автоматичному режимі та значно знижує витрати на енергоресурси, особливо в умовах, де енергоспоживання коливається залежно від часу доби чи сезону.

Застосування штучного інтелекту (ШІ) та машинного навчання (ML) у таких інтегрованих системах значно підвищує ефективність використання енергоресурсів. Telegram бот, обробляючи історичні дані про енергоспоживання, може прогнозувати майбутнє споживання і таким чином рекомендувати користувачам оптимальні сценарії використання приладів. Крім того, алгоритми ШІ дають змогу формулювати поради, що допомагають користувачам знижувати витрати шляхом адаптації системи під їхні звички. Наприклад, якщо бот фіксує, що певний пристрій використовується переважно у піковий час, він може запропонувати користувачу перенести його роботу на періоди з низьким тарифом. Крім того, враховуючи зовнішні фактори, такі як температура зовнішнього середовища або погодні умови, система може змінювати свої рекомендації для досягнення максимальної ефективності.

Не менш важливим є питання безпеки та конфіденційності даних, особливо в умовах, коли Telegram бот надає користувачу доступ до управління життєво важливими параметрами системи. Система Telegram API забезпечує безпечну взаємодію завдяки шифруванню повідомлень та функціям двофакторної аутентифікації, але розробники також можуть реалізувати додаткові рівні безпеки, обмежуючи доступ до команд або налаштовуючи повідомлення для користувача про спроби несанкціонованого доступу. Також корисним може бути налаштування певних правил доступу, таких як віддалений контроль тільки з певних IP-адрес або в певний час доби.

Telegram API також є потужним інструментом для масштабованих рішень, оскільки дозволяє створювати інтеграції, здатні ефективно функціонувати в умовах великих систем. Це корисно в офісних комплексах або комерційних приміщеннях, де можна розподіляти доступ до даних між користувачами залежно від їхніх зон відповідальності, що знижує витрати та підвищує контроль над ресурсами. За необхідності Telegram бот можна об'єднувати з іншими каналами зв’язку, такими як електронна пошта або SMS, для забезпечення максимальної надійності сповіщень.

Окрім цього, бот Telegram легко інтегрується в комплексні системи розумного будинку, що дозволяє використовувати його для управління не лише енергоспоживанням, але й іншими функціями — наприклад, освітленням, опаленням, вентиляцією. Це забезпечує максимальну зручність для користувачів, які можуть контролювати всі параметри будинку через один інтерфейс, що дозволяє ще більше оптимізувати витрати на енергію, підвищити рівень автоматизації та зробити будинок більш екологічно чистим.

Таким чином, інтеграція систем моніторингу з Telegram не тільки значно спрощує управління енергоспоживанням, але й надає можливість отримувати оперативну аналітику та прогнози. Це дозволяє користувачу адаптувати свої звички під оптимальні енерговикористання і контролювати витрати з максимальною ефективністю.[13]

##### Висновки до розділу 1

Проведено аналіз існуючих побутових та комерційних приладів для збору даних про споживання енергії. Розглянуто такі технології, як смарт-лічильники, розумні розетки, датчики споживання, та системи енергоменеджменту.

Цей розділ підсумовує, що сучасні технології моніторингу споживання електроенергії спрямовані на автоматизацію та інтеграцію з комунікаційними платформами, що робить процес управління більш зручним та ефективним. Telegram бот є ідеальним рішенням для забезпечення доступу до даних у реальному часі та управління споживанням електроенергії через інтерактивний інтерфейс.

### РОЗДІЛ 2. ТИПИ СЕНСОРІВ ТА ЗАСОБИ ІНТЕГРАЦІЇ З TELEGRAM БОТОМ

##### 2.1. Типи сенсорів та засобів контролю енергоспоживання

##### 2.1.1. Лічильники електроенергії

Лічильники електроенергії є ключовими елементами для моніторингу та керування енергоспоживанням, особливо в умовах інтеграції з IoT-технологіями та платформами, такими як Telegram API. Ці пристрої вимірюють споживання електроенергії та дозволяють автоматизувати процес збору та обробки даних у реальному часі. Використання лічильників із можливістю передачі даних через Wi-Fi,Zigbee або LoRa робить їх ефективним рішенням як для побутових, так і для промислових потреб. Інтеграція таких пристроїв з Telegram API забезпечує користувачам доступ до аналітики через зручний інтерфейс у месенджері, що дозволяє контролювати витрати та реагувати на проблеми в режимі реального часу.

У такій системі лічильник електроенергії може безпосередньо передавати дані до сервера або мікроконтролера (наприклад, ESP32), який виконує роль посередника для передачі інформації в Telegram бот. Бот отримує ці дані та надає користувачу актуальні звіти, графіки та сповіщення. Наприклад, команда “*/summary”* може вивести звіт про споживання за обраний період, а “*/predict”* — передбачити майбутнє споживання на основі історичних даних. Завдяки такій інтеграції користувач може оперативно відстежувати споживання, коригувати поведінку та уникати перевитрат.

Крім того, можливість налаштовувати автоматичні нагадування через Telegram (наприклад, через команду “*/schedule”*) дозволяє підтримувати контроль за споживанням навіть у відсутності користувача. Це особливо важливо для великих підприємств, де перевищення лімітів може призвести до штрафних санкцій.

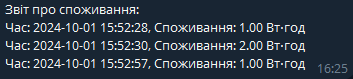


Рис. 2.1 – “*/summary”*

Telegram API також дозволяє вбудовувати інтерактивні кнопки, що спрощують взаємодію з системою: наприклад, можна вмикати та вимикати пристрої через команди бота.



Рис. 2.2 – “*/schedule”*

Завдяки цим можливостям лічильники стають частиною комплексної системи енергоменеджменту, що поєднує апаратне забезпечення та програмні інтерфейси. Така інтеграція забезпечує не тільки моніторинг, але й аналітику в режимі реального часу, допомагаючи користувачам оптимізувати споживання енергії та знижувати витрати.

##### 2.1.2. Датчики споживання електроенергії

Датчики струму та напруги підключаються до окремих ліній чи приладів, щоб контролювати їх енергоспоживання. Такі сенсори часто використовуються для:

* Моніторингу окремих приладів: кондиціонерів, бойлерів, кухонних пристроїв.
* Виявлення відхилень: датчики можуть надсилати сповіщення про несправності або аномальне споживання, що допомагає уникнути перевантажень і аварій.

Датчики споживання електроенергії відіграють ключову роль у моніторингу енергетичних ресурсів, забезпечуючи точний збір даних про струм, напругу та потужність. Ці пристрої дозволяють як побутовим, так і промисловим користувачам відстежувати споживання енергії на рівні окремих приладів або цілих мереж. Завдяки інтеграції з платформами на базі IoT та API, такими як Telegram, користувачі можуть отримувати звіти та сповіщення в режимі реального часу.



Рис. 2.3 – ESP32

Датчики струму, як-от ACS712, вимірюють силу струму в лінії та передають ці дані на мікроконтролери, наприклад, ESP32 або Arduino. Вони широко використовуються для моніторингу окремих пристроїв, що дозволяє ідентифікувати обладнання з високим рівнем споживання або перевитратами. Датчики напруги, як-от ZMPT101B, допомагають вимірювати рівень напруги та попереджають про коливання, які можуть впливати на стабільність роботи обладнання. Зібрані дані можуть автоматично надходити до Telegram бота для аналізу та візуалізації.

Ці сенсори інтегруються з протоколами MQTT або HTTP для передачі даних на сервер або в хмару. На основі отриманої інформації Telegram бот може надсилати користувачу звіти про стан мережі, повідомлення про перевищення лімітів та рекомендації щодо оптимізації. Наприклад, якщо датчик виявляє різке зростання споживання, бот може попередити користувача про можливе несправне обладнання або перевантаження. Крім цього, користувач може вмикати чи вимикати пристрої через Telegram бот, налаштовувати ліміти та розклад роботи приладів для зменшення витрат.

Завдяки такій інтеграції датчики споживання електроенергії забезпечують гнучке управління та підвищують енергоефективність. Вони дозволяють відслідковувати аномалії в споживанні та знижувати витрати шляхом оптимізації роботи приладів. Важливим є також підвищення безпеки: у разі виявлення аномальних значень датчики можуть автоматично відключати відповідні пристрої або надсилати сповіщення користувачам. Це робить такі рішення особливо корисними для підприємств та великих будівель, де необхідно постійно контролювати навантаження та уникати перевищень.[16]

##### 2.1.3. Смарт-розетки

Смарт-розетки дозволяють контролювати енергоспоживання на рівні кожного підключеного пристрою та дистанційно вимкнути або вмикати його через мобільний додаток чи бот Telegram. Приклади використання: дистанційне керування освітленням, електронагрівачами або зарядними пристроями. Смарт-розетки часто інтегруються у IoT екосистеми та підтримують такі протоколи як Wi-Fi або Zigbee​.

Інтеграція з Telegram ботом дозволяє користувачам керувати пристроями безпосередньо з месенджера, надсилаючи команди для увімкнення чи вимкнення розетки, а також отримуючи звіти про споживання. Наприклад, бот може повідомляти про те, що прилад спожив близький до встановленого ліміту обсяг електроенергії, або нагадувати про необхідність його вимкнути для економії. Така автоматизація сприяє зменшенню зайвого використання електроенергії та допомагає підтримувати контроль над енергетичними витратами навіть у віддаленому режимі.

Смарт-розетки також підвищують безпеку, оскільки здатні автоматично вимикати підключені пристрої у разі виявлення перевантаження або перегріву. Це особливо корисно для таких енергомістких приладів, як обігрівачі чи пральні машини, що знижує ризики виникнення пожежі чи несправностей. Деякі моделі розеток також підтримують оновлення прошивки через інтернет, що дає можливість додавати нові функції без необхідності заміни обладнання.



Рис. 2.4 – Інтеграція з Telegram ботом

У поєднанні з системами енергоменеджменту смарт-розетки дозволяють автоматизувати роботу приладів у великих будівлях або офісах, зменшуючи витрати та підвищуючи загальну енергоефективність. Вони допомагають не тільки в моніторингу, але й у прогнозуванні споживання, що дозволяє оптимізувати ресурси відповідно до потреб користувачів та змін у тарифах на електроенергію.[15]

##### 2.2. Інтеграція з Telegram API для передачі даних користувачу

Telegram API дозволяє розробникам створювати ботів для інтерактивної взаємодії з користувачами, що робить його зручним інструментом для інтеграції з системами моніторингу.

Як відбувається передача даних:

1. Бот отримує дані: Лічильник або датчик відправляє дані на сервер.
2. Сервер обробляє дані: Дані передаються на хмарний сервер для аналізу.
3. API викликає бота: Telegram бот через Webhook або polling отримує ці дані та відправляє користувачу повідомлення у чаті.

Telegram бот також може надсилати графіки та таблиці для візуалізації енергоспоживання.

Інтеграція з Telegram API відкриває можливості для передачі даних та автоматизованої взаємодії між користувачами та системами моніторингу енергоспоживання. Telegram API дозволяє створювати ботів, які можуть надсилати повідомлення, приймати команди, передавати файли та інтерактивні графіки у зручному форматі. Це рішення стає особливо корисним у системах енергоменеджменту, де потрібно постійно відслідковувати стан мережі, отримувати звіти в реальному часі та оперативно реагувати на зміни у споживанні.

Telegram боти використовуються як інтерфейс для передачі даних, дозволяючи користувачам отримувати аналітичні звіти про споживання та керувати пристроями безпосередньо через месенджер. За допомогою Python та бібліотеки python-telegram-bot розробники можуть інтегрувати бот з датчиками та лічильниками, які збирають і обробляють інформацію про електроенергію. Наприклад, користувачі можуть надсилати команди на зразок “*/summary”* для отримання звіту про споживання за певний період або “*/schedule”* для налаштування інтервалів нагадувань.

Однією з ключових переваг Telegram API є підтримка асинхронної взаємодії. Це означає, що дані з датчиків можуть автоматично передаватися до бота через Webhook або polling, і користувачі отримуватимуть актуальну інформацію без затримок. Наприклад, при виявленні аномалій у споживанні, бот може миттєво повідомляти користувача, що дає змогу оперативно вживати заходів для уникнення перевантаження мережі або надмірного споживання.

Telegram API також підтримує інтерактивні елементи, такі як клавіатури та кнопки, що полегшує навігацію для користувачів. Це дозволяє створювати зручні інтерфейси, через які можна дистанційно керувати пристроями: увімкнути або вимкнути прилад, переглянути поточне споживання чи змінити ліміт. Окрім цього, бот може автоматично надсилати графіки споживання, які генеруються за допомогою бібліотек Python, таких як Matplotlib, що підвищує наочність аналізу даних.

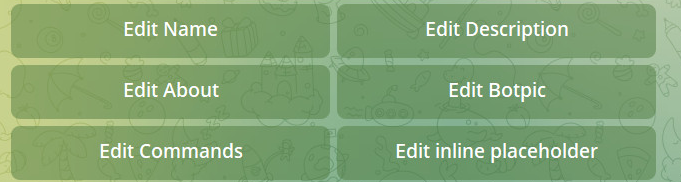


Рис. 2.5 – Навігація

Інтеграція з Telegram також спрощує доступ до даних у віддаленому режимі. Користувачі можуть отримувати сповіщення про стан енергоспоживання незалежно від свого місцезнаходження, що є особливо корисним для керування великими об'єктами або мережами. Така система підвищує енергоефективність, допомагає уникати перевитрат та надає можливість миттєвого реагування на будь-які зміни чи збої в мережі.

Інтеграція з Telegram API дозволяє створювати боти, які забезпечують зручний і швидкий доступ до даних для користувачів, надаючи їм можливість отримувати детальні звіти, графічну інформацію та сповіщення про енергоспоживання у режимі реального часу. Telegram API надає розробникам потужний інструментарій для інтерактивної взаємодії з користувачем, який легко інтегрується з системами моніторингу енергоспоживання та дозволяє автоматизувати передачу даних. Завдяки цьому підходу користувачі можуть відстежувати свій енергобаланс, бачити аналіз використання ресурсів і оперативно реагувати на зміни у споживанні.

Процес передачі даних через Telegram API побудований наступним чином: мікроконтролер, підключений до сенсорів, передає зібрані дані на сервер, де відбувається первинна обробка показників. Цей сервер може знаходитися в локальній мережі або бути хмарним, залежно від вимог проєкту. Після обробки даних сервер через Telegram API відправляє їх Telegram боту, який через Webhook або polling отримує актуальні значення енергоспоживання, аналізує їх і надсилає повідомлення користувачу в чаті. Завдяки інтеграції з API, бот може не лише передавати текстову інформацію, а й надсилати файли, графіки, таблиці для візуалізації даних у форматах, зручних для швидкого аналізу.

Telegram API надає ботам різноманітні можливості для передачі аналітичних звітів та взаємодії з користувачем. Наприклад, боти можуть відповідати на команди користувача для отримання детальних звітів: команда “/summary” викликає короткий огляд енергоспоживання за останній період, тоді як “/detailed\_report” може надати користувачу деталізовану інформацію з розбиттям по годинах або днях, залежно від потреб. Бот також може підтримувати розширені налаштування — наприклад, команду “/set\_limit”, яка встановлює максимальні ліміти споживання, при перевищенні яких користувач отримає сповіщення.

Однією з головних переваг Telegram API є підтримка асинхронної передачі даних, що дозволяє ботам працювати в реальному часі. Ця функція забезпечує миттєве оновлення інформації в момент її зміни на сервері. Використовуючи механізми Webhook або polling, бот може автоматично надсилати користувачу повідомлення, як тільки надходять нові дані або фіксується аномалія. Наприклад, у разі перевищення ліміту споживання або виявлення різких змін у навантаженні, бот миттєво надсилає сповіщення, яке дозволяє користувачу своєчасно реагувати, вживаючи заходів для зниження навантаження або оптимізації роботи пристроїв.

Telegram API також підтримує інтерактивні елементи, зокрема клавіатури та кнопки, що підвищує зручність навігації у бота. Завдяки цим елементам користувачі можуть легко керувати пристроями або змінювати налаштування безпосередньо у месенджері. Наприклад, використовуючи клавіатуру в Telegram, користувач може вибрати з меню потрібну команду для отримання звітів, налаштування інтервалів сповіщень або запуску аналізу споживання. Це дозволяє зменшити кількість ручного введення і прискорює процес отримання необхідних даних. Інтеграція з графічними бібліотеками, такими як Matplotlib, дозволяє боту надсилати наочні графіки й діаграми, що допомагає користувачу краще розуміти динаміку змін у споживанні.

Ще однією значною перевагою є можливість віддаленого доступу до інформації. Telegram API дозволяє користувачам отримувати звіти, навіть якщо вони знаходяться поза офісом або на відстані від місця, де встановлено систему моніторингу. Це особливо корисно для великих об’єктів або комплексів, де важливо мати постійний контроль над станом енергоспоживання. Користувачі можуть отримувати сповіщення про зміни, аномалії або досягнення встановлених лімітів у будь-який момент, що дозволяє зберігати контроль над витратами та оптимізувати енергоефективність об’єкта.

Завдяки інтеграції Telegram API з системою моніторингу енергоспоживання користувачі мають можливість миттєво отримувати дані, управляти налаштуваннями дистанційно та оперативно реагувати на будь-які зміни, що сприяє підвищенню ефективності використання енергоресурсів.

##### 2.3. Модуль збору та обробки даних із використанням мікроконтролерів

#### **2.3.1. Використання мікроконтролерів для моніторингу**

Arduino та ESP32 — найпоширеніші мікроконтролери для проектів з моніторингу енергії. Ці мікроконтролери зчитують дані з датчиків та передають їх через Wi-Fi до сервера або бота в Telegram.

Програмування мікроконтролерів відбувається за допомогою популярних середовищ, як-от Arduino IDE, що забезпечує простоту розробки та дозволяє легко інтегрувати нові функціональні можливості. Наприклад, код на ESP32 може бути налаштований для збору показників споживання кожні кілька секунд, а при перевищенні встановлених лімітів мікроконтролер автоматично надсилає сповіщення користувачу через Telegram бот.

Окрім збору даних, мікроконтролери можуть виконувати локальні обчислення. Наприклад, обчислення середнього споживання або виявлення пікових значень можуть виконуватися безпосередньо на пристрої, що зменшує навантаження на сервер. Це також дозволяє системі швидше реагувати на аномалії, як-от різкі стрибки споживання, і надсилати миттєві сповіщення.

Інтеграція з Telegram API дає змогу користувачам взаємодіяти з системою через чат-ботів, отримуючи оперативні дані та надсилаючи команди. Наприклад, команда “*/summary”* може викликати зчитування даних з мікроконтролера та формування звіту про споживання, який миттєво відображається у месенджері. Це забезпечує повну прозорість для користувачів і дозволяє вчасно виявляти проблеми чи перевитрати.

Використання мікроконтролерів у таких системах також підвищує масштабованість. Наприклад, до одного мікроконтролера можна підключити кілька сенсорів для моніторингу різних зон будівлі чи обладнання, а дані з різних пристроїв можуть бути об’єднані на одному сервері або аналітичній платформі для подальшої обробки та аналізу.

Модуль збору та обробки даних із використанням мікроконтролерів, таких як Arduino та ESP32, є основою сучасних систем моніторингу енергії завдяки їх доступності, простоті налаштування та потужним можливостям для збору й обробки інформації. Arduino та ESP32 можуть працювати з різними типами сенсорів для отримання даних про споживання електроенергії, температурний режим, напругу тощо. Дані, отримані мікроконтролерами, можуть передаватися через Wi-Fi або інші бездротові протоколи до центральної бази даних або ж безпосередньо до користувача через інтерфейс, як-от Telegram бот, що значно спрощує відстеження показників.

Програмування мікроконтролерів здійснюється за допомогою зручних інструментів, таких як Arduino IDE, що робить розробку таких систем доступною навіть для початківців. Завдяки цьому мікроконтролери можна налаштувати для регулярного зчитування даних із сенсорів і надсилання повідомлень у Telegram бот щоразу, коли споживання енергії перевищує встановлені параметри. Наприклад, система може бути налаштована так, щоб кожні 10 секунд зчитувати показники та відправляти звіт користувачу у разі виявлення аномальних значень. Це дозволяє користувачам не тільки спостерігати за станом енергоспоживання, а й вчасно реагувати на будь-які відхилення або перевитрати.

Однією з ключових особливостей мікроконтролерів є здатність обробляти отримані дані безпосередньо на пристрої. Виконання локальних обчислень дозволяє уникати зайвих запитів до сервера, тим самим знижуючи навантаження на центральну систему та прискорюючи процес реакції на певні події. Наприклад, ESP32 може використовуватися для обчислення середнього споживання енергії за певний проміжок часу, а при досягненні пікових значень автоматично надсилати попередження. Такий підхід забезпечує не тільки оперативність, а й додаткову надійність, оскільки основні обчислення виконуються локально, що зменшує залежність системи від зовнішніх серверів.

Інтеграція з Telegram API відкриває для користувачів зручний спосіб доступу до інформації через бот, який надає звіти, сповіщення та аналітику у зручному форматі. Користувачі можуть надсилати команди, наприклад “/summary”, для миттєвого отримання актуальних даних, зібраних мікроконтролером. Бот також дозволяє встановлювати параметри відстеження й отримувати історичні дані для аналізу, що забезпечує гнучкість у використанні системи та повний контроль над споживанням енергії.

Використання мікроконтролерів значно покращує масштабованість таких систем. Завдяки можливості підключення кількох сенсорів до одного контролера можна налаштувати моніторинг окремих зон або приладів у межах однієї будівлі. Такий підхід дозволяє знижувати витрати на обладнання та спрощує технічне обслуговування системи. Дані з різних мікроконтролерів можуть передаватися на єдину платформу, що дозволяє виконувати обробку та аналіз на одному сервері й отримувати повну картину енергоспоживання на всіх рівнях системи.

Отже, впровадження мікроконтролерів у систему моніторингу електроенергії дозволяє не лише підвищити ефективність збору даних, а й оптимізувати роботу всієї системи шляхом локальних обчислень і автоматичних сповіщень. Такий підхід забезпечує високу адаптивність системи, робить її доступною для розширення та спрощує інтеграцію з іншими пристроями та сервісами.

#### **2.3.2. Архітектура підключення**

Архітектура підключення мікроконтролерів у системі моніторингу енергоспоживання є ключовим компонентом, що забезпечує ефективну інтеграцію сенсорів, обробку даних і передачу результатів користувачам через Telegram API. Основна мета такої архітектури — забезпечити безперебійний зв'язок між апаратними компонентами (сенсорами та мікроконтролерами) і програмними інтерфейсами (серверами, хмарними платформами або безпосередньо Telegram ботом).

Підключення починається з інтеграції сенсорів до мікроконтролера, такого як ESP32 або Arduino. Мікроконтролери отримують сигнал від сенсорів через аналогові або цифрові входи, після чого дані попередньо обробляються і готуються для подальшої передачі.

Для зв'язку мікроконтролера з мережею використовуються різні протоколи: Wi-Fi, Bluetooth, MQTT або HTTP. Наприклад, у випадку з ESP32 з його вбудованим Wi-Fi модулем, дані можуть безпосередньо надсилатися на сервер або у хмару для зберігання й аналізу. Ці дані можуть також передаватися у Telegram бот через Webhook або механізм polling, що забезпечує асинхронний доступ до інформації у режимі реального часу.

Архітектура передбачає можливість організації локального чи хмарного сервера для проміжної обробки даних. Це дозволяє розподіляти навантаження та забезпечує гнучкість у роботі системи. На сервері або в хмарі дані можуть агрегуватися та аналізуватися перед відправленням користувачу. Наприклад, дані про споживання можуть бути об'єднані у зведені таблиці або візуалізовані у вигляді графіків за допомогою бібліотеки Matplotlib та надіслані через Telegram у вигляді повідомлень або зображень.

Telegram API виступає важливим елементом у цій архітектурі, оскільки забезпечує зручний інтерфейс для взаємодії з користувачем. Користувачі можуть надсилати команди, такі як “*/begin”* або “*/inspect”*, які запускають зчитування та обробку даних на мікроконтролері. Дані про споживання, отримані від сенсорів, передаються у вигляді текстових повідомлень або файлів CSV, які можна переглядати безпосередньо в Telegram.

У випадку, якщо система виявляє аномальні значення або перевищення лімітів, Telegram бот може автоматично надіслати сповіщення користувачу, що дозволяє вчасно вжити заходів для запобігання перевантаженню мережі. Архітектура підключення також передбачає можливість розширення — кілька мікроконтролерів і сенсорів можуть працювати в одній мережі, передаючи дані на один сервер для централізованого управління та аналізу.

Завдяки цій архітектурі система моніторингу енергоспоживання стає масштабованою, надійною та зручною у використанні, забезпечуючи прозорий і оперативний доступ до інформації через Telegram бот.

Архітектура підключення:

1. Мікроконтролер зчитує дані з сенсорів (струму, напруги, потужності).
2. Модуль Wi-Fi відправляє ці дані на сервер (можливо через MQTT протокол).
3. Telegram бот отримує проаналізовані дані і надсилає їх користувачу.

*import telebot*

*TOKEN = 'YOUR\_BOT\_API\_TOKEN'*

*bot = telebot.TeleBot(TOKEN)*

*@bot.message\_handler(commands=['begin'])*

*def send\_welcome(message):*

*bot.reply\_to(message, "Ласкаво просимо! Використовуйте команду /status для перевірки енергоспоживання.")*

*@bot.message\_handler(commands=['inspect'])*

*def send\_status(message):*

*# Тут інтегрується код для отримання даних з сервера*

*energy\_data = "Поточне споживання: 2.5 кВт/год"*

*bot.reply\_to(message, energy\_data)*

*bot.polling()*

Цей приклад показує, як можна створити бота, який отримуватиме та надсилатиме інформацію про енергоспоживання через Telegram.

У системі моніторингу енергоспоживання важливою є саме структура підключення всіх компонентів, яка дозволяє масштабувати та адаптувати систему до різних умов і завдань користувача. В процесі реалізації архітектури підключення враховуються кілька важливих аспектів, зокрема: способи взаємодії з інтерфейсом, безперебійність передачі даних, можливість адаптації системи для роботи з різними типами серверів, а також забезпечення необхідної швидкості та точності обробки показників енергоспоживання.

Роль мікроконтролерів в архітектурі підключення виходить за межі простої передачі даних від сенсорів. Наприклад, один мікроконтролер може отримувати дані одразу від декількох сенсорів, кожен з яких фіксує окремі параметри: рівень струму, напругу, температуру навколишнього середовища або навіть наявність руху в приміщенні. Це дає змогу об'єднати показники для комплексного аналізу і застосовувати їх у розрахунках. Для цього мікроконтролери можуть бути інтегровані з проміжними серверами або хмарними сховищами, які виконують додаткову обробку перед відправленням даних користувачу. Наприклад, дані можуть зберігатися в хмарному сховищі, де виконуються обчислення, і надсилатися користувачу у вигляді структурованих звітів.

Залежно від потреб користувача, Telegram бот може надсилати як короткі повідомлення про поточний стан енергоспоживання, так і докладні звіти в різних форматах, зокрема CSV чи JSON, які зручні для зберігання і подальшого аналізу. Бот може передбачати розширені можливості, включаючи використання команд для отримання зведених показників споживання за певний проміжок часу, доступ до історії даних і формування графічних звітів. Окрім команди “/begin” для старту взаємодії, можуть використовуватись команди на кшталт “/summary” для отримання звіту про загальне енергоспоживання або “/graph” для отримання візуалізації споживання за добу чи тиждень.

Архітектура зв’язку також передбачає гнучкість підключення мікроконтролерів з можливістю зміни протоколу. Наприклад, якщо пристрій використовує Wi-Fi для віддаленого доступу, але з'являється потреба перейти на менш енергозатратний Bluetooth, система може адаптуватися до таких вимог, налаштовуючи передачу з використанням нових параметрів. Це дозволяє пристрою функціонувати в різних мережевих умовах. У масштабних проєктах, наприклад для багатокімнатного будинку або офісу, кілька мікроконтролерів можуть бути налаштовані для взаємодії між собою через локальну мережу, що спрощує доступ до інформації.

Щодо автоматичного надсилання сповіщень: бот може бути налаштований для повідомлення про критичні значення з різними рівнями важливості, як-то "високе навантаження", "відхилення від норми", або "перевищення ліміту". Наприклад, якщо сенсори фіксують значне підвищення напруги, яке може призвести до перевантаження мережі, бот негайно надсилає користувачу попередження і пропонує варіанти дій для запобігання наслідкам.

Таким чином, гнучкість архітектури підключення дозволяє користувачам у будь-який момент контролювати енергоспоживання у приміщеннях та отримувати повні аналітичні дані, що забезпечує своєчасне прийняття рішень і підвищує ефективність використання енергії.

Завдяки використанню мікроконтролерів, таких як ESP32 чи Arduino, система може об'єднувати дані з декількох сенсорів у централізовану структуру, де дані передаються до центрального сервера для обробки. Це дозволяє не лише агрегувати інформацію в режимі реального часу, але й зберігати історичні дані для довготривалого аналізу.

Так, на великих об’єктах можна використовувати декілька рівнів архітектури: на першому рівні — сенсори та мікроконтролери, що збирають дані локально; на другому — проміжні шлюзи чи сервери, які агрегують і передають дані у хмару; і, нарешті, на третьому рівні — хмарні аналітичні системи, які забезпечують обробку великих обсягів інформації. Така багаторівнева структура дозволяє знизити навантаження на мережу, оскільки обробка частини даних відбувається на проміжних рівнях. Цей підхід також забезпечує безперервну роботу системи, оскільки в разі збою одного з компонентів дані можуть бути тимчасово збережені на іншому рівні і передані користувачу після відновлення зв'язку.

Окрім стандартних сенсорів струму та напруги, у систему можна інтегрувати датчики температури, вологості, освітленості та навіть датчики присутності для створення комплексного аналізу споживання енергії в різних умовах. Наприклад, за наявності датчиків присутності система може автоматично вимикати світло або знижувати потужність кондиціонерів у приміщеннях, де ніхто не знаходиться, що значно знижує загальне енергоспоживання. Така інтелектуальна автоматизація сприяє покращенню екологічних показників будівлі, а також дозволяє користувачам економити на рахунках за електроенергію.

Важливим аспектом є також забезпечення безпеки архітектури підключення, оскільки система може містити конфіденційну інформацію про споживання електроенергії та схеми роботи електрообладнання. Для цього застосовуються протоколи безпеки, такі як TLS (Transport Layer Security) для захищеного з'єднання з сервером, а також аутентифікація користувача з використанням багатофакторної верифікації. При впровадженні Telegram API забезпечується шифрування повідомлень, а також надається можливість обмеження доступу до певних функцій бота залежно від ролі користувача. Наприклад, адміністратори можуть мати доступ до налаштувань системи та історичних даних, тоді як звичайні користувачі можуть переглядати лише поточний статус споживання.

Для великих проєктів або підприємств важливо забезпечити інтеграцію із системами автоматизованого управління будівлями (BMS — Building Management Systems). Це дозволяє централізовано керувати енергоспоживанням і зменшувати витрати, завдяки оптимізації всіх інженерних систем будівлі. У рамках такої інтеграції Telegram бот може використовуватися як інтерфейс для моніторингу BMS, отримуючи інформацію не лише про енергоспоживання, але й про інші показники роботи інженерних систем, наприклад, рівень заповненості резервуарів води або тиск у трубах системи опалення.

Додатковою перевагою є можливість підключення системи до сервісів розумного будинку, таких як Amazon Alexa чи Google Home, що дозволяє користувачам керувати енергоспоживанням голосовими командами. У таких випадках Telegram бот може виступати як місток між системою моніторингу і сервісами голосового управління, отримуючи команди від користувача і передаючи їх мікроконтролерам. Наприклад, за командою користувача через Google Home система може автоматично знижувати температуру кондиціонера або вимикати світло в певних зонах будинку.

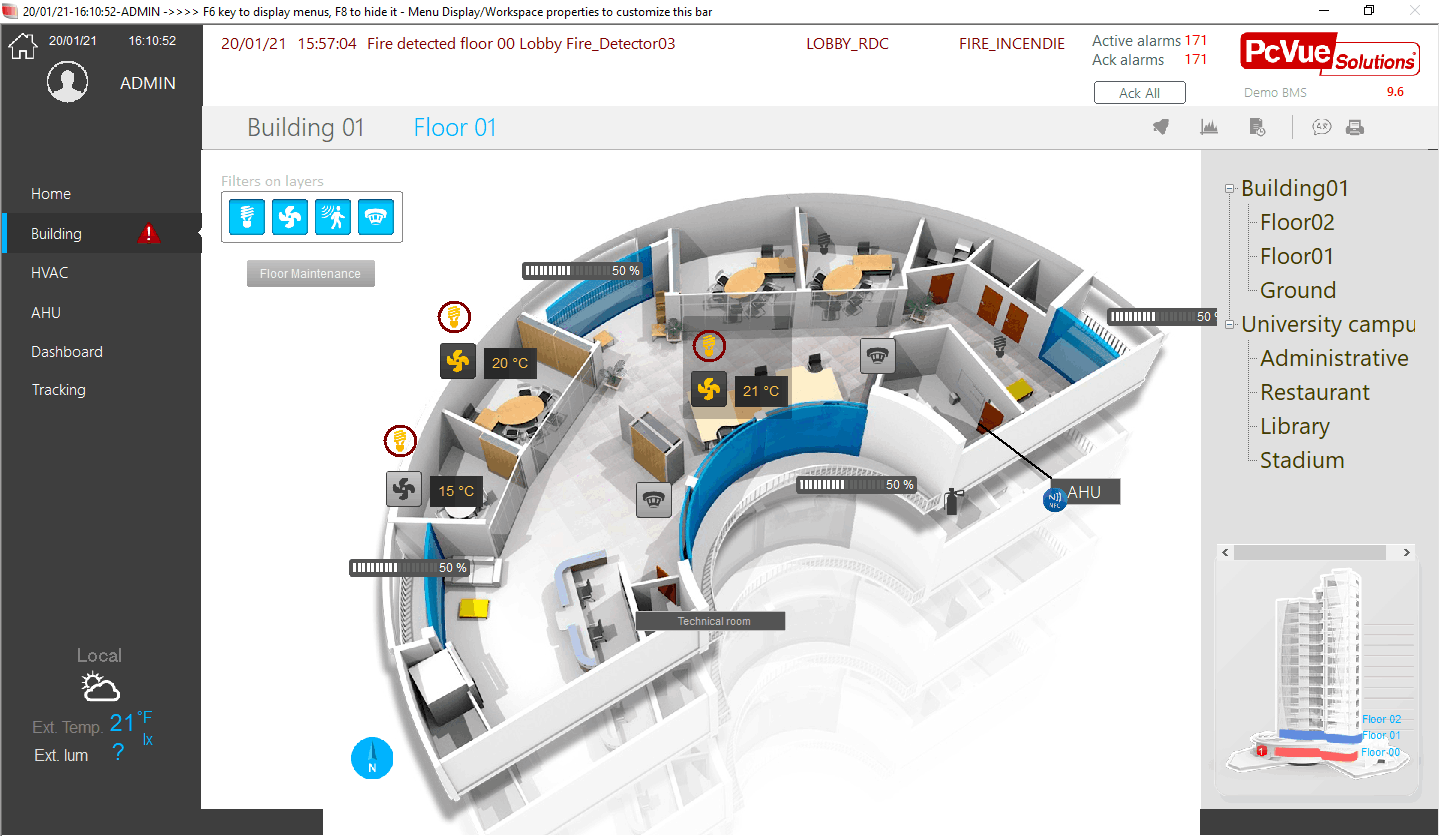


Рис. 2.6 – BMS — Building Management Systems

Таким чином, багаторівнева та гнучка архітектура підключення забезпечує високу адаптивність системи, робить її придатною для використання в найрізноманітніших умовах і дозволяє користувачам оперативно контролювати енергоспоживання, вчасно отримувати сповіщення про аномалії та приймати рішення для покращення ефективності використання енергоресурсів. Це особливо важливо в умовах великого споживання електроенергії, коли кожна деталь у зниженні енергетичних витрат може забезпечити значну економію та покращити екологічні показники об'єкта.[3]

### Висновок до розділу 2

Цей розділ розглядає основні види сенсорів та приладів для моніторингу енергоспоживання та детально описує їхню інтеграцію з Telegram API. Завдяки таким технологіям користувач може отримувати аналітичну інформацію у реальному часі та здійснювати управління енергоспоживанням через месенджер.

Використання смарт-лічильників, сенсорів і мікроконтролерів разом із Telegram API дозволяє створити інтегровану систему моніторингу та управління енергоспоживанням. Такі рішення підвищують ефективність використання енергії, забезпечують зручність у користуванні та допомагають уникати перевитрат.

#### **РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ДЛЯ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ**

Система складається з кількох ключових модулів:

1. **База даних:** CSV-файл для зберігання даних.
2. **Telegram бот:** Взаємодіє з користувачем, отримує дані та відправляє аналітику.
3. **Функції обробки даних:** Виконують аналіз та створюють візуальні звіти.
4. **Модуль нагадувань:** Використовує asyncio для періодичних повідомлень.

##### 3.1. Зберігання, обробка та скидання даних

##### 3.1.1. Зберігання та обробка даних

Дані про енергоспоживання зберігаються у CSV-файлі. Для забезпечення консистентності даних використовуються функції load\_data() та save\_data() для завантаження та збереження даних.

*def load\_data():*

*return pd.read\_csv('energy\_data.csv')*

*def save\_data(data):*

*data.to\_csv('energy\_data.csv', index=False)*

У такій архітектурі CSV-файл виступає як база даних, що дозволяє додавати нові записи через команду бота або зчитувати дані для аналізу.[17]

#### **3.1.2. Скидання даних**

Команда “*/reset”* виконує анулювання даних які були внесені або зчитані з файлу.

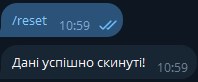


Рис. 3.1 – скидання даних

*async def reset(update: Update, context: CallbackContext) -> None:*

*os.remove(data\_file)*

*pd.DataFrame(columns=["timestamp", "consumption"]).to\_csv(data\_file, index=False)*

*await update.message.reply\_text('Дані успішно скинуті!')*

##### 3.2. Аналіз, візуалізація та звітність

##### 3.2.1. Команда “*/analyze”*: Аналіз даних та візуалізація

Команда “*/analyze”* виконує статистичний аналіз споживання електроенергії: підрахунок загального, середнього, максимального та мінімального споживання. Також створюється графік для візуалізації результатів.

**Алгоритм виконання:**

1. Завантаження даних із CSV-файлу.
2. Розрахунок ключових показників.
3. Побудова графіка із використанням matplotlib.
4. Надсилання результатів та графіка користувачу через Telegram.

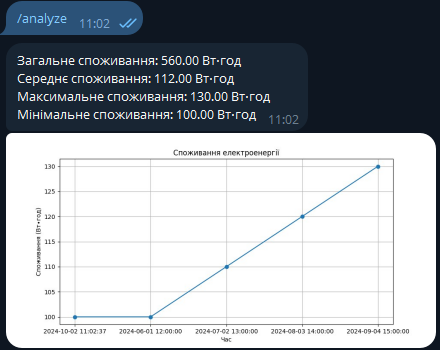


Рис. 3.2 – Надсилання результатів та графіка

*async def analyze(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для аналізу.')*

*return*

*total\_consumption = data['consumption'].sum()*

*average\_consumption = data['consumption'].mean()*

*max\_consumption = data['consumption'].max()*

*min\_consumption = data['consumption'].min()*

*plt.figure(figsize=(10, 5))*

*plt.plot(data['timestamp'], data['consumption'], marker='o')*

*plt.xlabel('Час')*

*plt.ylabel('Споживання (Вт∙год)')*

*plt.grid()*

*plt.savefig('consumption\_plot.png')*

*plt.close()*

*await update.message.reply\_text(*

*f'Загальне споживання: {total\_consumption:.2f} Вт∙год\n'*

*f'Середнє споживання: {average\_consumption:.2f} Вт∙год\n'*

*f'Максимальне: {max\_consumption:.2f} Вт∙год\n'*

*f'Мінімальне: {min\_consumption:.2f} Вт∙год'*

*)*

*await context.bot.send\_photo(update.effective\_chat.id, open('consumption\_plot.png', 'rb'))[18]*

##### 3.2.2. Команда “*/report”*. Звіт даних

Команда “*/report”* у Telegram боті надає користувачеві детальний звіт про всі записи споживання електроенергії, які містяться у файлі energy\_data.csv. Вона виконує кілька завдань: завантаження даних, перевірку їхньої наявності та форматування звіту для відправки в Telegram чат.

### Логіка реалізації команди:

1. **Завантаження даних:**Команда використовує функцію load\_data(), щоб завантажити дані із CSV-файлу. Це забезпечує доступ до історичних записів про споживання енергії.
2. **Перевірка наявності даних:**Якщо файл пустий, користувачу надсилається повідомлення: *"У вас ще немає даних для звіту."*
3. **Формування тексту звіту:**Для кожного запису у таблиці формується рядок із зазначенням часу та кількості спожитої енергії. Дані формуються у читабельний звіт.
4. **Відправка звіту:**Бот надсилає згенерований звіт у чат за допомогою “*update.message.reply\_text()”*.

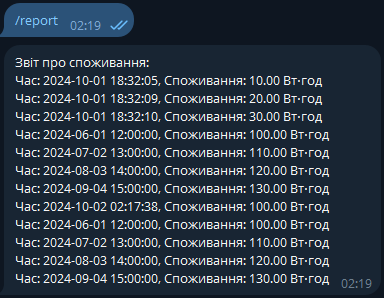


Рис. 3.5 – Звіт

*“async def report(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для звіту.')*

*return*

*report\_text = 'Звіт про споживання:\n'*

*for index, row in data.iterrows():*

*report\_text += f'Час: {row["timestamp"]}, Споживання: {row["consumption"]:.2f} Вт∙год\n'*

*await update.message.reply\_text(report\_text)”*

Команда “*/report”* є важливим інструментом для користувача, який дозволяє переглядати всю історію енергоспоживання. Вона забезпечує простий спосіб отримання звіту прямо у чаті Telegram, що підвищує зручність користування ботом.

##### 3.2.3. Команда “*/statistics”*. Статистика даних

Команда “*/statistics”* у Telegram боті надає користувачу статистичні дані про споживання електроенергії. Вона дозволяє переглянути максимальні, мінімальні, середні показники, а також кількість записів. Завдяки цьому користувач може швидко оцінити загальну картину енергоспоживання.

Логіка роботи команди “*/statistics”*:

1. **Завантаження даних:**Використовується функція load\_data(), яка відкриває та завантажує дані з CSV-файлу.
2. **Логування:**Код містить print() для перевірки, чи дані правильно завантажуються та в якому вони стані. Це допомагає у налагодженні та тестуванні бота.
3. **Перевірка наявності даних:**Якщо файл пустий, користувач отримує повідомлення: *"У вас ще немає даних для аналізу."*
4. **Розрахунок статистичних показників:**Команда обчислює кілька важливих параметрів:

* Найвище споживання: максимальне значення.
* Найнижче споживання: мінімальне значення.
* Середнє споживання: середнє арифметичне значення.
* Кількість записів: кількість рядків у таблиці.

1. **Відправка результатів у чат:**

Бот формує повідомлення зі статистикою та відправляє його користувачу.

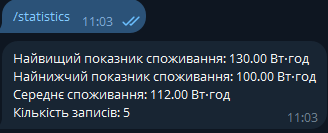


Рис. 3.6 – Статистика

*async def statistics(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*# Логування для перевірки вмісту даних*

*print("Завантажені дані:")*

*print(data)*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для аналізу.')*

*return*

*highest = data['consumption'].max()*

*lowest = data['consumption'].min()*

*average = data['consumption'].mean()*

*count = data['consumption'].count() # Кількість записів*

*await update.message.reply\_text(*

*f'Найвищий показник споживання: {highest:.2f} Вт∙год\n'*

*f'Найнижчий показник споживання: {lowest:.2f} Вт∙год\n'*

*f'Середнє споживання: {average:.2f} Вт∙год\n'*

*f'Кількість записів: {count}'*

*)*

Команда “*/statistics”* є ключовим інструментом для аналізу споживання електроенергії. Вона допомагає користувачам швидко отримувати важливі показники, що спрощує контроль за енергоспоживанням та полегшує прийняття рішень щодо оптимізації використання енергії.[7]

##### 3.3 Команди прогнозування та аналізу

##### 3.3.1. Команда “*/forecast”*. Прогнозування

Команда “*/forecast”* у боті дозволяє користувачеві отримати прогноз споживання електроенергії на наступний день. Алгоритм реалізує базовий підхід, який передбачає, що середнє споживання залишиться на рівні історичних даних. Цей метод забезпечує простий і швидкий прогноз, який може бути корисним для побутового використання.

### Логіка роботи команди “*/forecast”*:

1. **Завантаження даних**Бот використовує функцію load\_data() для отримання історичних даних про споживання з CSV-файлу.
2. **Перевірка наявності даних**Якщо файл із даними порожній або недоступний, бот надсилає повідомлення користувачу:  
   *"У вас ще немає даних для прогнозування."*
3. **Розрахунок середнього споживання**Прогноз базується на середньому значенні споживання за всі попередні записи. Це базовий підхід до прогнозування, який підходить для простих сценаріїв.
4. **Відправка прогнозу користувачу**Бот формує повідомлення із прогнозованим значенням споживання та надсилає його у чат.

*async def forecast(update: Update, context: CallbackContext) -> None:*

*# Простий алгоритм прогнозування: передбачає, що споживання залишиться незмінним*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для прогнозування.')*

*return*

*average\_consumption = data['consumption'].mean()*

*await update.message.reply\_text(*

*f'Прогнозоване споживання на наступний день: {average\_consumption:.2f} Вт∙год'*

*)*

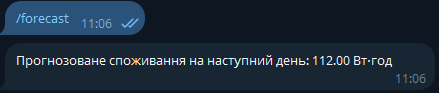


Рис. 3.7 – Прогнозування

Команда “*/forecast”* є корисним інструментом для оцінки майбутнього споживання електроенергії. Використання середнього значення як основи для прогнозування робить алгоритм простим та ефективним для побутового використання. Ця функція може бути розширена додатковими інструментами аналізу та візуалізації для отримання більш точних прогнозів.

##### 3.3.2. Команда “*/period\_analysis”*. Аналіз споживання

Команда “*/period\_analysis”* надає користувачу звіт про споживання електроенергії за останній період. Вона використовує фільтрацію даних за часом та обчислює ключові статистичні показники — загальне та середнє споживання за вибраний інтервал (за замовчуванням — останні 7 днів). Це дозволяє користувачеві отримати швидкий огляд енергоспоживання за конкретний період.

Логіка роботи команди “*/period\_analysis”*

1. **Завантаження даних із CSV:**Використовується функція load\_data() для отримання всіх записів про споживання електроенергії.
2. **Перевірка наявності даних:**Якщо файл із даними порожній, бот надсилає повідомлення:  
   *"У вас ще немає даних для аналізу."*
3. **Визначення періоду:**За замовчуванням аналізується період у 7 днів від поточної дати. Код використовує datetime і timedelta для обчислення часових меж.
4. **Фільтрація даних за періодом:**Відбираються лише ті записи, що потрапляють у визначений часовий інтервал. Якщо таких записів немає, бот повідомляє: *"У вас немає даних за цей період."*
5. **Розрахунок ключових показників:**

* Загальне споживання за обраний період.
* Середнє споживання.

1. **Відправка результату користувачу:**Після розрахунків бот формує повідомлення зі статистикою та надсилає його користувачу у чат.

*async def period\_analysis(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для аналізу.')*

*return*

*end\_date = datetime.now()*

*start\_date = end\_date - timedelta(days=7) # За останні 7 днів*

*period\_data = data[(pd.to\_datetime(data['timestamp']) >= start\_date) & (pd.to\_datetime(data['timestamp']) <= end\_date)]*

*if period\_data.empty:*

*await update.message.reply\_text('У вас немає даних за цей період.')*

*return*

*total\_consumption = period\_data['consumption'].sum()*

*average\_consumption = period\_data['consumption'].mean()*

*await update.message.reply\_text(*

*f'Загальне споживання за останні 7 днів: {total\_consumption:.2f} Вт∙год\n'*

*f'Середнє споживання за останні 7 днів: {average\_consumption:.2f} Вт∙год'*

*)*

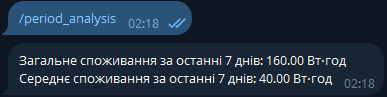


Рис. 3.8 - Аналіз за 7 днів

Команда “*/period\_analysis”* є корисним інструментом для моніторингу енергоспоживання за конкретний період. Вона допомагає користувачеві оцінити динаміку використання енергії та вчасно реагувати на зміни.

#### **3.3.3. Порівняння даних за місяцями**

Команда “*/compare”* дозволяє порівняти споживання за поточний та попередній місяці.



Рис. 3.4 – порівняння даних за поточний та минулий місяць

*async def compare(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*current\_month = datetime.now().month*

*previous\_month = (datetime.now() - timedelta(days=30)).month*

*current\_data = data[pd.to\_datetime(data['timestamp']).dt.month == current\_month]*

*previous\_data = data[pd.to\_datetime(data['timestamp']).dt.month == previous\_month]*

*current\_total = current\_data['consumption'].sum()*

*previous\_total = previous\_data['consumption'].sum()*

*await update.message.reply\_text(*

*f'Цей місяць: {current\_total:.2f} Вт∙год\n'*

*f'Попередній місяць: {previous\_total:.2f} Вт∙год'*

*)*

##### 3.4 Інтерактивна взаємодія з користувачем та експорт даних

#### **3.4.1. Обробка команд та нагадування через Telegram**. Реалізація нагадувань з використанням asyncio

Команда “*/set\_reminder\_interval”* дозволяє користувачу встановити інтервал нагадувань. Для цього ви використовуєте asyncio, що дозволяє виконувати асинхронні задачі.

*async def set\_reminder\_interval(update: Update, context: CallbackContext) -> None:*

*global reminder\_interval*

*if context.args:*

*try:*

*interval = int(context.args[0])*

*if interval <= 0:*

*await update.message.reply\_text('Інтервал має бути більшим за 0 годин.')*

*return*

*reminder\_interval = interval*

*await update.message.reply\_text(f'Інтервал встановлено: {reminder\_interval} годин.')*

*asyncio.create\_task(send\_reminders(update.effective\_chat.id, context.bot))*

*except ValueError:*

*await update.message.reply\_text('Будь ласка, введіть коректний інтервал.')*

*else:*

*await update.message.reply\_text('Будь ласка, введіть інтервал у годинах.')”*

Функція нагадувань періодично надсилає повідомлення:

*“async def send\_reminders(chat\_id, bot):*

*while True:*

*await asyncio.sleep(reminder\_interval \* 3600) # Перетворення годин на секунди*

*await bot.send\_message(chat\_id, 'Це нагадування про споживання електроенергії!')*

##### 3.4.2. Експорт CSV файлу

Користувач може експортувати дані у CSV-файл через команду “*/export”*.



Рис. 3.3 – експорт даних

*async def export\_data(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для експорту.')*

*return*

*export\_file = 'energy\_data\_export.csv'*

*data.to\_csv(export\_file, index=False)*

*await context.bot.send\_document(update.effective\_chat.id, open(export\_file, 'rb'))*

##### 3.4.3 Команда “*/help”*

Команда “*/help”* виконує функцію довідкового меню, надаючи користувачу короткий опис усіх доступних команд. Вона допомагає користувачу зрозуміти функціонал бота та спрощує навігацію між різними можливостями.

Логіка роботи команди “*/help”*:

1. **Виклик команди:**Користувач вводить “*/help”* у чаті бота.
2. **Формування повідомлення:**Бот генерує текст із переліком усіх доступних команд і короткими поясненнями до них.
3. **Відправка повідомлення:**Бот надсилає сформоване повідомлення у чат, допомагаючи користувачеві швидко орієнтуватися у можливостях.

*async def help\_command(update: Update, context: CallbackContext) -> None:*

*await update.message.reply\_text(*

*'/start - почати роботу з ботом\n'*

*'/analyze - проаналізувати дані\n'*

*'/reset - скинути дані\n'*

*'/statistics - показати статистику споживання\n'*

*'/export - експортувати дані в CSV\n'*

*'/period\_analysis - аналіз споживання за останні 7 днів\n'*

*'/forecast - прогнозування споживання\n'*

*'/set\_reminder\_interval - налаштування інтервалу нагадувань\n'*

*'/report - звіт про споживання\n'*

*'/compare - порівняння споживання з минулим місяцем'*

*)*

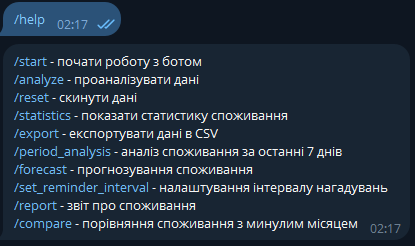


Рис. 3.9 – Меню

Команда “*/help”* є важливою частиною Telegram бота, оскільки надає користувачу зручний огляд усіх функцій. Це підвищує зручність системи, допомагаючи користувачам швидко знаходити потрібні функції та використовувати бота ефективно.[3]

##### 3.5. Аналіз і розрахунок споживання

Як приклад аналізу споживання можна розглянути користувача, який щоденно споживає електроенергію в межах 5-10 кВт∙год (для звичайного побутового споживача це стандартне споживання). Програма аналізує дані та виявляє пікові моменти використання, коли споживання найбільше (наприклад, вечірній час). Завдяки функції аналізу і прогнозування бот може запропонувати користувачу перенести частину споживання на періоди з низьким тарифом або автоматично вимкнути непотрібні пристрої.

Якщо припустити, що середнє денне споживання користувача становить 8 кВт∙год, то на місяць це приблизно 240 кВт∙год. При тарифі 2.5 грн за кВт∙год вартість електроенергії складе:

*240 кВт∙год×2.5грн/кВт∙год=600 грн*

За умови впровадження системи автоматизованого моніторингу та економії, споживання може зменшитися на 10-15%. Для середнього сценарію з економією 12% нове споживання становитиме:

*240 кВт∙год×(1−0.12)=211.2кВт∙год*

Відповідно, нові витрати на електроенергію:

*211.2кВт∙год×2.5грн/кВт∙год=528 грн*

Таким чином, щомісячна економія становить:

*600 грн−528 грн=72 грн*

Ця система, крім економії, підвищує контроль користувача над своїм енергоспоживанням, дозволяючи приймати обґрунтовані рішення щодо оптимізації споживання енергії.

Як приклад аналізу споживання для користувача з електричним опаленням можна розглянути домогосподарство, яке щомісячно споживає електроенергію в межах 2000-4000 кВт∙год під час опалювального сезону. Програма аналізує дані та виявляє періоди підвищеного використання енергії (наприклад, у холодні ночі або вечори). Завдяки функціям аналізу та прогнозування бот може запропонувати користувачу оптимізувати споживання: наприклад, за рахунок зниження температури опалення вночі або використання дешевшої електроенергії у пільгові години.

Припустимо, що середнє щомісячне споживання становить 3000 кВт∙год. За пільговим тарифом для електроопалення (1.44 грн за кВт∙год до 3000 кВт∙год), вартість електроенергії складе:

*3000 кВт∙год×1.44грн/кВт∙год=4320 грн*

Якщо домогосподарство впроваджує систему моніторингу та економії, воно може зменшити споживання на 10-15%. Для середнього сценарію з економією в 12%, нове споживання буде:

*3000 кВт∙год×(1−0.12)=2640 кВт∙год*

Відповідно, нові витрати на електроенергію:

*2640 кВт∙год×1.44грн/кВт∙год=3801.6грн*

Таким чином, щомісячна економія становить:

*4320 грн−3801.6грн=518.4грн*

Окрім економії, така система допомагає підвищити контроль користувача над енергоспоживанням і дозволяє приймати ефективні рішення щодо оптимізації використання електроенергії під час опалювального сезону.[14]

##### Висновки до розділу 3

У цьому розділі реалізовано повноцінну систему для аналізу, візуалізації та управління даними про енергоспоживання через Telegram бот. Завдяки інтеграції з **pandas**, **matplotlib** та **Telegram API** користувач отримує доступ до потужних інструментів для моніторингу та аналізу енергії в реальному часі.

Цей розділ охоплює ключові аспекти створення програмного забезпечення для збору, зберігання, аналізу та візуалізації даних про споживання електроенергії. Важливий акцент робиться на автоматизації процесів, визначенні пікових навантажень та інтерактивній візуалізації результатів через Telegram бот.

##### ВИСНОВКИ

У даній роботі було успішно розроблено Telegram бота для моніторингу, аналізу та прогнозування споживання електроенергії. Головною метою дослідження було створення ефективної та інтегрованої системи, яка б спрощувала контроль енергоспоживання для користувачів за допомогою доступних інструментів, зокрема месенджера Telegram.

Ця кваліфікаційна робота продемонструвала успішне поєднання технологій збору даних, автоматизації та комунікації через Telegram. Система є зручною, ефективною та має потенціал для подальшого розвитку та розширення. Використання таких рішень сприяє підвищенню енергоефективності як у побутових, так і у комерційних приміщеннях, роблячи вагомий внесок у зменшення енергетичних витрат та впливу на довкілля.

Для подальшого підвищення енергоефективності, користувачі можуть впроваджувати автоматизовані системи моніторингу енергоспоживання, які дозволяють оперативно реагувати на відхилення від нормальних показників і використовувати прогнози для оптимізації енергоресурсів. На основі даних про середнє споживання електроенергії користувачами системи (наприклад, 2.5 кВт∙год на день) впровадження таких заходів може забезпечити економію до 10-20% щомісячних витрат на електроенергію завдяки автоматичному вимкненню пристроїв у непікові години або використанню енергії у періоди низьких тарифів.

##### СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IJERT. *Smart Energy Meter and Monitoring System Using IoT*. IJERTCONV8IS14011. URL:<https://www.ijert.org/research/smart-energy-meter-and-monitoring-system-using-iot-IJERTCONV8IS14011.pdf>
2. IJARBEST. *Smart Meter Systems*. Vol. 7, Issue 4. URL:<https://ijarbest.com/journal/v7i4/2101>
3. ResearchGate. *IoT Based Real-Time Weather Monitoring System Using Telegram Bot and Thingsboard Platform*. URL:<https://www.researchgate.net/publication/369437069_IoT_Based_Real-Time_Weather_Monitoring_System_Using_Telegram_Bot_and_Thingsboard_Platform>
4. IJRTI. *IoT-Based Energy Monitoring*. Paper ID: IJRTI2305028. URL:<https://ijrti.org/papers/IJRTI2305028.pdf>
5. IJITEE. *Advanced Smart Meter Technology*. Vol. 9, Issue 6. URL:<https://www.ijitee.org/wp-content/uploads/papers/v9i6/F4301049620.pdf>
6. ELA KPI. *Моніторинг енергоспоживання з Telegram-сповіщенням*. URL:<https://ela.kpi.ua/server/api/core/bitstreams/4c8702d2-0796-4789-b2fc-72841aa405da/content>
7. Polymetrica. *Power Supply Monitoring with Telegram Notification*. URL:<https://polymetrica.com/uk/power-supply-monitoring-with-telegram-notification/>
8. ENEFCITIES. *Експертний огляд технологій енергоменеджменту у містах України*. 2020. URL:<https://enefcities.org.ua/upload/files/Publications/Analytics/Expert_review_2020.pdf>
9. ВНТУ. *Моніторинг енергоспоживання у великих підприємствах*. URL:<https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/21320/4559.pdf?sequence=3>
10. НПУ. *Енергоефективність в будівлях: Огляд сучасних методів*. 2018. URL:<https://enpuir.npu.edu.ua/bitstream/handle/123456789/32273/energo_2018.pdf?sequence=1>
11. ELA KPI. *Системи моніторингу та управління енергоспоживанням на підприємствах*. URL:<https://ela.kpi.ua/server/api/core/bitstreams/aea10752-d4fc-403f-8420-addc937bdad1/content>
12. Васильченко, О. І. *Системи моніторингу енергоспоживання: основи побудови та інтеграція IoT*. Київ: Видавництво НТУУ «КПІ», 2019.
13. Ільченко, М. Ю., Скрипник, Ю. В. *Технології IoT у моніторингу електроенергії*. Київ: Наукова думка, 2020.
14. ДСТУ ISO 50001:2018. Системи енергетичного менеджменту. Вимоги з настановами щодо застосування.
15. Петренко, А. В., Сидоренко, П. Л. *Мікроконтролери ESP32 для проєктів збору даних*. Харків: ХПІ, 2021.
16. Довідковий посібник: *Сенсори для вимірювання енергоспоживання: характеристики, особливості інтеграції*. Київ: Техніка, 2020.
17. Воронов, С. М. *Розробка Telegram ботів для моніторингових систем*. Запоріжжя: Політехніка, 2021.
18. Telegram API: Офіційна документація. Доступно: <https://core.telegram.org/bots/api>.
19. Білик, О. П., Савчук, Д. В. *Сучасні методи аналізу енергоспоживання у системах IoT*. Журнал «Інформаційні технології та енергетика», №4, 2022.
20. Дрозд, І. В., Чорний, В. К. *Інтеграція IoT-рішень із засобами комунікації для побутового енергоменеджменту*. Вісник НТУУ «КПІ», 2023.
21. Кравченко, О. Г. *Тестування систем енергоспоживання з інтерактивними функціями Telegram*. Тези конференції «Цифрові технології в енергетиці», Київ, 2023.
22. Левченко, П. О. *Впровадження Telegram-ботів для аналізу енергоспоживання: кейс-дослідження*. Журнал «Розумний будинок», 2022.

##### 

##### ДОДАТОК А

*import pandas as pd*

*import matplotlib.pyplot as plt*

*from datetime import datetime, timedelta*

*from telegram import Update*

*from telegram.ext import Application, CommandHandler, MessageHandler, filters, CallbackContext*

*import os*

*import asyncio*

*from telegram import ReplyKeyboardMarkup*

*# Перевірка наявності файлу з даними*

*data\_file = 'energy\_data.csv'*

*if not os.path.exists(data\_file):*

*# Створення файлу з колонками, якщо його не існує*

*pd.DataFrame(columns=["timestamp", "consumption"]).to\_csv(data\_file, index=False)*

*# Функція для завантаження даних*

*def load\_data():*

*return pd.read\_csv(data\_file)*

*# Функція для збереження нових даних*

*def save\_data(data):*

*data.to\_csv(data\_file, index=False)*

*# Команда для завантаження даних з energy\_data.txt*

*async def load\_data\_from\_file(update: Update, context: CallbackContext) -> None:*

*# Перевірка наявності файлу*

*if not os.path.exists('energy\_data.txt'):*

*await update.message.reply\_text('Файл energy\_data.txt не знайдено.')*

*return*

*# Завантаження даних з energy\_data.txt*

*new\_data = pd.read\_csv('energy\_data.txt')*

*# Завантаження існуючих даних з energy\_data.csv*

*existing\_data = load\_data()*

*# Об'єднання нових та існуючих даних*

*combined\_data = pd.concat([existing\_data, new\_data], ignore\_index=True)*

*# Збереження комбінованих даних у energy\_data.csv*

*save\_data(combined\_data)*

*await update.message.reply\_text('Дані успішно завантажено з energy\_data.txt!')*

*# Функція для створення клавіатури*

*def get\_main\_keyboard():*

*keyboard = [*

*["/analyze", "/reset", "/load\_data"],*

*["/report", "/statistics", "/export"],*

*["/forecast", "/compare", "/help"],*

*["/set\_reminder\_interval", "/period\_analysis"]*

*]*

*return ReplyKeyboardMarkup(keyboard, resize\_keyboard=True)*

*# Команда /start*

*async def start(update: Update, context: CallbackContext) -> None:*

*await update.message.reply\_text('Вітаю! Я бот для збору даних про споживання електроенергії. '*

*'Надішліть мені значення споживання у форматі: 100 (Вт∙год).',*

*reply\_markup=get\_main\_keyboard()*

*)*

*# Команда /analyze*

*async def analyze(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для аналізу.')*

*return*

*total\_consumption = data['consumption'].sum()*

*average\_consumption = data['consumption'].mean()*

*max\_consumption = data['consumption'].max()*

*min\_consumption = data['consumption'].min()*

*# Графік*

*plt.figure(figsize=(10, 5))*

*plt.plot(data['timestamp'], data['consumption'], marker='o')*

*plt.title('Споживання електроенергії')*

*plt.xlabel('Час')*

*plt.ylabel('Споживання (Вт∙год)')*

*plt.grid()*

*# Збереження графіка*

*plt.savefig('consumption\_plot.png')*

*plt.close()*

*# Відправка результатів користувачу*

*await update.message.reply\_text(*

*f'Загальне споживання: {total\_consumption:.2f} Вт∙год\n'*

*f'Середнє споживання: {average\_consumption:.2f} Вт∙год\n'*

*f'Максимальне споживання: {max\_consumption:.2f} Вт∙год\n'*

*f'Мінімальне споживання: {min\_consumption:.2f} Вт∙год'*

*)*

*await context.bot.send\_photo(chat\_id=update.effective\_chat.id, photo=open('consumption\_plot.png', 'rb'))*

*# Команда для скидання даних*

*async def reset(update: Update, context: CallbackContext) -> None:*

*os.remove(data\_file)*

*pd.DataFrame(columns=["timestamp", "consumption"]).to\_csv(data\_file, index=False)*

*await update.message.reply\_text('Дані успішно скинуті!')*

*# Обробка текстових повідомлень*

*async def handle\_message(update: Update, context: CallbackContext) -> None:*

*try:*

*consumption = float(update.message.text)*

*timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')*

*new\_data = pd.DataFrame({"timestamp": [timestamp], "consumption": [consumption]})*

*# Завантаження та збереження даних*

*data = load\_data()*

*data = pd.concat([data, new\_data], ignore\_index=True)*

*save\_data(data)*

*await update.message.reply\_text('Дані отримано! Дякую.')*

*except ValueError:*

*await update.message.reply\_text('Будь ласка, надішліть число у форматі: 100 (Вт∙год).')*

*# Статистика*

*async def statistics(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*# Логування для перевірки вмісту даних*

*print("Завантажені дані:")*

*print(data)*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для аналізу.')*

*return*

*highest = data['consumption'].max()*

*lowest = data['consumption'].min()*

*average = data['consumption'].mean()*

*count = data['consumption'].count() # Кількість записів*

*await update.message.reply\_text(*

*f'Найвищий показник споживання: {highest:.2f} Вт∙год\n'*

*f'Найнижчий показник споживання: {lowest:.2f} Вт∙год\n'*

*f'Середнє споживання: {average:.2f} Вт∙год\n'*

*f'Кількість записів: {count}'*

*)*

*# Команда /export*

*async def export\_data(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для експорту.')*

*return*

*export\_file = 'energy\_data\_export.csv'*

*data.to\_csv(export\_file, index=False)*

*await context.bot.send\_document(chat\_id=update.effective\_chat.id, document=open(export\_file, 'rb'))*

*# Команда для аналізу за період*

*async def period\_analysis(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для аналізу.')*

*return*

*end\_date = datetime.now()*

*start\_date = end\_date - timedelta(days=7) # За останні 7 днів*

*period\_data = data[(pd.to\_datetime(data['timestamp']) >= start\_date) & (pd.to\_datetime(data['timestamp']) <= end\_date)]*

*if period\_data.empty:*

*await update.message.reply\_text('У вас немає даних за цей період.')*

*return*

*total\_consumption = period\_data['consumption'].sum()*

*average\_consumption = period\_data['consumption'].mean()*

*await update.message.reply\_text(*

*f'Загальне споживання за останні 7 днів: {total\_consumption:.2f} Вт∙год\n'*

*f'Середнє споживання за останні 7 днів: {average\_consumption:.2f} Вт∙год'*

*)*

*# Команда /forecast (прогнозування)*

*async def forecast(update: Update, context: CallbackContext) -> None:*

*# Простий алгоритм прогнозування: передбачає, що споживання залишиться незмінним*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для прогнозування.')*

*return*

*average\_consumption = data['consumption'].mean()*

*await update.message.reply\_text(*

*f'Прогнозоване споживання на наступний день: {average\_consumption:.2f} Вт∙год'*

*)*

*# Змінна для зберігання інтервалу нагадувань*

*reminder\_interval = None*

*# Команда /set\_reminder\_interval (налаштування нагадувань)*

*async def set\_reminder\_interval(update: Update, context: CallbackContext) -> None:*

*global reminder\_interval # Використання глобальної змінної для доступу до інтервалу*

*print("Аргументи команди:", context.args)*

*if context.args:*

*try:*

*interval = int(context.args[0])*

*if interval <= 0:*

*await update.message.reply\_text('Інтервал повинен бути більшим за 0 годин.')*

*return*

*reminder\_interval = interval*

*await update.message.reply\_text(f'Інтервал нагадувань встановлено на {reminder\_interval} годин.')*

*# Запускаємо задачу для відправки нагадувань*

*asyncio.create\_task(send\_reminders(update.effective\_chat.id, context.bot))*

*except ValueError:*

*await update.message.reply\_text('Будь ласка, введіть число для інтервалу.')*

*else:*

*await update.message.reply\_text('Будь ласка, введіть інтервал у годинах.')*

*async def send\_reminders(chat\_id, bot):*

*while True:*

*await asyncio.sleep(reminder\_interval \* 3600) # Перетворюємо години в секунди*

*await bot.send\_message(chat\_id=chat\_id, text='Це нагадування про споживання електроенергії!')*

*# Команда /report (звіт)*

*async def report(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для звіту.')*

*return*

*report\_text = 'Звіт про споживання:\n'*

*for index, row in data.iterrows():*

*report\_text += f'Час: {row["timestamp"]}, Споживання: {row["consumption"]:.2f} Вт∙год\n'*

*await update.message.reply\_text(report\_text)*

*# Команда /compare (порівняння споживання)*

*async def compare(update: Update, context: CallbackContext) -> None:*

*data = load\_data()*

*if data.empty:*

*await update.message.reply\_text('У вас ще немає даних для порівняння.')*

*return*

*# Порівняння споживання за поточний місяць з минулим*

*current\_month = datetime.now().month*

*current\_year = datetime.now().year*

*current\_month\_data = data[pd.to\_datetime(data['timestamp']).dt.month == current\_month]*

*previous\_month\_data = data[pd.to\_datetime(data['timestamp']).dt.month == current\_month - 1]*

*current\_month\_consumption = current\_month\_data['consumption'].sum()*

*previous\_month\_consumption = previous\_month\_data['consumption'].sum()*

*comparison\_text = (f'Споживання цього місяця: {current\_month\_consumption:.2f} Вт∙год\n'*

*f'Споживання минулого місяця: {previous\_month\_consumption:.2f} Вт∙год')*

*await update.message.reply\_text(comparison\_text)*

*# Команда /help*

*async def help\_command(update: Update, context: CallbackContext) -> None:*

*await update.message.reply\_text(*

*'/start - почати роботу з ботом\n'*

*'/analyze - проаналізувати дані\n'*

*'/reset - скинути дані\n'*

*'/statistics - показати статистику споживання\n'*

*'/export - експортувати дані в CSV\n'*

*'/period\_analysis - аналіз споживання за останні 7 днів\n'*

*'/forecast - прогнозування споживання\n'*

*'/set\_reminder\_interval - налаштування інтервалу нагадувань\n'*

*'/report - звіт про споживання\n'*

*'/compare - порівняння споживання з минулим місяцем'*

*)*

*# Головна функція*

*def main():*

*# Створення об'єкта Application*

*application = Application.builder().token("TOKEN").build()*

*# Додавання обробників команд*

*application.add\_handler(CommandHandler("start", start))*

*application.add\_handler(CommandHandler("analyze", analyze))*

*application.add\_handler(CommandHandler("reset", reset))*

*application.add\_handler(CommandHandler("load\_data", load\_data\_from\_file))*

*application.add\_handler(CommandHandler("export", export\_data))*

*application.add\_handler(CommandHandler("period\_analysis", period\_analysis))*

*application.add\_handler(CommandHandler("forecast", forecast))*

*application.add\_handler(CommandHandler("set\_reminder\_interval", set\_reminder\_interval))*

*application.add\_handler(CommandHandler("report", report))*

*application.add\_handler(CommandHandler("statistics", statistics))*

*application.add\_handler(CommandHandler("compare", compare))*

*application.add\_handler(CommandHandler("help", help\_command))*

*# Додавання обробника текстових повідомлень*

*application.add\_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle\_message))*

*# Запуск бота*

*application.run\_polling()*

*if \_\_name\_\_ == '\_\_main\_\_':*

*main()*