Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ЦИФРОВИЙ ДВІЙНИК СИСТЕМИ РОЗУМНОГО ОСВІТЛЕННЯ

DIGITAL TWIN OF A SMART LIGHTING SYSTEM

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Геледчак Владислав

(підпис)

Керівник:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 11 » червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
« 10 » 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Геледчаку Владиславу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Цифровий двійник системи розумного освітлення*

Керівник роботи *к.т.н., доцент Лавренчук Світлана Василівна*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2024р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Загальні відомості про цифрового двійника системи розумного освітлення
Вибір та огляд засобів розробки
Опис програми цифрового двійника системи розумного освітлення
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Загальні відомості про цифрового двійника системи розумного освітлення	*Лавренчук С.В., доцент*		
Вибір та огляд засобів розробки	*Лавренчук С.В., доцент*		
Опис програми цифрового двійника системи розумного освітлення	*Лавренчук С.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,асистент*		

7. Дата видачі завдання 10.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Розділ 1. Загальні відомості про цифрового двійника системи розумного освітлення*	до 15.02.2024 р.	Виконано
2.	*Розділ 2. Вибір та огляд засобів розробки*	до 15.03.2024 р.	Виконано
3.	*Розділ 3. Опис програми цифрового двійника системи розумного освітлення*	до 04.05.2024 р.	Виконано
4.	*Висновки та пропозиції*	до 07.05.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 10.05.2024 р.	Виконано
6.	*Формування додатків*	до 15.05.2024 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 20.05.2024 р.	Виконано
8.	*Нормоконтроль*	до 01.06.2024 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 04.06.2024 р	Виконано
10.	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 11.06.2024 р.	Виконано

Здобувач вищої освіти _____ Геледчак В.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Лавренчук С.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Геледчак В. Цифровий двійник системи розумного освітлення.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2024. 48 с.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

Перший розділ присвячено огляду предметної області, тут розглядаються основні поняття про цифрового двійника системи розумного освітлення, наведено багато практичних прикладів.

В другому розділі здійснено вибір та обґрунтування засобів розробки. Обрано засоби: Python, HTML та JavaScript.

Третій розділ присвячено опису розроблених програм цифрового двійника системи розумного освітлення: детально описано розробку програм для цифрового двійника інтелектуальної системи освітлення. Він включає проєктування архітектури, процес впровадження та тестування цифрового двійника. У розділі також розглядається інтеграція різних датчиків і виконавчих механізмів, алгоритми обробки даних і інтерфейс користувача, розроблений для керування та моніторингу системи освітлення.

Ключові слова: цифровий двійник, розумна система освітлення, Python, HTML, JavaScript.

ANNOTATION

Geledchak V. Digital Twin of a Smart Lighting System

Qualifying work of a bachelor of EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2024. 48 c.

Qualification work consists of an introduction, three sections, conclusions, a references, appendices.

The first section is dedicated to reviewing the subject area, discussing the basic concepts of a digital twin of a smart lighting system, and providing numerous practical examples.

In the second section, the selection and justification of development tools. The chosen tools are Python, HTML, and JavaScript.

The third section have details the development of programs for the digital twin of the smart lighting system. It includes the architecture design, implementation process, and testing of the digital twin. The chapter also covers the integration of various sensors and actuators, data processing algorithms, and the user interface developed for controlling and monitoring the lighting system.

Keywords: digital twin, smart lighting system, Python, HTML, JavaScript.

ЗМІСТ

ВСТУП

Актуальність теми. Тема цифрового двійника системи розумного освітлення є актуальною завдяки її здатності підвищувати енергоефективність, комфорт і безпеку користувачів, а також інтегруватись з іншими системами розумного будинку. Цифрові двійники дозволяють збирати і аналізувати дані для оптимізації роботи системи, відкриваючі нові можливості для інновацій і наукових досліджень. Це сприяє поліпшенню якості життя та розвитку інноваційної економіки.

Метою роботи є розробка та впровадження цифрового двійника для системи розумного освітлення. Це включатиме дослідження та аналіз існуючих технологій, розробку алгоритмів адаптивного управління освітленням, тестування та оцінку ефективності розробленої системи.

Об’єкт дослідження – система розумного освітлення, оснащена цифровим двійником.

Предмет дослідження – процеси та технології розробки і впровадження цифрового двійника для системи розумного освітлення, включаючи алгоритми управління, методи збору і аналізу даних.

Завдання, які необхідно виконати:

– Аналіз літератури та існуючих рішень.
– Вибір засобів розробки.
– Реалізація цифрового двійника.
– Розробка інтерфейсу у вигляді сайту.
– Тестування та оптимізація.

Практичне значення – цифровий двійник системи розумного освітлення має здатність оптимізувати енергоспоживання, підвищувати комфорт і безпеку, забезпечувати моніторинг і обслуговування в реальному часі, інтегруватися з іншими розумними системами, аналізувати та прогнозувати. Це робить його важливим інструментом для ефективного управління ресурсами та зменшення екологічного впливу.

РОЗДІЛ 1
ОСНОВНІ ПОНЯТТЯ ТА ТЕОРЕТИЧНІ АСПЕКТИ

1.1 Розумне освітлення: визначення та принцип роботи

Цифровий двійник – це цифрова модель фізичного об’єкту або процесу, яка використовується в контексті конкретної предметної області та може містити різні функції залежно від потреб системи та способу використання. Цифрові двійники дають змогу моделювати реальні ситуації або результати внесення якихось змін, що досить часто дозволяє приймати оптимізовані рішення.

У сфері освітлення архітектури цифрові двійники мають величезний потенціал для революції в, експлуатації та управлінні системами освітлення.

Створюючи віртуальні копії активів освітлення, таких як освітлювальні прилади для приміщень, вуличне освітлення, розумні лампочки, освітлення для садівництва чи транспортних засобів, цифрові двійники забезпечують моніторинг, аналіз і оптимізацію всієї системи освітлення в реальному часі.

Ця технологія дозволяє візуалізувати та моделювати різні сценарії освітлення, полегшуючи оцінку споживання енергії та визначення областей для покращення [16].

Розумне освітлення – це комплекс апаратних та програмних рішень, що дозволяє автоматично регулювати освітленість в залежності від зовнішніх умов та потреб користувачів. Система включає в себе мережу датчиків, розумних ламп та контролерів, які взаємодіють між собою за допомогою Інтернету або локальної мережі.

Основний принцип роботи розумного освітлення полягає у використанні датчиків та мережевих технологій для керування джерелами світла. Система складається з декількох ключових компонентів:

- Датчики руху та освітленості. Датчики руху фіксують присутність людей в приміщенні або на вулиці. Вони можуть розміщуватися на стелі, стінах або навіть вбудовуватися в освітлювальні прилади. Датчики освітленості вимірюють рівень природного освітлення та передають ці дані до контролерів.

Це дозволяє системі автоматично регулювати яскравість штучного освітлення в залежності від зовнішніх умов.

- Розумні лампи. Світлодіодні лампи з можливістю змінювати яскравість та колір світла оснащені вбудованими модулями для зв’язку з контролерами через Wi-Fi, Zigbee, Z-Wave або інші протоколи. Розумні світильники можуть бути інтегровані в різні освітлювальні прилади, включаючи лампи, люстри, настінні та стельові світильники.
- Контролери та програмне забезпечення. Контролери збирають дані від датчиків та передають команди до розумних ламп. Вони можуть бути окремими пристроями або інтегрованими в інші компоненти системи. Програмне забезпечення дозволяє налаштовувати сценарії освітлення, планувати розклади, керувати системою за допомогою мобільних додатків або голосових асистентів (наприклад, Google Assistant, Amazon Alexa).

Розумне освітлення має низку переваг.

По-перше, це економія енергії. Автоматичне вимикання світла в порожніх приміщеннях та регулювання яскравості в залежності від рівня природного освітлення зменшує витрати електроенергії. Використання енергоефективних світлодіодних ламп значно скорочує споживання електроенергії порівняно з традиційними лампами розжарювання.

По-друге, розумне освітлення забезпечує зручність і комфорт. Можливість керувати освітленням за допомогою смартфона або голосових команд додає зручності в повсякденному житті. Налаштування різних сценаріїв освітлення (наприклад, для роботи, відпочинку, перегляду фільмів) підвищує комфорт користувачів.

По-третє, розумне освітлення підвищує безпеку. Автоматичне вмикання світла при виявленні руху підвищує безпеку в нічний час. Віддалений контроль дозволяє імітувати присутність вдома під час відпустки або відрядження, що може запобігти крадіжкам.

Технічні аспекти розумного освітлення включають використання різних протоколів зв’язку, таких як Wi-Fi, Zigbee, Z-Wave та Bluetooth. Wi-Fi забезпечує високу швидкість передачі даних та можливість інтеграції з існуючою

домашньою мережею. Zigbee та Z-Wave – це енергоефективні протоколи, що забезпечують надійну роботу в умовах низького енергоспоживання. Bluetooth підходить для локальних систем освітлення без потреби в інтернет-з'єднанні.

Розумне освітлення може бути інтегроване з іншими системами «розумного дому», такими як опалення, вентиляція, кондиціювання, системи безпеки та аудіо/відео обладнання. Використання платформ, таких як Apple HomeKit, Google Home або Amazon Echo, дозволяє створити єдину екосистему керування всіма аспектами домашнього життя.

Проте розумне освітлення стикається з певними викликами. Одна з проблем – сумісність різних пристроїв, не всі виробники забезпечують безперебійну роботу своїх продуктів з іншими. Вирішення цієї проблеми можливе через прийняття загальних стандартів та протоколів. Також важливо забезпечити захист даних користувачів через використання сучасних методів шифрування та автентифікації, оскільки зростання кількості підключених до інтернету пристроїв підвищує ризики кібератак.

Екологічні аспекти розумного освітлення також мають значення. Використання енергоефективних технологій сприяє зменшенню екологічного навантаження. Розумне освітлення може бути частиною стратегії сталого розвитку, що включає впровадження «зелених» технологій у повсякденне життя [10].

Отже, розумне освітлення є важливим елементом сучасних інтелектуальних систем керування будівлями та міським середовищем. Воно забезпечує значні переваги в плані економії енергії, зручності та безпеки, сприяючи створенню комфортних і екологічно чистих умов для життя та роботи.

1.2 Цифровий двійник системи розумного освітлення: поняття, класифікація та основні характеристики

Цифровий двійник системи розумного освітлення – це віртуальна копія фізичної системи освітлення, яка використовується для аналізу, моніторингу та оптимізації її роботи. Ця технологія дозволяє створити точну модель

освітлювальної системи, що відображає її поточний стан, функціонування та можливі сценарії розвитку.

Цифровий двійник системи розумного освітлення – це інтегрована модель, яка включає всі компоненти фізичної системи освітлення: лампи, датчики, контролери та програмне забезпечення. Вона збирає дані в реальному часі, аналізує їх і надає можливість оптимізувати роботу системи. Цей підхід забезпечує більш ефективне використання енергії, підвищення комфорту та безпеки.

Класифікація цифрових двійників системи розумного освітлення.

Цифрові двійники системи розумного освітлення можна класифікувати за кількома критеріями.

За рівнем деталізації:

- Мікро-двійники: моделюють окремі компоненти системи, такі як індивідуальні лампи або датчики.
- Макро-двійники: охоплюють всю систему освітлення, включаючи мережу ламп, датчики, контролери та центральне програмне забезпечення.

За фазою життєвого циклу:

- Двійники проєктування: використовуються на етапі розробки для моделювання та оптимізації дизайну системи.
- Двійники експлуатації: допомагають в моніторингу та керуванні системою освітлення в реальному часі.
- Двійники обслуговування: використовуються для планування та виконання технічного обслуговування, прогнозування зношування та заміни компонентів.

За функціональними можливостями:

- Статичні двійники: відображають поточний стан системи без урахування динамічних змін.
- Динамічні двійники: оновлюються в реальному часі на основі даних, що надходять від фізичних компонентів системи.

Основні характеристики цифрових двійників системи розумного освітлення:

Точність і реалістичність:

- Цифровий двійник повинен точно відображати всі аспекти фізичної системи освітлення, включаючи параметри роботи ламп, стан датчиків та інші характеристики. Це забезпечує можливість точного аналізу та прогнозування.

Інтерактивність і динамічність:

- Важливо, щоб цифровий двійник оновлювався в реальному часі, реагуючи на зміни в системі освітлення. Це дозволяє виявляти проблеми та приймати рішення оперативно.

Аналітичні можливості:

- Цифровий двійник має володіти потужними інструментами для аналізу даних та моделювання різних сценаріїв. Це включає використання алгоритмів машинного навчання, штучного інтелекту та інших передових технологій.

Інтеграція та сумісність:

- Ефективний цифровий двійник повинен інтегруватися з іншими системами управління та бути сумісним з різними джерелами даних. Це дозволяє створити єдину інформаційну екосистему для всієї інфраструктури освітлення.

Масштабованість:

- Цифровий двійник повинен бути здатним масштабуватися від моделювання окремих компонентів до цілих систем освітлення. Це забезпечує гнучкість у використанні та адаптацію до різних потреб.

Використання цифрових двійників у системах розумного освітлення забезпечує ряд значних переваг:

– Підвищення енергоефективності: завдяки точному моніторингу та оптимізації роботи системи можна значно знизити споживання енергії.

– Зниження експлуатаційних витрат: прогнозування зношення та своєчасне технічне обслуговування дозволяють зменшити витрати на ремонт і заміну компонентів.

– Покращення комфорту та безпеки: інтерактивний контроль за освітленням дозволяє створити комфортні та безпечні умови для користувачів.

– Прийняття обґрунтованих рішень: завдяки аналітичним та прогнозуючим функціям цифрового двійника, можна приймати обґрунтовані рішення на основі реальних даних.

– Зменшення екологічного впливу: оптимізація використання енергії та зменшення витрат на обслуговування сприяють зниженню екологічного навантаження.

– Підвищення надійності системи: прогнозування можливих збоїв і несправностей дозволяє завчасно вжити заходів для їх запобігання, що підвищує загальну надійність системи.

Цифрові двійники є ключовим елементом сучасних систем розумного освітлення, що забезпечують їх надійність, ефективність та адаптивність до змін.

1.3 Технологія інтернету речей (IoT) та її роль у системах розумного освітлення

Технологія Інтернету речей (IoT) представляє собою концепцію, що полягає в підключенні до інтернету різних фізичних об’єктів, які можуть обмінюватися даними та взаємодіяти один з одним без прямого втручання людиною.[14] Ця технологія стала можливою завдяки розвитку датчиків, мікроконтролерів, бездротового зв’язку та хмарних обчислень.

Основні принципи технології IoT включають:

Підключення до інтернету: фізичні об’єкт, такі як датчики, пристрої керування, домашні прилади тощо, обладнані засобами зв’язку, що дозволяють їм підключатись до мережі інтернет.

Сенсори та датчики: об’єкти можуть бути обладнані різними сенсорами та датчиками, які збирають різноманітні дані про навколишнє середовище, такі як температура, вологість, рівень освітленості тощо.

Збір та передача даних: зібрані датчиками дані передаються через мережу інтернет до хмарних сервісів або локальних сервісів для подальшої обробки та аналізу.

Аналіз та взаємодія: отримані дані аналізуються, і на їх основі приймаються рішення або виконуються дії. Наприклад в умовах розумного будинку система IoT може взаємодіяти з системами опалення, кондиціонування повітря та освітлення для оптимізації комфорту та енергоефективності.

Зворотній зв’язок: в залежності від результатів аналізу можуть бути відправленні команди назад до фізичних об’єктів для зміни їх стану або поведінки.

Технологія Інтернету речей (IoT) відіграє значну роль у системах розумного освітлення, надаючи можливості для автоматизації, оптимізації та управління освітленням. Завдяки IoT, освітлювальні пристрої можуть бути обладнані датчиками, які збирають дані про навколишнє середовище, такі як рівень освітленості та присутність людей.

Ці дані аналізуються системою, що дозволяє системі автоматично реагувати на зміни та оптимізувати роботу освітлення. Користувачі можуть віддалено керувати системою освітлення через мобільні додатки або веб-інтерфейси, а система може інтегруватися з іншими системами розумного будинку для створення інтегрованого середовища.

Дані, зібрані системою, використовуються для аналізу ефективності освітлення та вдосконалення роботи системи, забезпечуючи оптимальний комфорт та енергоефективність. IoT допомагає створити інтелектуальні та автоматизовані системи розумного освітлення, що відповідають потребам сучасного користувача.

Після детального розгляду розумного освітлення, цифрового двійника та ролі технології Інтернету речей (IoT) у цих системах, можна зробити наступні висновки.

Розумне освітлення – це інноваційна система освітлення, яка використовує сучасні технології для автоматичного контролю та управління. Основний принцип роботи полягає у використанні різноманітних сенсорів, контролерів та програмного забезпечення для автоматизованого управління освітленням з метою оптимізації його роботи з урахуванням умов навколишнього середовища та потреб користувачів.

Цифровий двійник – це віртуальна модель реального об’єкта, процесу або системи, яка дозволяє аналізувати, моніторити та оптимізувати їхню роботу. В контексті систем розумного освітлення цифровий двійник дозволяє створити

віртуальне відображення освітлення, яке можна аналізувати та оптимізувати для покращення ефективності та комфорту.

Технологія Інтернету речей відіграє ключову роль у системах розумного освітлення, надаючи можливості для збору даних, автоматизації, віддаленого керування та аналізу. Вона дозволяє створити інтелектуальні та ефективні системи освітлення, які забезпечують оптимальний комфорт для користувачів та ефективне використання енергії.

Однією з ключових переваг використання IoT в системах освітлення є можливість створення адаптивних систем, які здатні налаштовувати освітлення в реальному часі. Наприклад, система може зменшувати яскравість освітлення, коли в приміщенні достатньо природного світла, або включати світло, коли датчики виявляють рух. Це не тільки підвищує комфорт користувачів, але й значно знижує енергоспоживання.

Іншою важливою перевагою є можливість інтеграції з іншими системами розумного будинку. Наприклад, освітлення може бути синхронізоване з системами безпеки, опалення та кондиціонування повітря для створення більш зручного та безпечного середовища. Користувачі можуть налаштовувати різні сценарії, такі як «нічний режим» або «режим відпустки», які автоматично регулюють освітлення відповідно до заданих параметрів.

Крім того, зібрані дані можуть використовуватися для довгострокового аналізу та вдосконалення системи. Наприклад, аналіз даних про використання освітлення може допомогти виявити патерни споживання та розробити більш ефективні стратегії управління енергією. Це може включати прогнозування пікових навантажень та оптимізацію розкладів освітлення для максимального зниження енергоспоживання.

Користувачі можуть віддалено керувати системою освітлення через мобільні додатки або веб-інтерфейси, а система може інтегруватися з іншими системами розумного будинку для створення інтегрованого середовища. Дані, зібрані системою, використовуються для аналізу ефективності освітлення та вдосконалення роботи системи, забезпечуючи оптимальний комфорт та енергоефективність. IoT допомагає створити інтелектуальні та автоматизовані

системи розумного освітлення, які відповідають потребам сучасного користувача.

Отже, розумне освітлення, цифровий двійник та технологія Інтернету речей утворюють комплексний підхід до створення ефективних, комфортних та інтелектуальних систем освітлення, які відповідають потребам сучасного світу.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ІНСТРУМЕНТІВ

2.1 Огляд сучасних мов програмування для розробки цифрових двійників системи розумного освітлення

Цифровий двійник – це цифрова копія фізичної системи, що дозволяє моніторити, моделювати та оптимізувати її роботу. У випадку систем розумного освітлення, такі двійники можуть бути використані для моделювання ефективності освітлення, прогнозування зносу обладнання та енергоспоживання, а також для налаштування індивідуальних сценаріїв освітлення.

Для розробки таких систем можна використовувати різні мови програмування, серед яких Python, C++, JavaScript та інші.

Python є однією з найбільш популярних мов для розробки цифрових двійників завдяки своїй простоті та великій кількості бібліотек для моделювання, аналізу даних та машинного навчання [4].

– Простота синтаксису – Python легко вивчати та використовувати, що дозволяє швидко створювати прототипи.

– Багато бібліотек для аналізу даних (Pandas, NumPy), машинного навчання (TensorFlow, Scikit-Learn), а також для побудови цифрових двійників (Pandas, SimPy).

– Активна спільнота: велика спільнота розробників, що постійно створює та підтримує численні інструменти та ресурси.

C++ є однією з найпотужніших мов програмування, особливо коли мова йде про високопродуктивні додатки та системне програмування.

– Висока продуктивність забезпечує швидке виконання коду, що важливо для реального часу.

– Контроль над ресурсами дозволяє детальний контроль над апаратними ресурсами.

– Широке використання в індустрії – часто використовується в системах, де потрібна висока надійність і ефективність.

JavaScript зазвичай використовується для розробки фронтенд додатків, але також може бути корисним у створенні цифрових двійників, особливо в поєднанні з веб-технологіями.

– Легко інтегрується з веб-інтерфейсами, що дозволяє створювати візуалізації та користувацькі інтерфейси.

– Широкий вибір бібліотек та фреймворків. Наприклад, D3.js для візуалізації даних.

– Підтримка асинхронних операцій, що важливо для обробки реального часу.

Java є однією з найпопулярніших мов для розробки корпоративних додатків та систем, що потребують надійності та масштабованості [2].

– Java-програми можуть виконуватися на будь-якій платформі, яка має JVM (Java Virtual Machine).

– Має велику кількість бібліотек та фреймворків для розробки різних типів додатків, включаючи IoT та обробку даних.

– Вбудовані механізми безпеки роблять Java підходящою для розробки додатків, де потрібна надійність та захист даних.

MATLAB є популярною мовою для математичного моделювання, аналізу даних та розробки алгоритмів.

– Середовище інтегрованої розробки: MATLAB має потужні інструменти для аналізу даних, візуалізації та моделювання.

– Широкі можливості для моделювання використовуються в інженерії та науці для моделювання складних систем.

– Інтеграція з іншими мовами – може взаємодіяти з кодом на інших мовах, таких як C, C++, Python.

C# (C-Sharp) – це об'єктно-орієнтована мова програмування. Вона є частиною платформи .NET і широко використовується для створення різноманітних додатків, включаючи веб-додатки, десктопні додатки та ігри [13].

– Середовище інтегрованої розробки C# підтримується потужним середовищем розробки Visual Studio, яке забезпечує широкий спектр інструментів для налагодження, тестування та профілювання додатків.

– Широкі можливості для моделювання на C# дозволяють створювати складні моделі систем завдяки підтримці об'єктно-орієнтованого програмування та багатопотоковості.

– Інтеграція з іншими мовами – C# може взаємодіяти з кодом, написаним на інших мовах програмування, таких як C, C++, Python та інші. Це забезпечує гнучкість і дозволяє використовувати існуючі бібліотеки та модулі, що написані на інших мовах, інтегруючи їх у проєкти на C#.

Python є однією з найбільш ефективних мов програмування для розробки цифрових двійників системи розумного освітлення завдяки своїй простоті у використанні та великій кількості спеціалізованих бібліотек. Простота синтаксису Python дозволяє розробникам швидко освоювати мову та створювати прототипи, що значно зменшує час на розробку та налагодження систем. Це особливо важливо для проектів, де швидкість впровадження нових рішень є ключовим фактором.

Крім того, Python має велику кількість бібліотек для моделювання, аналізу даних та машинного навчання. Бібліотеки, такі як Pandas, NumPy, TensorFlow та Scikit-Learn, забезпечують потужні інструменти для аналізу даних та створення моделей машинного навчання, які можуть використовуватися для прогнозування зносу обладнання та оптимізації енергоспоживання. Це робить Python надзвичайно потужним інструментом для розробки інтелектуальних систем, що здатні адаптуватися до змінюваних умов та вимог користувачів.

Ще однією значною перевагою Python є його активна спільнота розробників та велика кількість плагінів і розширень. Це забезпечує швидке вирішення технічних проблем та підтримку новітніх технологій, що дозволяє інтегрувати систему розумного освітлення з іншими системами розумного будинку, створюючи більш інтегроване та ефективне середовище.

Використання Python для розробки цифрових двійників системи розумного освітлення стало фінальним вибором з огляду на його функціональність, простоту використання та широкий спектр можливостей для аналізу та моделювання.

2.2 Аналіз платформ та фреймворків для створення розумного освітлення

Для розробки систем розумного освітлення необхідні потужні інструменти та середовища розробки, які забезпечують ефективну роботу з кодом, тестування та впровадження програмних рішень. Розглянемо основні платформи та фреймворки, які можуть бути використані для цієї мети: PyCharm, Visual Studio Code, Jupyter Notebook/JupyterLab, Atom та Spyder.

PyCharm – це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, спеціально для мови програмування Python. Воно має безліч функцій, які роблять його ідеальним для розробки складних проєктів, включаючи системи розумного освітлення [9].

Основні характеристики PyCharm:

- Інтелектуальний редактор коду: підтримує автодоповнення, аналіз коду та рефакторинг, що прискорює процес розробки.
- Інструменти для тестування: вбудовані засоби для написання та запуску тестів, що допомагає забезпечити якість коду.
- Інтеграція з системами контролю версій: підтримка Git, SVN, Mercurial та інших, що дозволяє ефективно керувати змінами у коді.
- Плагіни та розширення: широкий набір плагінів для розширення функціональності, включаючи інтеграцію з базами даних та веб-фреймворками.

PyCharm є потужним інструментом для розробників, які займаються створенням комплексних систем, включаючи розумне освітлення, завдяки своїй функціональності та зручності використання.

Visual Studio Code (VS Code) – це безкоштовний редактор коду від Microsoft, який підтримує велику кількість мов програмування та розширень, що робить його універсальним інструментом для розробки [11].

Основні характеристики Visual Studio Code:

- Легкість та швидкість: VS Code швидко завантажується і працює, що забезпечує високу продуктивність.

- Розширення: велика кількість доступних розширень для різних мов програмування, середовищ та інструментів, включаючи Python, IoT та інші.
- Інтеграція з Git: вбудована підтримка Git та інших систем контролю версій для ефективного керування проєктами.
- Дебаггер: потужні інструменти для налагодження коду, що дозволяють швидко знаходити та виправляти помилки.

VS Code є популярним вибором серед розробників завдяки своїй гнучкості, швидкості та великій кількості розширень, що робить його придатним для розробки систем розумного освітлення.

Jupyter Notebook та JupyterLab – це інтерактивні середовища для програмування, які дозволяють створювати та ділитися документами, що містять живий код, рівняння, візуалізації та пояснення.

Основні характеристики Jupyter Notebook/JupyterLab:

- Інтерактивність: можливість виконувати код по частинах, що робить його зручним для тестування та візуалізації даних.
- Підтримка численних мов: хоча найчастіше використовується з Python, підтримує й інші мови через відповідні ядра.
- Вбудовані інструменти для аналізу даних: легко інтегрується з бібліотеками для обробки та візуалізації даних, такими як NumPy, pandas, Matplotlib.
- Спільна робота: можливість ділитися нотатками з колегами для спільної роботи над проєктами.

Jupyter Notebook та JupyterLab ідеально підходять для дослідницької роботи, аналізу даних та прототипування в проєктах розумного освітлення, де важлива швидкість і гнучкість тестування ідей [12].

Atom – це текстовий редактор, розроблений GitHub, який позиціонується як «хакований» редактор для сучасного програмування. Він також є безкоштовним та підтримує широкий спектр розширень.

Основні характеристики Atom:

- Налаштовуваність: велика кількість плагінів та тем, що дозволяють налаштувати редактор під свої потреби.

- Інтеграція з GitHub: легка інтеграція з Git та GitHub, що зручно для керування проєктами.
- Підтримка пакетів: можливість встановлювати додаткові пакети для розширення функціональності, включаючи підтримку різних мов програмування та інструментів.
- Спільна робота: функція Teletype дозволяє спільно редагувати код в реальному часі.

Atom є чудовим вибором для розробників, які цінують високу налаштовуваність та інтеграцію з GitHub, особливо для невеликих та середніх проєктів з розумного освітлення.

Spyder – це потужне середовище розробки, спеціально створене для наукового програмування на Python. Воно включає в себе численні інструменти для роботи з даними та аналізу [7].

Основні характеристики Spyder:

- Інтегрований редактор коду: підтримує автодоповнення, перевірку синтаксису та рефакторинг, що полегшує написання коду.
- Консоль Python: дозволяє виконувати код у реальному часі та тестувати окремі фрагменти.
- Інструменти для аналізу даних: вбудована підтримка бібліотек для наукових обчислень, таких як NumPy, SciPy, pandas та Matplotlib.
- Перегляд змінних: зручний інтерфейс для відстеження та аналізу змінних під час виконання коду.

Spyder є оптимальним вибором для наукових досліджень та проєктів, що вимагають інтенсивного аналізу даних, таких як розробка алгоритмів для розумного освітлення.

При виборі платформи або фреймворку для розробки систем розумного освітлення важливо враховувати кілька ключових аспектів: специфіку проєкту, вимоги до функціональності, особисті переваги розробників, а також можливості інтеграції обраних інструментів з іншими технологіями та платформами.

Перш за все, необхідно чітко визначити, які задачі потрібно вирішувати в рамках проєкту. Якщо проєкт передбачає роботу з великими обсягами даних та

складними алгоритмами, то Jupyter Notebook/JupyterLab та Spyder будуть найкращими виборами завдяки їх можливостям інтерактивного аналізу та візуалізації даних. Для великих та довготривалих проєктів, де важлива продуктивність і підтримка чистоти коду, PyCharm стане незамінним інструментом завдяки своїм розширеним можливостям для написання та тестування коду.

Не можна нехтувати особистими уподобаннями членів команди розробників. Кожен розробник має свої звички та індивідуальні потреби, які впливають на продуктивність роботи. Наприклад, деякі розробники віддають перевагу Visual Studio Code через його гнучкість і можливість налаштування під себе, тоді як інші можуть обрати PyCharm за його інтелектуальні функції та підтримку великих проєктів. Врахування цих аспектів може значно підвищити задоволеність команди від роботи і, як наслідок, якість кінцевого продукту.

Вибір відповідних інструментів може значно підвищити продуктивність команди та якість кінцевого продукту. Наприклад, використання PyCharm з його потужними інструментами для рефакторингу коду може значно зменшити кількість помилок та підвищити читабельність коду. Використання Jupyter Notebook для прототипування та експериментів може прискорити процес розробки нових алгоритмів та рішень.

Огляд сучасних програмних засобів для розробки систем розумного освітлення показує, що існує широкий спектр інструментів, кожен з яких має свої унікальні особливості та переваги. Вибір конкретного інструменту залежить від багатьох факторів, включаючи функціональність, простоту використання, підтримку необхідних бібліотек та фреймворків, інтеграцію з іншими системами та вартість.

Висновок можна зробити такий: кожен інструмент має свої сильні та слабкі сторони, і вибір конкретного інструменту залежить від специфіки проєкту, вимог до функціональності, особистих переваг розробників та можливостей інтеграції.

2.3 Порівняльний аналіз інструментів і вибір оптимальних рішень

Розглянемо переваги та недоліки програм (PyCharm, Visual Studio Code, Jupyter Notebook/JupyterLab, Atom, Spyder,) які представленні в таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки програм для розробки на Phyton

IDE	Підтримка мов	Спільнота та підтримка	Вартість	Сумістність з платформами	Призначення
PyCharm	Phyton, JavaScript, інші	Велика спільнота, багато плагінів, активна підтримка	Версія Community – безкоштовна, версія Professional – платна	Windows, macOS, Linux	Веб-розробка, розробка програмного забезпечення
Visual Studio Code	Підтримує багато мов за допомогою розширень	Велика спільнота, багато плагінів, активна підтримка	Безкоштовна	Windows, macOS, Linux	Універсальне використання
Jupyter Notebook/ JupyterLab	Phyton, Julia, R, та інш.	Велика спільнота, активна підтримка	Безкоштовна	Windows, macOS, Linux	Наукові дослідження, аналіз даних
Atom	Підтримує багато мов за допомогою розширень	Велика спільнота, багато плагінів, активна підтримка	Безкоштовна	Windows, macOS, Linux	Універсальне використання
Spyder	Phyton	Активна спільнота, багато плагінів	Безкоштовна	Windows, macOS, Linux	Наукове програмування

Після детального порівняльного аналізу різних інструментів для розробки цифрових двійників системи розумного освітлення можна зробити кілька ключових висновків на користь використання PyCharm.

PyCharm виявився потужним інструментом для розробки, особливо у контексті Python-проєктів. Він пропонує низку переваг, які роблять його привабливим вибором для команди розробників.

Розширені можливості для рефакторингу коду. PyCharm забезпечує потужні інструменти для рефакторингу, що допомагає зменшити кількість помилок та підвищити читабельність коду. Це особливо важливо для великих і тривалих проєктів, де підтримка чистоти коду є ключовою.

Підтримка великих проєктів. Завдяки своїм інтелектуальним функціям, таким як автозавершення коду, налагодження та інтеграція з системами контролю версій, PyCharm ідеально підходить для роботи з великими проєктами. Це забезпечує високу продуктивність та ефективність команди розробників.

Велика спільнота та активна підтримка. PyCharm має велику спільноту користувачів та розробників, що забезпечує швидкий доступ до різноманітних ресурсів, плагінів та допомоги. Це дозволяє швидко вирішувати проблеми та знаходити оптимальні рішення.

Інтеграція з іншими інструментами та платформами. PyCharm підтримує інтеграцію з багатьма інструментами та платформами, що дозволяє легко адаптувати його до специфічних вимог проєкту. Це забезпечує гнучкість та універсальність у використанні.

Враховуючи всі ці фактори, можна сказати, що PyCharm є одним із найкращих інструментів для розробки цифрових двійників системи розумного освітлення. Проте, вибір конкретного інструменту завжди залежить від специфіки проєкту, вимог до функціональності, особистих уподобань розробників та можливостей інтеграції. Тому ретельно зваживши всі ці аспекти було прийнято остаточне рішення використовувати PyCharm як основний інструмент розробника.

РОЗДІЛ 3
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ЦИФРОВОГО ДВІЙНИКА СИСТЕМИ РОЗУМНОГО ОСВІТЛЕННЯ

3.1 Розробка прототипу системи розумного освітлення

У цьому розділі ми детально розглянемо процес розробки прототипу системи розумного освітлення з використанням мови програмування Python та емульованого об’єкта. Наведений код є основою для створення функціональної системи, яка може вмикати та вимикати світло, а також зберігати журнал стану для подальшого аналізу (Додаток А).

На початку коду імпортуються дві бібліотеки, наведено на рисунку 3.1.

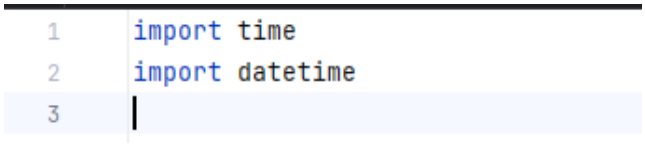

Рисунок 3.1 – Бібліотеки

Бібліотека «time» надає різні функції для роботи з часом, а «datetime» використовується для роботи з датою та часом, зокрема для створення міток часу.

Визначення класу SmartLighting наведено на рисунку 3.2.

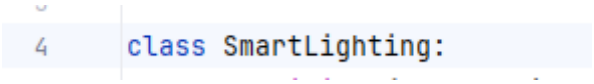

Рисунок 3.2 – Клас

Клас «SmartLighting» містить методи та атрибути, необхідні для управління розумним освітленням.

Ініціалізацію об’єкта розумного освітлення наведено на рисунку 3.3.

```
5       def __init__(self, gpio_pin):
6           # Ініціалізація об'єкту розумного освітлення за допомогою номера PIN-коду GPIO
7           self.gpio_pin = gpio_pin
8           self.on = False
9           self.state_log = []
10
```

Рисунок 3.3 – Ініціалізація об’єкта розумного освітлення

«__init__(self, gpio_pin): » – конструктор класу, який ініціалізує об’єкт розумного освітлення. В цю функцію передаємо номер PIN-коду GPIO, який використовується для керування фізичним пристроєм.

«self.gpio_pin» зберігає номер GPIO PIN.

«self.on» зберігає поточний стан світла (ввімкнено або вимкнено).

«self.state_log» – список для зберігання історії змін стану світла.

Метод «toggle» наведено на рисунку 3.4.

```
11      def toggle(self):
12          # Вмикаємо та вимикаємо світло залежно від його поточного стану
13          if self.on:
14              self.turn_off()
15          else:
16              self.turn_on()
```

Рисунок 3.4 – Реалізація методу «toggle»

«toggle(self)» – метод, який перемикає стан світла. Якщо світло ввімкнено, метод вимикає його, і навпаки.

Методи «turn_on» та «turn_off» наведено на рисунку 3.5.

```
18      def turn_on(self):
19          # Вмикаєм світло
20          self.on = True
21          self.log_state("on")
22
23      def turn_off(self):
24          # Вимкніть світло
25          self.on = False
26          self.log_state("off")
27
```

Рисунок 3.5 – Реалізація методів ввімкення та вимкнення світла

«turn_on(self)» – метод для ввімкнення світла. Встановлює стан світла в «True» та записує цю зміну в лог.

«turn_off(self)» – метод для вимкнення світла. Встановлює стан світла в «False» та записує цю зміну в лог.

Метод «get_state» наведено на рисунку 3.6.

```
28      def get_state(self):
29          # Повернути поточний стан
30          return "on" if self.on else "off"
31
```

Рисунок 3.6 – Метод для отримання поточного стану світла

«get_state(self)» – метод для отримання поточного стану світла. Повертає рядок «on» або «off» залежно від стану світла.

Метод «log_state» наведено на рисунку 3.7.

```
32      def log_state(self, state):
33          # Зазначаємо зміну стану з міткою часу
34          timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
35          self.state_log.append(f"{timestamp} - State: {state}")
36
```

Рисунок 3.7 – Зазначаємо зміну стану з міткою часу

«log_state(self, state)» – метод для запису змін стану світла з міткою часу. Додає рядок з міткою часу та станом до списку «state_log».

Приклад використання класу SmartLighting наведено на рисунку 3.8.

```
37  # Створення екземпляру класу Smart Lighting
38  light = SmartLighting(18)
39
40  # Вмикаємо світло
41  light.turn_on()
42  print("Current state:", light.get_state())
43
44  # Затемнюємо світло
45  light.turn_off()
46  print("Current state:", light.get_state())
47
48  # Знову вмикаємо світло
49  light.turn_on()
50  print("Current state:", light.get_state())
```

Рисунок 3.8 – Використання класу «SmartLighting»

Спочатку створюється екземпляр класу «SmartLighting» з номером GPIO PIN 18.

Викликаються методи «turn_on» та «turn_off», щоб вмикати та вимикати світло. Поточний стан світла виводиться на екран за допомогою методу «get_state».

Цей код є основою для створення системи розумного освітлення, яка може бути інтегрована з апаратним забезпеченням через GPIO-піни, наприклад, на платі Raspberry Pi. Логи змін стану світла дозволяють відслідковувати історію використання системи.

3.2 Інтеграція цифрового двійника системи розумного освітлення

3.2.1 Створення цифрового двійника

Цифровий двійник – це віртуальна модель фізичного об’єкта, яка дозволяє здійснювати моніторинг, аналіз та управління фізичним об’єктом в реальному часі. Додати цифрового двійника до системи розумного освітлення можна, створивши клас «DigitalTwin», який буде відповідати за відстеження стану та надання інтерфейсу для взаємодії з об’єктом (рисунок 3.9).

```
37  class DigitalTwin:
38      def __init__(self, smart_lighting):
39          self.smart_lighting = smart_lighting
40          self.state = smart_lighting.get_state()
41          self.update_log = []
42
```

Рисунок 3.9 – Цифровий двійник системи розумного освітлення

«__init__(self, smart_lighting) » – конструктор класу, який ініціалізує цифрового двійника. Приймає об’єкт «SmartLighting» та встановлює його початковий стан.

«self.smart_lighting» зберігає посилання на фізичний об’єкт «SmartLighting».

«self.state» зберігає поточний стан цифрового двійника.

«self.update_log» – список для зберігання історії змін стану цифрового двійника.

Синхронізація стану наведено на рисунку 3.10.

```
43      def sync_state(self):
44          # Синхронізація стану з фізичним пристроєм
45          current_state = self.smart_lighting.get_state()
46          if current_state != self.state:
47              self.state = current_state
48              self.log_update(current_state)
49
```

Рисунок 3.10 – Метод для синхрнізації

«sync_state(self)» – метод для синхронізації стану цифрового двійника з фізичним об’єктом. Якщо стан змінився, цей метод оновлює стан цифрового двійника та записує цю зміну в лог.

Логування оновлень наведено на рисунку 3.11.

```
50      def log_update(self, state):
51          # Логування оновлення стану цифрового двійника
52          timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
53          self.update_log.append(f"{timestamp} - Digital Twin State: {state}")
54
```

Рисунок 3.11 – Метод для запису змін

«log_update(self, state)» – метод для запису змін стану цифрового двійника з міткою часу.

Отримання стану цифрового двійника наведено на рисунку 3.12.

```
55      def get_twin_state(self):
56          # Повернути поточний стан цифрового двійника
57          return self.state
```

Рисунок 3.12 – Метод для отримання поточного стану цифрового двійника

3.2.2 Використання цифрового двійника

Створено екземпляр класу «DigitalTwin», який пов’язано з об’єктом «SmartLighting».

Використано методи «turn_on» та «turn_off» для керування станом фізичного світла, а також метод «sync_state» для синхронізації стану цифрового двійника з фізичним об’єктом.

Поточний стан як фізичного об’єкта, так і цифрового двійника виводиться на екран для демонстрації синхронізації.

Цей підхід дозволяє відслідковувати стан фізичного об’єкта та його цифрового двійника в реальному часі, що є основою для більш складних систем моніторингу та управління.

Було створено веб-додаток для управління розумним освітленням, який дозволяє користувачам легко взаємодіяти з лампами в кімнаті через зручний веб-інтерфейс. Додаток реалізований з використанням Flask для серверної частини та HTML/CSS/JavaScript для клієнтської частини. Основна мета полягала у створенні інтерактивної моделі кімнати, де користувачі можуть вмикати, вимикати і перемикати лампи.

Для реалізації бекенду було використано Flask, легкий веб-фреймворк на Python, який забезпечує простий і гнучкий спосіб створення веб-додатків. Було розроблено маршрути для обробки запитів, таких як отримання стану ламп та керування ними. Веб-інтерфейс використовує JavaScript для відправки запитів до сервера та оновлення стану ламп на сторінці в реальному часі. Додатково було налаштовано CORS (Cross-Origin Resource Sharing) для забезпечення безпечного доступу до API з сайту.

Для створення веб-додатка було обрано React, який дозволяє створювати інтерактивні користувацькі інтерфейси з використанням JavaScript та React. Використовуючи бібліотеку Axios, було реалізовано функції для відправки HTTP-запитів до Flask-сервера, що дозволяє керувати лампами безпосередньо з веб-браузера. Завдяки цьому користувачі можуть взаємодіяти з розумним освітленням через інтуїтивний веб-інтерфейс, що значно підвищує зручність використання та функціональність системи.

Налаштування Flask-додатка наведено на рисунку 3.13.

```
app = Flask(__name__)
CORS(app)

1 usage
@app.route('/')
def index():
    return render_template('index.html')

@app.route( rule: '/status', methods=['GET'])
def get_status():
    return jsonify({
        'lamp1_physical_state': light1.get_state(),
        'lamp1_digital_twin_state': digital_twin1.get_twin_state(),
        'lamp1_state_log': light1.state_log,
        'lamp1_update_log': digital_twin1.update_log,
        'lamp2_physical_state': light2.get_state(),
        'lamp2_digital_twin_state': digital_twin2.get_twin_state(),
        'lamp2_state_log': light2.state_log,
        'lamp2_update_log': digital_twin2.update_log
    })
```

Рисунок 3.13 – Flask-додаток

Функція «index()» відповідає за рендеринг головної сторінки.

Метод «get_status()» повертає поточний стан ламп та цифрових двійників у форматі JSON.

Маршрути для керування лампами наведено на рисунку 3.14.

```
@app.route( rule: '/lamp1/turn_on', methods=['POST'])
def lamp1_turn_on():
    digital_twin1.turn_on()
    return jsonify({'message': 'Lamp 1 turned on'})

@app.route( rule: '/lamp1/turn_off', methods=['POST'])
def lamp1_turn_off():
    digital_twin1.turn_off()
    return jsonify({'message': 'Lamp 1 turned off'})

@app.route( rule: '/lamp1/toggle', methods=['POST'])
def lamp1_toggle():
    digital_twin1.toggle()
    return jsonify({'message': 'Lamp 1 toggled'})

@app.route( rule: '/lamp2/turn_on', methods=['POST'])
def lamp2_turn_on():
    digital_twin2.turn_on()
    return jsonify({'message': 'Lamp 2 turned on'})

@app.route( rule: '/lamp2/turn_off', methods=['POST'])
def lamp2_turn_off():
    digital_twin2.turn_off()
    return jsonify({'message': 'Lamp 2 turned off'})

@app.route( rule: '/lamp2/toggle', methods=['POST'])
def lamp2_toggle():
    digital_twin2.toggle()
    return jsonify({'message': 'Lamp 2 toggled'})

@app.route( rule: '/generate_link', methods=['GET'])
def generate_link():
    # Генерація посилання на сайт
    link = url_for( endpoint: 'index', _external=True)
    return jsonify({'link': link})
```

Рисунок 3.14 – Реалізація роботи ламп

Ці маршрути дозволяють вмикати, вимикати і перемикати лампи через цифрові двійники. Кожен маршрут повертає повідомлення про виконану дію у форматі JSON.

Запуск додатка наведено на рисунку 3.15.

```
127
128 ▶ if __name__ == '__main__':
129         app.run(debug=True, host='0.0.0.0')
130
```

Рисунок 3.15 – Реалізація запуску додатка

Таке рішення дозволяє запускати додаток так, щоб він був доступний через локальну мережу. Вказівка «host=‘0.0.0.0’» означає, що додаток буде слухати на всіх інтерфейсах мережі.

Цей код створює Flask-додаток для керування лампами, який можна відкрити з будь-якого пристрою в локальній мережі, включаючи смартфон.

3.3 Розробка сайту що імітує систему розумного освітлення

Створено простий веб-інтерфейс для керування двома розумними лампами. Він складається з сітки, що представляє кімнату, елементів керування для кожної лампи та функції для генерації посилання. Розглянемо код детальніше.

3.3.1 Структура HTML-документу

Розділ Head містить мета-теги та заголовок, а також CSS стилі для макетування кімнати, ламп та елементів керування.

Розділ Body:

– Заголовок сторінки.

– Сітка кімнати з двома елементами ламп.

– Розділи керування для кожної лампи, включаючи кнопки для перемикання, вмикання/вимикання, встановлення яскравості та таймера.

– Розділ для генерації посилання.

Стилі CSS:

– .room визначає макет сітки для кімнати;

– .room div встановлює базові властивості для елементів сітки;

– .lamp задає специфічні стилі для елементів ламп;

– .controls – відступи для розділів керування;

– .controls button та .controls input – відступи для кнопок та полів вводу.

3.3.2 Функції JavaScript

Утилітарні функції:

– updateLampState(lampId, state) оновлює візуальний стан лампи на основі наданого стану.

– getLampState(lampId) отримує поточний стан лампи з сервера та відповідно оновлює інтерфейс користувача.

Функції керування:

– toggleLamp(lampId) надсилає POST-запит для перемикання стану лампи.

– turnOnLamp(lampId) надсилає POST-запит для увімкнення лампи.

– turnOffLamp(lampId) надсилає POST-запит для вимкнення лампи.

– setBrightness(lampId) надсилає POST-запит для встановлення яскравості лампи на вказане значення.

– setTimer(lampId) надсилає POST-запит для встановлення таймера на вимкнення лампи через вказаний час.

Генерація посилання generateLink() отримує згенероване посилання з сервера та відображає його на сторінці.

Слухач подій DOMContentLoaded отримує початковий стан обох ламп при завантаженні сторінки.

3.3.3 Аналіз коду сайту з імітацією системи розумного овсітлення

Код, що відповідає за розташування ламп на сайті, наведено на рисунку 3.16.

```
    <div class="room">
        <div id="lamp1" class="lamp" style="grid-column: 5; grid-row: 5;">
</div>
        <div id="lamp2" class="lamp" style="grid-column: 15; grid-row: 5;">
</div>
    </div>
```

Рисунок 3.16 – Розташування ламп

Два div-елементи, що представляють лампи, розташовані у певних позиціях в сітці.

Елементи керування наведено на рисунку 3.17.

```
<div class="controls">
    <h2>Lamp 1</h2>
    <button onclick="toggleLamp('lamp1')">Toggle</button>
    <button onclick="turnOnLamp('lamp1')">Turn On</button>
    <button onclick="turnOffLamp('lamp1')">Turn Off</button>
    <br>
    <label for="lamp1-brightness">Brightness:</label>
    <input type="number" id="lamp1-brightness" min="0" max="100" value="100">
    <button onclick="setBrightness('lamp1')">Set Brightness</button>
    <br>
    <label for="lamp1-timer">Timer (seconds):</label>
    <input type="number" id="lamp1-timer" min="1" max="3600" value="60">
    <button onclick="setTimer('lamp1')">Set Timer</button>
</div>
<div class="controls">
    <h2>Lamp 2</h2>
    <button onclick="toggleLamp('lamp2')">Toggle</button>
    <button onclick="turnOnLamp('lamp2')">Turn On</button>
    <button onclick="turnOffLamp('lamp2')">Turn Off</button>
    <br>
    <label for="lamp2-brightness">Brightness:</label>
    <input type="number" id="lamp2-brightness" min="0" max="100" value="100">
    <button onclick="setBrightness('lamp2')">Set Brightness</button>
    <br>
    <label for="lamp2-timer">Timer (seconds):</label>
    <input type="number" id="lamp2-timer" min="1" max="3600" value="60">
    <button onclick="setTimer('lamp2')">Set Timer</button>
</div>
<div class="controls">
    <h2>Generate Link</h2>
    <button onclick="generateLink()">Generate Link</button>
    <p id="generatedLink"></p>
</div>
```

Рисунок 3.17 – Реалізація методів керування

Кожен розділ керування містить кнопки та поля вводу для взаємодії з відповідною лампою.

Функцію оновлення стану ламп наведено на рисунку 3.18.

```
    <script>
        function updateLampState(lampId, state) {
            const lampElement = document.getElementById(lampId);
            if (state === 'on') {
                lampElement.style.backgroundColor = 'yellow';
            } else {
                lampElement.style.backgroundColor = 'red';
            }
        }

        function getLampState(lampId) {
            return fetch('/status')
                .then(response => response.json())
                .then(data => {
                    if (lampId === 'lamp1') {
                        updateLampState('lamp1', data.lamp1_physical_state);
                        document.getElementById('lamp1-brightness').value = data.lamp1
_brightness;
                    } else {
                        updateLampState('lamp2', data.lamp2_physical_state);
                        document.getElementById('lamp2-brightness').value = data.lamp2
_brightness;
                    }
                });
```

Рисунок 3.18 – Реалізація методів керування

З точки зору середовища програмування, методи керування – це функції для оновлення стану лампи на основі відповідей сервера та відповідного оновлення інтерфейсу користувача.

3.3.4 Тестування роботи сайту

Рисунок 3.19 – Функціональна схема сайту

При першому завантаженні сторінки викликаються функції getLampState(‘lamp1’) та getLampState(‘lamp2’), щоб отримати та відобразити початковий стан обох ламп.

Взаємодія користувача з інтерфейсом відбувається шляхом натискання на кнопки, які мають інтуїтивно зрозумілі назви.

Натискання на кнопки керування викликає відповідні функції (toggleLamp, turnOnLamp, turnOffLamp, setBrightness, setTimer). Ці функції надсилають POST-запити на сервер для оновлення стану ламп.

При взаємодії з сервером він надає відповіді у вигляді даних формату JSON, що вказують на успішність дій та поточний стан ламп.

Інтерфейс користувача оновлюється на основі відповіді сервера, змінюючи колір лампи для відображення стану та налаштовуючи значення яскравості.

Вигляд сайту з зображенням роботи цифрового двійника системи розумного освітлення наведено на рисунку 3.20.

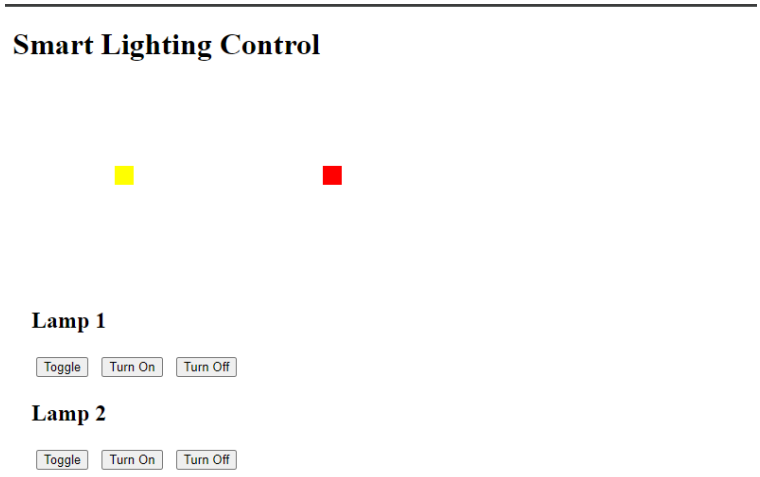

Рисунок 3.20 – Сайт для керування лампами

Ця структура коду забезпечує ефективне керування станом ламп за допомогою комбінації HTML, CSS та JavaScript, надаючи користувачам зручний інтерфейс для керування розумним освітленням.

ВИСНОВКИ

Виконано аналіз літератури та пошук існуючих рішень, що дало можливість зрозуміти поточний стан технологій у сфері систем розумного освітлення та цифрових двійників. Цей етап підготовки допоміг ідентифікувати основні виклики та можливості для подальших досліджень.

Порівнявши сучасні мови програмування та платформ для створення систем розумного освітлення, було обрано Python для реалізації основної логіки системи, а також HTML і JavaScript для розробки веб-інтерфейсу. В якості інтегрованого середовища розробки було використано PyCharm, що забезпечило ефективний процес написання та відлагодження коду.

Створено прототип системи розумного освітлення, що включає інтеграцію різних сенсорів та активаторів. Це дозволило моделювати реальні умови роботи системи та перевіряти її функціональність у різних сценаріях.

Розроблено веб-інтерфейс, що дозволяє користувачам взаємодіяти з системою розумного освітлення. Це забезпечує зручність використання та надає можливість керувати освітленням з будь-якого пристрою, що має доступ до Інтернету.

Тестування проєкту стало завершальним етапом, де було перевірено роботу системи в різних умовах та внесено необхідні корективи для підвищення її ефективності та надійності.

Отже, в результаті виконання поставлених завдань було успішно розроблено та впроваджено цифровий двійник для системи розумного освітлення, що сприятиме підвищенню її енергоефективності, комфорту та безпеки користувачів, а також її інтеграції з іншими системами розумного будинку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Applications of digital twin system in a smart city system with multi-energy. IEEE Xplore. URL: https://ieeexplore.ieee.org/document/9540135 (дата звернення: 17.03.2024).

2. Azure iot digital twins client library for java. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/java/api/overview/azure/digitaltwins-core-readme?view=azure-java-stable (дата звернення: 19.04.2024).

3. Bevor Sie fortfahren. Bevor Sie fortfahren. URL: https://store.google.com/category/connected_home (дата звернення: 14.05.2024).

4. Development of a digital twin operational platform using Python Flask– ADDENDUM / M. S. Bonney та ін. Data-Centric engineering. 2022. Т. 3. URL: https://doi.org/10.1017/dce.2022.13 (дата звернення:13.04.2024).

5. Flask-CORS – Flask-Cors 3.0.10 documentation. 404 Not Found | Read the Docs. URL: https://flask-cors.readthedocs.io/en/latest/ (дата звернення: 17.03.2024).

6. HTML: hypertext markup language MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 05.04.2024).

7. Home – spyder IDE. Home – Spyder IDE. URL: https://www.spyder-ide.org/ (дата звернення: 11.02.2024).

8. JavaScript MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 24.04.2024).

9. JetBrains. PyCharm: the Python IDE for data science and web development. JetBrains. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 02.05.2024).

10. Kreon. Smart lighting: what is it and what are the benefits?. Kreon Blog. URL: https://blog.kreon.com/smart-lighting-benefits (дата звернення: 21.03.2024).

11. Microsoft. Visual studio code – code editing. redefined. Visual Studio Code – Code Editing. Redefined. URL: https://code.visualstudio.com/ (дата звернення: 18.04.2024).

12. Project jupyter. Project Jupyter Home. URL: https://jupyter.org/ (дата звернення: 25.04.2024).

13. Sensors ti.com. Analog Embedded processing Semiconductor company TI.com. URL: https://www.ti.com/sensors/overview.html (дата звернення: 05.05.2024).

14. Smart iot lighting and sustainability – AGC lighting. AGC Lighting. URL: https://www.agcled.com/blog/smart-iot-lighting-sustainability.html (дата звернення: 28.3.2024).

15. Tutorial: code a client app – azure digital twins. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/azure/digital-twins/tutorial-code (дата звернення: 16.04.2024).

16. Towards a digital twin architecture for the lighting industry / V. Guerra та ін. Future generation computer systems. 2024. URL: https://doi.org/10.1016/j.future.2024.01.028 (дата звернення: 21.03.2024).

17. Welcome to flask – flask documentation (2.0.x). Welcome to Flask – Flask Documentation (3.0.x). URL: https://flask.palletsprojects.com/en/2.0.x/ (дата звернення: 12.05.2024).

18. Welcome to python.org. Python.org. URL: https://www.python.org/ (date of access: 21.05.2024).

19. What is a digital twin?. MathWorks – Makers of MATLAB and Simulink – MATLAB & Simulink. URL: https://www.mathworks.com/discovery/digital-twin.html (дата звернення: 08.06.2024).