МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)
магістра
(ступінь вищої освіти (освітній ступінь))

на тему КОМП'ЮТЕРНА СИСТЕМА З ВИКОРИСТАННЯМ ХМАРНИХ ТЕХНОЛОГІЙ

Виконав: студент 2 курсу, групи КНТ-512м
спеціальності
123 Комп'ютерна інженерія

Освітня програма (спеціалізація)
Комп'ютерні системи та мережі
(код і назва спеціальності)

ПЕЧЕНИЙ Р. Ю.
(прізвище та ініціали)
Керівник ІЛЬЯШЕНКО М.Б.
(прізвище та ініціали)
Рецензент МАЛИЙ О.Ю.
(прізвище та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) магістерський

Спеціальність 123 Комп'ютерна інженерія

(код і назва)

Освітня програма (спеціалізація) Комп'ютерні системи та мережі

(назва)

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

"____" _____ 2023 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

ПЕЧЕНИЙ Родіон Юрійович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Комп'ютерна система з використанням хмарних технологій

керівник проекту (роботи) ІЛЬЯШЕНКО Матвій Борисович, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "24" жовтня 2023 року № 400

2. Строк подання студентом проекту (роботи) 01 грудня 2023 року

3. Вихідні дані до проекту (роботи) програмування Python, NOD-RED, AWS IoT core

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

1) Загальна характеристика Інтернету речей

2) Задача забезпечення енергоефективності розумного будинку;

3) Використання AWS IoT Core і node-red для тестування системи

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ІЛЬЯШЕНКО М.Б., к. т. н., доцент		
Нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	10.10.2023 р.	
2	Вибір апаратного забезпечення	15.10.2023 р.	
3	Вибір програмного забезпечення	01.10.2023 р.	
4	Проектна реалізація	25.11.2023 р.	
5	Проектування програмних засобів	01.12.2023 р.	
6	Оформлення отриманих результатів у ПЗ	10.11.2023 р.	
7	Оформлення графічного матеріалу	15.11.2023 р.	
8	Захист роботи	16.12.2023 р.	

Студент  **Родіон ПЕЧЕНИЙ**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи)  **Матвій ІЛЬЯШЕНКО**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 71 с., 14 рис., 5 табл., 16 джерел.

ІНТЕРНЕТ РЕЧЕЙ, АВТОМАТИЗАЦІЯ, СИСТЕМА УПРАВЛІННЯ ЕНЕРГОСПОЖИВАННЯМ, ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ, ХМАРНІ ТЕХНОЛОГІЇ, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ.

Об'єкт дослідження- системи інтернету речей, інструмент візуального програмування Node-RED, хмарний сервіс AWS IoT Core, а також методи імітаційного моделювання та тестування для оцінки якості систем керування енергоспоживанням.

Цілі роботи— оцінити можливість інтеграції Node-RED та AWS IoT Core, застосування цих технологій у сфері інтернету речей. Розробити імітаційну модель з використанням цих технологій та перевірити її спроможність у випробуванні кількох підходів до забезпечення ефективності енергоспоживання у сфері домашньої автоматизації

Методи дослідження— аналіз джерел у сфері тестування систем інтернету речей, а також пов'язаних із використанням домашньої автоматизації для забезпечення ефективності енергоспоживання, прототипування, імітаційного моделювання, аналізу даних.

У результаті розроблено програму для імітаційного моделювання систем інтернету речей з використанням Node-RED та AWS IoT Core. На основі проведених експериментів сформульовано рекомендації щодо застосування кількох підходів до забезпечення енергоефективності.

Галузь застосування— для тестування та дослідження у сфері інтернету речей, зокрема для оцінки якості систем управління енергоспоживанням.

ABSTRACT

Explanatory note to the master's work: 71 p., 14 figures,5 tables, 16 sources.

INTERNET OF THINGS, AUTOMATION, ENERGY MANAGEMENT SYSTEM, VISUAL PROGRAMMING, CLOUD TECHNOLOGIES, SIMULATION MODELING.

Object of study- Internet of Things systems, visual programming tool Node-RED, AWS IoT Core cloud service, as well as simulation modeling and testing methods for assessing the quality of energy management systems.

The objectives of the study are to evaluate the possibility of integrating Node-RED and AWS IoT Core, and to apply these technologies in the field of the Internet of Things. To develop a simulation model using these technologies and test its ability to test several approaches to ensuring energy efficiency in home automation.

Research methods: analysis of sources in the field of testing IoT systems, as well as those related to the use of home automation to ensure energy efficiency, prototyping, simulation modeling, data analysis.

As a result, a program for simulation modeling of IoT systems using Node-RED and AWS IoT Core was developed. Based on the experiments, recommendations for the application of several approaches to energy efficiency are formulated.

Scope of application: for testing and research in the field of the Internet of Things, in particular, for assessing the quality of energy management systems.

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій в останні десятиліття призвело до появи такої концепції як інтернет речей. У той час як звичний інтернет пов'язує комп'ютери, в рамках нової ідеї до нього підключаються навіть звичайні речі: освітлювальне обладнання, побутові прилади, опалювальні системи, автомобілі та багато інших. Передбачається, що впровадження глобального інтернету речей призведе до далекосяжних наслідків. Стане можливим отримати доступ до даних у раніше небачених масштабах, що, свою чергу, призведе до оптимізації та автоматизації багатьох процесів реального світу. У певному сенсі перехід до інтернету речей є логічним продовженням інформатизації та комп'ютеризації всіх сфер життя.

За всіх переваг, які мають принести системи інтернету речей, при впровадженні подібних систем розробники стикаються з низкою складнощів. Як і будь-яка сфера, що розвивається, інтернет речей страждає від нестачі стандартизації і відсутності загальноприйнятих інструментів. Розробка систем інтернету речей пов'язана з унікальними складнощами, пов'язаними з їхньою зануреністю в потік даних навколишнього середовища. У цьому нерідко необхідно забезпечити коректність роботи систем, оскільки особливістю середовища роботи є високий рівень відповідальності. Тому необхідно дослідження підходів до тестування та вивчення таких систем. У цій роботі вивчається підхід, що ґрунтується на імітаційному моделюванні з використанням Node-RED та AWS IoT Core. Node-RED є гнучким програмним засобом, що дозволяє реалізувати різні методи імітаційного моделювання у поєднанні зі створенням прототипів систем та графічного інтерфейсу для проведення експериментів, тоді як AWS IoT Core дозволяє інтегрувати систему з хмарним сервером, передавати та обробляти дані.

Таким чином, завданням даної роботи є, з одного боку, аналіз технологій Node-RED та AWS IoT Core, а з іншого – розробка підходу до моделювання та тестування систем інтернету речей з їх використанням.

Розділ 1 дає загальну характеристику проблеми тестування систем інтернету речей. Аналізуються особливості таких систем, типові вимоги до них. Розглядаються можливі підходи до їхнього тестування.

Надається обґрунтування використання імітаційного моделювання. Аналізується роль парадигми потокового програмування та натхненного їй інструменту розробки Node-RED, а також хмарних технологій на прикладі AWS Cloud та сервісу AWS IoT Core у сфері інтернету речей, потенційні можливості та причини їх використання для імітаційного моделювання.

Розділ 2 дається більш детальна характеристика використання домашньої автоматизації для управління енергоспоживанням - прикладної області в рамках сфери інтернету речей. Розглядається середовище їхньої роботи, основні компоненти. Описуються типові вимоги та показники якості для таких систем. Аналізуються існуючі підходи.

Розділ 3 описує процес розробки системи тестування та моделювання систем інтернету речей з використанням Node-RED та AWS IoT Core, включаючи математичну модель процесів, що відбуваються, та способи її реалізації.

Розділ 4 описує приклад використання розробленої системи для проведення експериментів, що дозволяють порівняти кілька підходів, на основі яких робляться рекомендації щодо їх використання. Проводиться аналіз використання Node-RED та AWS IoT Core.

1 ПРОБЛЕМА ТЕСТУВАННЯ ТА ОЦІНКИ ЯКОСТІ У СФЕРІ ІНТЕРНЕТУ РЕЧІВ

1.1 Загальна характеристика Інтернету речей

Розвиток мікроелектроніки призвів до мініатюризації та здешевлення електронних компонентів, завдяки чому стало можливим вбудовувати комп'ютерні чіпи в різні речі, а також використовувати безліч мініатюрних датчиків. Комунікаційні компоненти таких пристроїв дозволили об'єднувати їх в одну мережу для обміну даними та командами. Так виникла концепція інтернету речей.

Кожна річ окремо може запропонувати деяку частку автоматизації: наприклад, лампочка в кімнаті може включатися за датчиком руху — проте в інтернеті речей предмети об'єднані в таку систему, яка може враховувати більшу кількість факторів одночасно. Наслідком такої конфігурації є широка доступність інформації для аналізу та прийняття рішень. Таким чином, інтернет речей дозволяє автоматизувати та оптимізувати процеси, до яких він застосовується, що багато в чому знімає навантаження та відповідальність з користувача, що особливо корисно у сферах, що потребують постійного контролю. Найчастіше системи інтернету речей призначені для надання фонової підтримки або стеження за різними щоденними процесами.

Як модель для опису роботи систем інтернету речей пропонується обробка потоку даних. Центральною частиною робочого циклу системи є реакція на події, що приходять, проте варто відзначити, що не всі датчики працюють за такою моделлю: іноді контролюючий пристрій отримують спостережувані дані через фіксовані проміжки часу. Таким чином, система поступово пропускає через себе дані, приймаючи рішення про відправлення команд підконтрольним пристроям по необхідності. Крім того, найчастіше окремою вимогою є зберігання одержуваних даних, щоб пізніше використовувати їх для аналізу.

Уникаючи абстракції потоку даних, варто відзначити, що системи інтернету речей зазвичай працюють у неоднорідному середовищі, де пристрої як складові

елементи являють собою мобільні гетерогенні об'єкти, оснащені датчиками та маніпуляторами. Кожен пристрій може відрізнятися від інших за функціоналом та доступними ресурсами, причому роль відіграє як програмне, так і апаратне забезпечення. Для зв'язку елементів використовується комунікативно-обчислювальна інфраструктура інтернету. Отже, потрібно забезпечення коректності як обробки даних, і взаємодії між різними частинами аналізованої розподіленої системи.

Враховуючи перелічені особливості, цікавить перевірка правильності функціонування такої комунікативно-обчислювальної системи її відповідність заданим вимогам. При цьому часто можливо відокремити технічні аспекти реалізації, такі як вибір конкретних пристроїв або протоколів взаємодії, щоб сконцентруватися на забезпеченні коректної поведінки програмної частини. Відзначимо також подібне завдання визначення придатності того чи іншого підходу, що виникає у процесі проведення досліджень у сфері Інтернету речей. Інакше кажучи, потрібно розробити метод, який дозволив би залежно від конкретної ситуації:

- перевірити систему або її прототип на відповідність заздалегідь визначеним критеріям,
- або дати оцінку якості роботи системи шляхом обчислення заданих показників.

Процес вирішення однієї з обох перерахованих задач для даного програмного продукту або прототипу називатимемо тестуванням.

Враховуючи, що центральне місце в процесі тестування в контексті даної роботи займає обчислення деяких показників для деякої системи, слід визначити, які вимоги пред'являються до подібних продуктів.

Існує безліч варіантів класифікації та систематизації можливих вимог до програмного забезпечення, так само як і показників, за якими визначається відповідність цією вимогою. У стандарті ISO/IEC TR 9126-2:2003 пропонується шість категорій метрик для оцінки поведінки програмних продуктів: функціональність, надійність, зручність використання, ефективність, можливість

супроводу та переносимість. Для розрахунку цих показників систему необхідно помістити до умов реального використання [19, с. 4]. Усі перелічені метрики можуть бути використані для тестування будь-якого програмного забезпечення, однак у конкретній сфері, зокрема — у сфері інтернету речей, деякі категорії корисніші за інші.

Перевірка програмного забезпечення на функціональність найчастіше займає основне місце у тестуванні. За однією з класифікацій вимоги поділяються на функціональні та нефункціональні, тобто всі інші. Фундаментально, під функціональністю розумітимемо відповідність результатів роботи системи очікуванням користувача. До цієї ж категорії відносять показники зовнішньої сумісності та безпеки системи [19, с. 6-7]. Таким чином, поняття функціональності саме по собі включає багато з основних критеріїв оцінки системи.

Функціональні вимоги можна назвати найважливішими й у контексті Інтернету речей. Завданням систем інтернету речей є автоматизація та оптимізація різних процесів, нерідко у сферах із високим ступенем відповідальності, де потрібна точна відповідність розробленим приписам, інакше наслідки можуть бути тяжкими. Тому потрібно забезпечити відсутність помилок у поведінці системи настільки, наскільки це можливо. Стороннє втручання також може бути проблемою: успішна атака, наприклад, на «розумний дім», потенційно може призвести до контролю зловмисником таких важливих для якості життя систем, як опалення або освітлення. Крім того, через системи інтернету речей нерідко проходять конфіденційні дані, з яких можна робити чутливі висновки про користувачів.

Серед нефункціональних вимог як особливо важливих можна виділити вимоги до надійності та ефективності. Незважаючи на те, що більшість систем інтернету речей не вимагають прийняття рішень у жорсткому реальному часі, реакція на події, що відбуваються, має бути своєчасною, відмова може призвести до катастрофічних наслідків. Для оцінки надійності пропонуються метрики зрілості, стійкості до відмов, здатності до відновлення, а для оцінки ефективності — метрики своєчасності реакції системи та завантаження ресурсів [19, с. 16, 46].

Додатковою причиною використання вимог ефективності є обмеженість ресурсів більшості пристроїв інтернету речей.

Інші категорії вимог, такі як зручність використання, можливість супроводу, переносимість, можуть також застосовуватися до систем інтернету речей, проте їхня автоматизація розрахунку таких показників можлива значно меншою мірою. Наприклад, під зручністю використання розуміється простота розуміння, вивчення та управління роботою системи [19, с. 28-29], що важливо і у сфері інтернету речей, проте для оцінки цих показників необхідне тестування реальними користувачами.

Підсумовуючи, відзначимо, більшість описаних вимог є широкими категоріями, а конкретної системи чи принаймні предметної області необхідно уточнення обраних метрик. Приклад такого аналізу буде описано у подальших розділах.

Як було описано раніше, у контексті даної роботи під тестуванням розуміється обчислення деяких показників, на підставі яких оцінюється якість роботи системи, можливо з проміжним кроком у вигляді перевірки відповідності її заздалегідь заданим вимогам. Така задача може виникати як за наявності готового продукту, для якого необхідно забезпечити коректність його поведінки, так і при попередньому дослідженні предметної галузі визначення найбільш перспективних підходів.

Середовище роботи систем інтернету речей часто таке, що зміни можна простежити лише на великих проміжках часу. Наприклад, для перевірки коректної реакції на сезонні фактори необхідно тестувати систему протягом року або більше, тому що менший час не охопить весь можливий спектр. Водночас деякі події настільки рідкісні, що вони можуть не зустрітися навіть у такому довгому дослідженні. Як приклад можна навести екстремально високі або екстремально низькі температури, які можуть зустрічатися один раз на кілька років.

Іноді можна протестувати готову систему з урахуванням усіх особливостей її фізичної реалізації. З іншого боку, нерідко продукт існує тільки у вигляді прототипу або взагалі не існує, а необхідно протестувати той чи інший теоретично

розроблений підхід. У разі побудова фізичної реалізації системи пов'язані з великими витратами ресурсів, зокрема грошових.

Зазначимо, що навіть у разі існування готової системи може мати сенс відокремлення її апаратної реалізації від програмного забезпечення.

Зокрема, можна відтворити деякі задані ситуації, щоб перевірити коректність деяких змін. Крім того, як було зазначено раніше, час дослідження може представляти проблему, якщо потрібно відстежити зміни, що відбуваються на великих часових проміжках.

У такому разі може мати сенс використання технологій, що дозволяють прискорити цей процес.

Незважаючи на ці складності, неможливо недооцінити значущість тестування у сфері інтернету речей. Високі вимоги до функціональності, надійності, ефективності роблять цю стадію необхідною.

З іншого боку, може бути зручним, особливо на початкових етапах розробки, відійти від фізичної реалізації системи та використовувати програмні технології.

У разі використання тих чи інших технологій симуляції, емуляції чи імітації варто звернути увагу на те, що потрібна побудова окремої тестової інфраструктури для вирішення цього завдання. Нерідко потрібна певна адаптація систем, що тестують і тестуються, для їх спільної роботи.

Нарешті, зазначимо, що складність може представляти розробка таких вимог чи критеріїв тестування, які б враховували кілька цілей, що конфліктують між собою.

Найчастіше завдання систем інтернету речей включають кілька приписів, між якими потрібно знайти компроміс, причому на етапі дослідження нерідко саме співвідношення ще не визначено.

Таким чином, необхідно забезпечення певної гнучкості системи тестування, щоб було можливо без труднощів переходити між різними показниками якості.

1.2 Тестування систем Інтернету речей

Підсумуємо описане, коротко перерахувавши основні особливості систем Інтернету речей. Такі системи тісно занурені у навколишнє середовище та постійно отримують інформацію про неї через датчики. Як моделювання показань цих датчиків, так і оцінка роботи системи неможливі в ручному режимі через великий об'єм та відсутність людино-машинних інтерфейсів. Ці особливості призводять до того, що всебічна оцінка роботи програмних продуктів у сфері систем інтернету речей серед їх кінцевого розгортання вимагає великої кількості ресурсів, насамперед фінансових і тимчасових. Ще складніше виявляється вирішити завдання дослідження гіпотез щодо перспективних підходів.

Таким чином, виникає необхідність розробки особливої програми тестування, в рамках якої могли б проводитися експерименти над тестованою системою, тобто порівняння різних сценаріїв її функціонування за різних умов навколишнього середовища. Виникає кілька ступенів свободи, якими можна порівнювати поведінка системи: зміна умов зовнішньої стосовно системи середовища, зміна поведінки чи можливих станів керованих пристроїв, зміна алгоритмів, якими приймає рішення тестована система. Важливо, що при тестуванні такої системи потрібно детальне відтворення показників, пов'язаних з усіма процесами, що належать до системи протягом тривалого віртуального часу.

Імітаційне моделювання якраз і є різновидом моделювання, що дозволяє задовольнити описані вимоги: детальне відтворення процесів, тривалий період віртуального часу, відсутність обмежень на структуру та обсяг вхідних та вихідних даних [1].

Розглянемо деякі методи імітаційного моделювання, які є корисними у сфері інтернету речей.

Метод системної динаміки система представляється як сукупність потоків, накопичувачів, допоміжних змінних і підмоделей зі своїми елементами. При цьому підході реалізується математична модель, що є динамічною системою одночасних

рівнянь, інтегрована з джерелами статистичних даних [2, с. 17-18]. Наприклад, такі параметри, як температура та вологість у приміщенні, можуть бути представлені як функції кількох змінних: зовнішньої температури, вологості, присутності людини, стану системи опалення тощо. У процесі моделювання кожен момент необхідні показники розраховуються за цими формулами. У системі також може бути представлений зворотний зв'язок [2, с. 23].

У попередньому прикладі температура в приміщенні впливає на керування опаленням, яке в свою чергу впливає на температуру.

Агентний метод полягає в тому, щоб індивідуально задати поведінку окремих агентів системи, з якої виникає глобальна поведінка системи. Під агентами розуміються різноманітні елементи системи: соціальні, економічні, технічні, екосистемні. Для кожного агента визначається безліч його станів, переходи між ними, що викликають їх події, тимчасові затримки, чинні ними дії [2, с. 37-38]. Наприклад, агентом, що діє у системі «розумний будинок», може бути мешканець цього будинку. У найпростішій моделі можна уявити два стани «вдома» і «не вдома», між якими агент переходить після досягнення обраного віртуального часу.

Можливі також гібридні методи, за яких у процесі моделювання поєднуються кілька підходів. У контексті інтернету речей найбільш підходящим є такий спосіб. Зокрема, окремі елементи системи можуть бути представлені як агенти, стан яких розраховується, за допомогою системної динаміки.

Завершення розділу коротко торкнемося інших підходів, елементи яких можуть бути інтегровані в таку гібридну систему. По-перше, це дискретно-подійна парадигма, в якій фокусом моделювання є ресурси, черги, затримки, переходи по подіях [2]. У гібридній моделі може використовуватись властивість дискретності такої моделі замість безперервних потоків системної динаміки. Для обліку в системі недетермінованості довкілля можуть бути використані методи стохастичного моделювання, в якому стан системи описується, зокрема, і випадковими величинами [2]. Для підбору найбільш правильної моделі можна використовувати еволюційне моделювання.

Перш ніж перейти до обговорення використання Node-RED та AWS IoT Core для імітаційного моделювання, проведемо короткий аналіз одного з найпоширеніших існуючих рішень у цій галузі. Simulink - це візуальне середовище для проектування та моделювання, що включає графічний редактор з різними компонентами для побудови моделей, а також інтегрована з MATLAB для підтримки чисельних методів рішення [17].

Створення моделі у цьому середовищі ведеться шляхом з'єднання блоків, відповідальних прості операції, такі як підсумовування, множення на константу, інтегрування за часом та інші. Після цього користувач може запустити моделювання та переглянути результати як у вигляді файлів з даними, так і графічно.

Завдяки використанню MATLAB середовище Simulink може запропонувати високу точність моделювання, зрозуміло, за умови правильної розробки моделі. Наприклад, система може підібрати крок автоматично, щоб забезпечити максимальну точність. Крім того, в середовищі Simulink доступний широкий вибір компонентів моделей, що дозволяє застосовувати її практично в будь-якій предметній області.

Тим не менш, це рішення не позбавлене проблем, насамперед пов'язаних відносною складністю входження, навіть попри використання візуального подання. Як професійний продукт, Simulink також відрізняється відносно високою вартістю, особливо для розробників або дослідників, які не мають доступу в рамках деякої організації.

Що стосується процесу розробки та тестування, модель, розроблена в Simulink, не може бути використана безпосередньо, більше того, для існуючої системи потрібна розробка окремої моделі.

Всі ці фактори призводять до того, що використання Simulink у деяких випадках виявляється неможливим чи неефективним.

1.3 Огляд технологій інтернету речей та можливостей їх застосування в імітаційному моделюванні

З аналізу імітаційного моделювання як підходу до тестування інтернету речей, а також існуючих рішень цього завдання, можна виділити деякі фактори, на які варто орієнтуватися при виборі технологій створення середовища симуляції. У тому числі, потрібно виділити недоліки існуючих рішень, які, можливо, роблять їх непридатними для того чи іншого завдання. У такому разі розробка нового підходу чи системи є особливо актуальною.

Інструменти для імітаційного моделювання повинні дозволяти реалізовувати широке коло сценаріїв тестування та пристроїв, що емулюються, а також розраховувати різні оцінки якості. При цьому створена система має бути легко змінюваною та доповнюваною. Через це найважливішим критерієм вибору технології є спектр її можливостей.

Simulink є найпоширенішим інструментом для імітаційного моделювання, проте він має високий поріг входження як у сенсі необхідних навичок, так і засобів. Тому хотілося б розробити інтуїтивніше зрозумілий підхід. При цьому візуальне середовище бачиться як перевага, хоча тільки її недостатньо. Крім того, процес розробки з боку користувача спрощується, коли є доступні компоненти для вирішення деяких завдань у форматі бібліотеки, як додаткову вимогу можна також висунути можливість перенесення розробленої системи між різними середовищами виконання, у тому числі спрощений перехід від прототипу до кінцевого продукту.

Зрештою, при аналізі Simulink ще одним недоліком слід назвати ціну, високий розмір якої пов'язаний з моделлю доступу та розповсюдження. На противагу цьому середовищу бажано використовувати програмні засоби з відкритим вихідним кодом або принаймні з оплатою лише реально спожитих ресурсів.

Традиційно, відповідно до принципів фон Неймана, комп'ютерна програма моделювалася як послідовність операцій, що відбуваються в певному порядку [6].

У такій моделі фокусом є команди, тоді як дані статичні. Кожна команда може змінити значення, але з положення даних, після чого процесор переходить до наступній команді. Пам'ять, що розділяється, в моделі фон Неймана не становить проблем для однопотокових програм, проте призводить до складнощів при спробі розпаралелювання. В якості альтернативи було запропоновано архітектуру, засновану на потоках даних. Програма у разі представляється як спрямований граф, у якому дані «перетікають» між командами [6]. Кожна окрема операція функціонує як чорна скринька з чітко визначеними входами та виходами.

Розвитком ідеї програмування потоків даних є потокове програмування. У цій парадигмі програму представляють як мережу процесів, внутрішній зміст яких ігнорується. У мережі визначаються з'єднання між процесами, якими можуть передавати порції даних. Розроблені з використанням описаної вище структури програми є слабко пов'язаними. Вузли в мережі пов'язані тільки тим, який формат даних вони приймають на вхід та виробляють на виході.

Node-RED – це візуальний інструмент розробки, що базується на парадигмі потокового програмування. Він був розроблений компанією IBM та написаний мовою програмування JavaScript. У Node-RED додаток представляється як мережа про вузлів. Кожен вузол має певну мету: на вхід надходять деякі дані, вони обробляються всередині, після чого передаються далі мережею. За переміщення даних між вузлами відповідає мережа [15].

Для об'єднання та організації вузлів в єдину концептуальну одиницю Node-RED використовується потік. Потік представляється як окрема сторінка у редакторі та пропонує проміжний рівень контексту між окремим вузлом всією системою загалом. Не всі вузли в рамках потоку повинні бути пов'язані між собою, проте слово «потік» також може використовуватися для позначення набору зв'язаних між собою вузлів.

Кожен вузол у потоці очікує або повідомлення від попереднього вузла, або якоїсь зовнішньої події, не пов'язаної з діяльністю потоку. В якості приклад зовнішньої події можна навести HTTP-запит. Після отримання повідомлення або виникнення події вузол переходить до обробки. Завершивши її, вузол може

надіслати повідомлення. Кожен вузол може мати лише один вхідний порт, але будь-яку кількість вихідних портів. Вузли надсилають одне одному повідомлення, які є JavaScript-об'єктами. З'єднання між вузлами називають дроти.

Крім повідомлення, інформація може передаватися між вузлами через контекст. У Node-RED визначено три рівні контексту: рівень вузла, коли інформація доступна лише вузлу, що її виробив, рівень потоку, коли інформація доступна всім вузлам потоку, та глобальний [15].

Середовищем виконання Node-RED є Node.js. Розробка також ведеться через веб-браузер. Всередині редактора потоків користувач може додавати вузли та з'єднувати їх проводами. Також є можливість імпортувати та експортувати потоки у вигляді JSON-файлів. Node-RED допускає реалізацію своїх вузлів JavaScript, для візуалізації використовується HTML.

Усередині кожен вузол реалізує шаблон "Спостерігач". Завдяки цьому вузли, з'єднані проводом, можуть повідомляти про події. Вхідні вузли в потоці можуть підписуватись на зовнішні сервіси, прослуховувати порти або обслуговувати HTTP-запити. Вузол-обробник отримує дані та здійснює з ними задані дії, після чого повідомляє наступні вузли за допомогою подій. Вузол також може генерувати додаткові події, встановлювати зв'язок із зовнішніми сервісами чи операційною системою [21]. Також існують вузли конфігурації, в яких можна зберігати загальну для потоку конфігураційну інформацію.

Програму Node-RED можна розділити на три рівні: глобальний, рівень потоку та рівень вузла. На глобальному рівні знаходиться вся програма: всі вузли, їх параметри та зв'язки між ними. Запуск на виконання та збереження глобального контексту здійснюється одночасно. При цьому виконання безперервно, можливе лише оновлення та перезапуск усієї програми одночасно. Кожен вузол може прийняти дані будь-якої миті і всі вузли працюють паралельно.

Рівень потоку грає роль двох випадках. По-перше, змінні, що розділяються між декількома вузлами, можуть бути збережені в контекст потоку. По-друге, поділ програми з кількох потоків, представлених окремими вкладками у візуальному

редакторі Node-RED, допомагає у структуризації. Зауважимо, що це потоки виконуються одночасно, хоча деякі потоки може бути відключені.

Нарешті, особливий інтерес становить рівень вузла. Якщо не зважати на можливість додати самостійно розроблений вузол, найбільш загальним за своїм функціоналом вузлом є Function. У такому вузлі можна записати будь-який алгоритм за допомогою JavaScript, причому можна визначити різну поведінку для запуску, зупинки та кожної активації вузла. Такий вузол може мати кілька виходів.

Для деяких поширених завдань замість вузла Function можна використовувати вузли більш вузького призначення, які узагальнюють деяку операцію. Зокрема, одним із таких вузлів є вузол Change, який дозволяє змінювати властивості повідомлень та встановлювати значення змінних контексту. Для того, щоб надіслати повідомлення в одну з кількох гілок потоку в залежності від вхідних даних або змінних контексту, використовується вузол Switch. Цей вузол відправляє повідомлення у всі свої виходи, що відповідають зазначеним у ньому правилам, хоча він може бути заданий і таким чином, щоб надсилати тільки за першим відповідним правилом.

Водночас існують і спеціалізовані вузли, призначені для вирішення певної задачі: розбір того чи іншого формату даних, обробка запитів з використанням заданого протоколу, файлового виводу-введення-виводу, відображення об'єктів графічного інтерфейсу. У Node-RED також можна підключати сторонні колекції вузлів для завдань, для яких не вдалося знайти відповідний вузол в колекції за замовчуванням.

Крім візуального редактора, у розпорядженні користувача Node-RED також є кілька інших уявлень, що допомагають аналізувати поведінку системи. Найбільш універсальним є так звана приладова панель, на якій демонструються елементи графічного інтерфейсу, що задаються спеціальними вузлами. При цьому можна як виводити інформацію на екран у вигляді тексту, графіків та зображень, так і використовувати такі елементи керування, як кнопки та поля введення.

Node-RED пропонує можливість ручного запуску повідомлення по потоку за допомогою вузла Inject, а також читання повідомлень, що передаються, за

допомогою вузла Debug. Усі повідомлення, що зчитуються вузлом Debug, відображаються на окремій панелі налагодження. Можна додати такі кілька вузлів.

При використанні у сфері інтернету речей Node-RED пропонує кілька переваг у порівнянні з іншими варіантами: візуальне подання, різноманітність підтримуваних платформ, можливість абстракції щодо фізичної реалізації системи. Зрозуміло, як і будь-який додатковий шар системи Node-RED вносить додаткові витрати на пересилання та обробку даних, але у більшості систем це допустимо. Крім того, завдання деяких систем можуть бути занадто складні для функціонала, запропонованого Node-RED, хоча це частково і може вирішуватися доповненнями користувача. Незважаючи на це, інструмент Node-RED добре підходить для розробки систем інтернету речей, причому його переваги в цій галузі також є сильними сторонами з точки зору моделювання та тестування таких систем. Докладніше опишемо можливий метод використання Node-RED у цій галузі.

Сама система інтернету речей представляється як окремий потік Node-RED, хоча вона може бути реалізована незалежно і просто бути підключена через мережні вузли. В окремому потоці чи потоках реалізується програма щодо імітаційних експериментів. До неї входять джерела даних, генератори різних показників, у тому числі і системного часу, а також вузли, що відповідають за комунікацію з системою, що тестується, та аналіз її поведінки.

Використання Node-RED не накладає обмежень на формат даних, крім того, що він має бути представлений як об'єкт JavaScript. Велику труднощі становлять обмеження щодо частоти подій. Насправді було випробувано, що з посиланні сигналу частіше, ніж 10 разів у секунду, у системі з'являються видимі затримки. Таким чином, при тимчасовому вирішенні всередині системи в одну годину можна досягти прискорення порівняно з реальним часом 36000 разів, а за тимчасового дозволу за хвилину — лише 600. Подальше зменшення періоду віртуального часу між двома сигналами має обмежене практичне застосування, оскільки масштаби необхідних тимчасових проміжків експериментів у системах інтернету речей досягають періодів від дня до року.

Node-RED є відповідним інструментом для імітаційного моделювання, оскільки він пропонує велику різноманітність різних вузлів, а також способи з об'єднання таким чином, щоб відтворити практично будь-які необхідні показники. Завдяки використанню візуального програмування побудова складних зв'язків між показниками спрощується з погляду розробника, а водночас і користувачеві легше розуміти будову системи. Завдання тестування додатково спрощується, якщо сама система, що тестується, була реалізована за допомогою Node-RED.

Сервіс AWS IoT Core був запущений компанією, як складова хмарна платформа AWS Cloud, також відома як Amazon cloud. Цей сервіс пропонує рішення для Інтернету речей у форматі «програмне забезпечення як послуга». У такій моделі постачальник надає готове програмне забезпечення, яке він повністю обслуговує сам, а користувач може отримати доступ до його функцій. Таким чином, цей сервіс дозволяє значно скоротити витрати на розробку та створення інфраструктури інтернету речей.

Як було описано раніше, більшість систем інтернету речей і двох частин (рисунок 1.1). Перша з них пов'язана з передачею даних і команд між сховищем та пристроями, а друга полягає у роботі з отриманими даними. При цьому перша частина одночасно характеризується чіткими дисфункціями, для задоволення яких добре підходять хмарні сховища, і великою схожістю серед різних систем інтернету речей. Тому команда при розробці сервісів для використання у сфері інтернету речей сфокусувалася в першу чергу на наданні можливості швидко та надійно отримувати дані від пристроїв та надсилати їм команди.

Крім самого AWS IoT Core, в екосистемі сервісів для інтернету речей також є такі сервіси, як AWS Cloud Functions та AWS Object Storage. Сервіс IoT Core є MQTT-брокером з набором додаткових функцій. Сервіс Cloud Functions дозволяє запускати програми користувача для обробки даних без створення віртуальних машин. Сервіс Object Storage дає можливість зберігати великі масиви даних, у тому числі архівні записи, накопичені за великий період роботи системи.

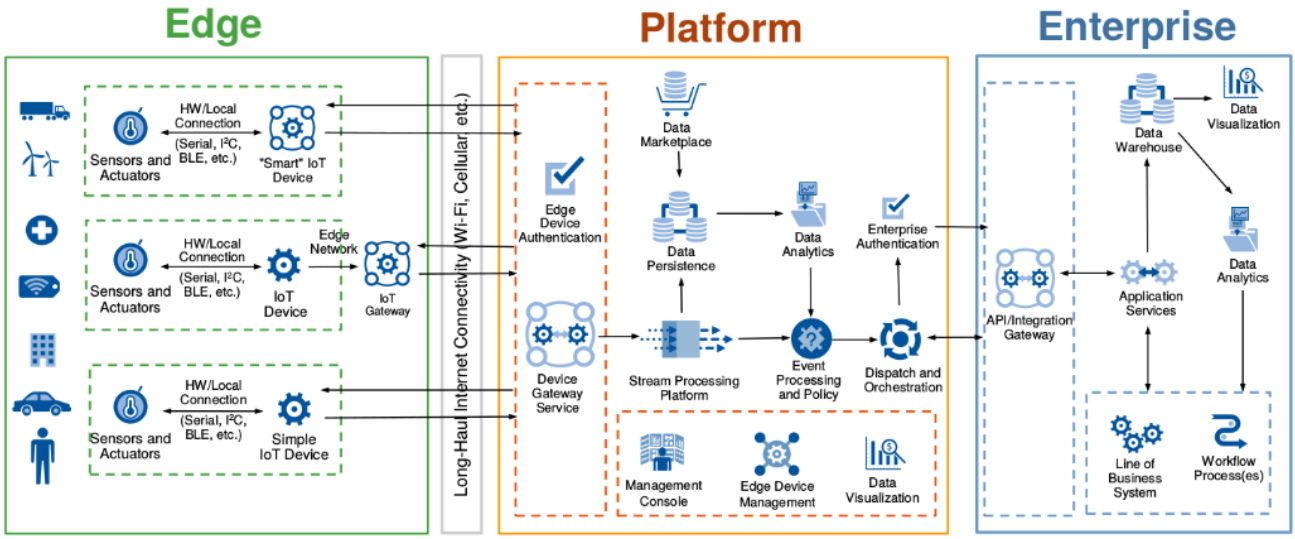

Рисунок 1.1 - Архітектура системи інтернету речей з використанням хмарних технологій

На додаток до цих сервісів у AWS Cloud також є пропозиції для зберігання та аналізу даних, наприклад AWS Managed Service for ClickHouse.

Основним сервісом AWS Cloud, що пропонує рішення в галузі інтернету речей, є AWS IoT Core. Цей сервіс призначений для двостороннього обміну даними між хмарою та пристроями з використанням протоколу MQTT. У цьому протоколі передачі даних використовуються іменовані черги, інакше звані темами чи топіками, у які можна записувати дані. Підписавшись на події черги, об'єкти можуть асинхронно отримувати дані з неї.

Основними елементами AWS IoT Core є пристрій та реєстр, які обмінюються даними та командами через MQTT-брокер [4]. Під пристроєм розуміється екземпляр фізичного пристрою інтернету речей. Пристрій може надсилати телеметричні дані та отримувати команди. Реєстр

Це набір логічно пов'язаних між собою пристроїв. Пристрої в реєстрі можуть взаємодіяти між собою, надсилаючи дані та отримуючи команди. Реєстр може читати дані та надсилати команди.

MQTT-брокер відповідає за обмін MQTT-повідомленнями між пристроями та реєстрами. З одного боку, він отримує та обробляє повідомлення, а з іншого, контролює їх доставку MQTT-клієнтам, тобто пристроям та реєстрам. Обмін

даними може відбуватися лише через нього. Кожне повідомлення містить тему, або топік, які класифікують дані. Клієнти отримують лише повідомлення з тих тем, куди вони підписані. MQTT-брокер забезпечує необхідний рівень якості обслуговування під час обміну повідомленнями. AWS IoT Core підтримує два рівні: QoS 0, коли повідомлення надсилається лише один раз і доставка не гарантується, і QoS 1, коли повідомлення обов'язково доставляється щонайменше один раз, проте допускається отримання дублікатів.

Щоб пристрої та реєстр могли отримувати повідомлення через MQTT-брокер, вони повинні бути підписані на потрібні топики. Для особливо важливих повідомлень, наприклад показань датчиків, що вимагають негайної реакції, особливо якщо можлива втрата з'єднання, AWS IoT Core пропонує перманентні топіки, які зберігають останнє повідомлення, відправлене до нього. Це повідомлення буде відображено при відновленні з'єднання, навіть якщо в цей момент не було опубліковано нових повідомлень [4].

Що стосується імітаційного моделювання, AWS IoT Core доповнює Node-RED, створюючи можливість комунікації з системою, що тестується. Швидше за все, пристрої у фізичній реалізації системи інтернету речей у будь-якому

У разі будуть пов'язані з керуючим центром. У такому випадку достатньо лише адаптувати систему, замінивши існуючі вузли комунікації на вузли зв'язку з AWS IoT Core. При цьому хмарні технології дозволяють розширити можливості тестуючої системи: наприклад, можливо зберігати результати симуляції в базі даних або запускати ті чи інші дії в залежності від команд, що приходять. Обчислювальні ресурси, доступні на сервері хмар, також можуть бути корисні, наприклад для більш глибокого аналізу даних, створення більш досконалих алгоритмів з використанням машинного навчання.

Висновки. Перший розділ був присвячений розгляду та вирішенню наступних проблем:

- проведено аналіз сфери інтернету речей. Виділено основні особливості систем інтернету речей, що впливають на процес їхнього тестування;
- розглянуто основні підходи до тестування систем Інтернету речей;

- обґрунтовано використання імітаційного моделювання;
- розглянуто основні технології, що використовуються у роботі.

2 ЗАДАЧА ЗАБЕЗПЕЧЕННЯ ЕНЕРГОЕФЕКТИВНОСТІ РОЗУМНОГО БУДИНКУ

2.1 Система керування енергоспоживання

Зміна клімату, викликана викидами парникових газів в атмосферу, залишається однією з найважливіших проблем, які стоять перед людством нині. Як основні напрями її вирішення розглядається використання поновлюваних джерел енергії з одного боку та підвищення енергоефективності з іншого. Інакше ці два шляхи можна охарактеризувати як зміну підходу до виробництва та споживання відповідно.

Домогосподарства роблять значний внесок у загальне енергоспоживання. За даними Національного статистичного комітету, структурі кінцевого споживання паливно-енергетичних ресурсів у 2019 році житловий сектор займав майже 27% [5,]. Таким чином, оптимізація енергоспоживання у цій сфері є доцільною.

Для підвищення енергоефективності домогосподарств було розроблено безліч технологій, наприклад, теплоізоляція стін, даху, ефективніші котли опалення. Традиційно більшість рішень було «статичною», тобто впливати на їхню поведінку після встановлення практично неможливо. На противагу їм нині набирають популярність інтерактивні підходи, до яких належать технології, які потребують налаштування та взаємодії з людиною після встановлення.

Питання, який із підходів ефективніший, залишається відкритим. Деякі дослідження показують, що прості інтерактивні технології, такі як термостати, що програмуються, не тільки не знижують енергоспоживання, але навіть збільшують його [16]. Найбільш імовірною причиною цього ефекту є те, що користувачі часто

використовують їх неправильно, навмисно чи ні. Інтерактивні технології в цілому більш вимогливі до користувача, що часто призводить до їх неправильного використання [11].

З іншого боку, під час використання статичних технологій завжди залишається можливість поліпшити енергоефективність шляхом використання інтерактивних технологій. Дослідження так званих «зелених» будівель показали, що вони страждають від жорстких схем енергоспоживання, проте навіть прості схеми автоматичного управління дозволяють знизити витрати енергії [7]. Крім того, аргументом на користь розвитку інтерактивних технологій є можливість взяти контроль над споживанням енергії, яку вони пропонують користувачеві.

Використання інтернету речей дозволяє піти далі у двох важливих напрямках. З одного боку, використання множини датчиків дає можливість контролювати велику кількість різних показників, щоб забезпечувати максимальний комфорт за мінімальною ціною. З іншого, відносно потужніші комп'ютери, потенційно хмарні сервери, дозволять досягти значної міри автоматизації і зняти з користувача частину відповідальності за правильне налаштування системи.

Загальна концепція використання домашньої автоматизації в контексті забезпечення енергоефективності проста: система інтернету речей керує різними побутовими приладами, ґрунтуючись на даних про довкілля. Її завданням є досягнення максимального рівня комфорту за мінімального використання енергії.

Коротко торкнемося особливостей. У такому контексті зручно класифікувати побутову техніку залежно від того, чи можливо планувати її роботу, чи інакше кажучи, чи пристрій може працювати в будь-який час і без участі людини. Це зроблено, щоб система могла призначати роботу таких приладів на зручний час. Крім того, система може бути доповнена більш рідкісними пристроями локального виробництва та зберігання енергії, що дозволяє скорочувати споживання мережі.

Розвитком ідеї домашніх систем управління енергоспоживанням є нова модель виробництва та розподілу енергії — так звана «розумна енергосистема», що

характеризується двонаправленим потоком даних та енергії між її елементами. Усі вузли мережі можуть виробляти енергію, віддаючи її за потреби. У такій системі можна реалізувати динамічне управління всіма етапами виробництва енергії і потенційно наблизитися до максимально можливої ефективності.

Системи домашньої автоматизації призначені для використання в різних будинках, і системи управління енергоспоживанням - не виняток. Коли йдеться про продукти, призначені для використання одному домогосподарстві слід розрізняти квартири та індивідуальні житлові будинки, оскільки допустимі стратегії регулювання енергоспоживання, а також можливі пристрої, що встановлюються, різняться. Щодо квартир, оптимізація споживання енергії значно утруднена, оскільки багато комунальних послуг надаються централізовано або принаймні спільно з іншими мешканцями будинку. Доцільно орієнтуватися на повний спектр можливостей, доступних в індивідуальних житлових будинках, враховуючи при цьому обмеження щодо їх застосування в квартирах.

Для визначення найперспективніших напрямів розглянемо структуру кінцевого споживання енергії у домашніх господарствах. За даними Національного статистичного комітету, на опалення в середньому витрачається 59,2% від сумарної використаної енергії, на гаряче водопостачання – 17,8%, на приготування їжі – 13%, на побутові прилади та освітлення – 10% [3]. Структура енергоспоживання за різними типами житлових будинків показана у таблиці 2.1. Зауважимо, що витрати енергії на опалення ще вищі за інші при розгляді основного цільового середовища — індивідуальних житлових будинків. Водночас зазначимо, що для квартир із центральним опаленням більшу відносну вагу мають інші напрями використання енергії.

Таким чином, здається розумним концентруватися в першу чергу на опаленні, а потім - на гарячому водопостачанні, для чого є корисними рішення, що регулюють роботу пристроїв на основі показань датчиків про навколишнє середовище. Використання систем управління енергоспоживанням нерідко передбачає перехід на автономне опалення та водопостачання, оскільки у такому разі можливий більший контроль за використанням енергії.

Таблиця 2.1 - Структура загального кінцевого споживання енергії в домашніх господарствах [3]

	Будинки з автономним опаленням	Будинки з центральним опаленням	Квартири з автономним опаленням	Квартири з центральним опаленням
Опалення	70,6	73,4	66,6	52,0
Гаряче водопостачання	9,8	9,2	10,9	23,0
Приготування їжі	10,8	9,6	12,7	14,3
Побутові прилади	8,8	7,8	9,8	10,7

Разом з тим, незважаючи на відносно невелику їх частку у загальній структурі споживання енергії домогосподарствами, не варто ігнорувати й інші напрямки: приготування їжі, побутові прилади, освітлення.

Основними причинами є поширеність деяких простих оптимізаційних рішень, до яких належать, зокрема, автоматичне керування освітленням, планування використання побутових приладів.

Однією з важливих переваг систем управління енергоспоживанням на основі технологій інтернету речей є широта їхнього проникнення в різні аспекти довкілля та повсякденного життя користувачів. На відміну від вузьконаправлених рішень, таких як програмовані термостати або включення-вимикання світла по датчику руху, системи домашньої автоматизації здатні враховувати більшу кількість факторів. Щоб щось подібне було можливим, необхідно забезпечити достатню гнучкість системи для підключення до неї різних пристроїв, від котлів опалення до електроплит, а потенційно та відновлюваних джерел енергії, локальних засобів зберігання та інших приладів, які традиційно не належать до побутової техніки.

Зауважимо, що пристрої, що підключаються до інтернету, повинні бути здатними як мінімум до комунікації по мережі та автоматичному управлінню, причому можуть використовуватися команди, які не мають прямих аналогів серед

можливостей ручного управління. Прямий централізований контроль не є обов'язковим, як альтернатива система управління енергоспоживанням може постачати пристрої достатніми даними, з використанням яких вони повинні самі приймати рішення [14]. Корисним може виявитися доповнення побутової техніки додатковими датчиками, значення яких також передаються по мережі. Таким чином, потрібна широка модифікація побутових приладів, що використовуються.

Іншою перевагою є те, що завдяки вбудованим процесорам стає можливим використовувати програмне забезпечення для автоматизації, наприклад складання розкладу роботи пристроїв опалення. Потенційно комунікативно-обчислювальні можливості інтернету речей дозволяють навіть пов'язувати групи домашніх господарств та постачальників енергії в системи, де досягається ще більша ефективність енергоспоживання за рахунок широкої доступності даних про процес виробництва енергії, вимоги та шаблони поведінки користувачів.

тоді як підвищення ефективності роботи системи бажано підключити до неї максимальну кількість пристроїв і датчиків, слід враховувати, що з погляду користувачів потрібне мінімальне втручання у життя. Зрозуміло, ступінь допустимого може відрізнятися, але для того, щоб забезпечити максимальне охоплення, найкращою стратегією відштовхуватиметься від суворіших вимог. Важливо, що тут є два аспекти, що частково суперечать один одному. Для простоти використання ціль розробників повинна полягати в автоматизації якомога більшої кількості завдань, щоб користувачеві не доводилося приймати рішення, пов'язані з необхідністю глибокого вивчення предметної галузі, аналізу свого способу життя та подібних областей. Щоб підвищити якість роботи такої системи, необхідно збільшити кількість даних, що отримуються, а також ступінь керованості підконтрольних їй елементів, що пов'язано з впровадженням нових або заміною існуючих пристроїв і датчиків. З іншого боку, крім зручності використання користувачів, також цікавить простота входження, для чого необхідність переобладнання житла має негативні наслідки.

Як завершення обговорення середовища роботи систем управління енергоспоживанням відзначимо, що в перспективі розроблені для індивідуальних

домогосподарств технології можна розширити на інші типи будівель: багатоквартирні будинки, офісні та виробничі будівлі та інші. Насамперед це стосується засобів забезпечення комфортного перебування у приміщенні: підтримка температури, вологості повітря, освітлення, водопостачання тощо. Для визначення можливостей оптимізації енергоспоживання за іншими напрямками потрібен додатковий аналіз.

У літературі спостерігається значне різноманітність пристроїв, запропонованих для використання у тих систем управління енергоспоживанням. Найчастіше розглядаються пристрої ОВіК - опалення, вентиляції та кондиціювання - такі як обігрівач повітря, водонагрівач, кондиціонер. Дещо рідше включається освітлення [14]. Що стосується побутової техніки, рідко включаються пристрої, які працюють на вимогу користувача, наприклад, комп'ютер. Часто зустрічаються такі прилади, як холодильник, пральна машина, посудомийна машина, духовка [14]. Таким чином, вибір пристроїв узгоджується з аналізом найперспективніших напрямів споживання енергії.

Деякі дослідження доповнюють традиційні пасивні домашні господарства технологіями виробництва та зберігання енергії. Прогнозується, що з часом вони стануть доступнішими [8]. Рішення, що включають засоби зберігання енергії: акумулятори, батареї, електро- та гібридні автомобілі - зустрічаються дещо частіше, ніж враховують внесок ВІЕ (сонячні батареї, вітрогенератори) [14]. Такі пристрої, як і раніше, мало поширені, проте їх включення є перспективним для цілей екологічно чистого розвитку.

У той час як ВІЕ, електромобілі, акумулятори рідкісні, але доступні для звичайного користувача, деякі пропоновані компоненти та стратегії ґрунтуються на впровадженні інших технологій. Частим припущенням є тарифікація різних видів комунальних послуг відповідно до поточного попиту та пропозиції, з чого випливає підключення до енергосистеми через інтелектуальний лічильник, що поєднує індивідуальні домогосподарства у глобальну мережу. Впровадження цієї технології дозволить скоротити втрати енергії, удосконалити планування виробництва енергії з урахуванням прогнозованого попиту, скоротити витрати

користувачів, зробити технічне обслуговування ефективнішим, зменшити екологічні збитки [12]. Незважаючи на передбачувані переваги, поки що невідомо, чи йтиме розвиток енергосистем у цьому напрямку, тому практичне впровадження в рамках індивідуальних домогосподарств поки що неможливе.

Отже, можна назвати такі можливі фізичні компоненти: побутова техніка, джерела енергії, пристрої зберігання енергії, розумні комунальні послуги. Список наведено за зменшенням доступності. Щодо практичної реалізації подібних систем, про її можливість можна говорити лише в тому випадку, коли її компоненти не вимагають появи тих чи інших нових технологій чи підходів до виробництва та розподілу енергії.

Для ефективного управління підконтрольними пристроями в систему також включаються датчики, які зчитують такі показання навколишнього середовища, як температура, вологість, освітленість тощо. Конкретний набір датчиків залежатиме від вимог до системи та доступних у ній пристроїв. Додатково можуть застосовуватися технології визначення перебування людини приміщенні за допомогою датчиків руху або розташування.

Що стосується логічної архітектури систем управління енергоспоживанням, як основні компоненти можна виділити відстеження та зберігання даних про стан пристроїв, управління та планування їх роботи, а також оповіщення про небажані ситуації [18]. При цьому передбачається, що система може навчатися, тобто виділяти шаблони поведінки користувачів, тенденції споживання та виробництва енергії. Нарешті, система надавати інтерфейс користувача для спостереження та управління, коректно обробляти команди.

2.2 Вимоги та показники якості

Середовище роботи систем управління енергоспоживанням — житлове приміщення, домогосподарство, навіть офісний чи виробничий будинок —

призводить до багатоцільового характеру таких систем [14], тобто до необхідності знаходження компромісу між кількома завданнями. У той час як глобальною метою завжди є підвищення ефективності використання енергії, її досягнення накладаються обмеження, зокрема, за комфортом користувачів. Крім того, формулювання конкретного завдання може різнитися залежно від контексту: зниження викидів, грошова економія, балансування навантаження на енергосистему — деякі можливі напрямки.

Цілі управління споживанням енергії можуть бути виражені через вартість. Хоча найбільш очевидним і переважним чинником є ціна споживаної енергії, до загальної формули розрахунку можуть включатися такі чинники, як витрати на початкову установку системи, штраф за внесок у загальне навантаження, прогнозований знос устаткування, податку викиди парникових газів [14]. Таким чином, система може наслідувати єдину мету - зниження одержуваних за цією формулою грошових витрат. Вибір коректного співвідношення може становити складність, якщо немає усталених грошових значень деяких цілей.

Основне завдання з тих, які складно уявити в грошовому еквіваленті, - це комфорт користувачів, тобто їх незручності, пов'язані з якістю послуг, що надаються при доставці енергії. Система не повинна призводити до значної зміни їхнього способу життя. Для оцінки впливу системи на комфорт користувачів можуть використовуватись різні штрафні функції: обмеження за значенням, відхилення, штраф за відсутність сервісу [14]. Інакше кажучи, якщо вартість є деяким співвідношенням, яке потрібно мінімізувати, то вимоги щодо комфорту накладають обмеження на можливі стратегії.

Описані вище міркування можуть застосовуватись у тестуванні загальних підходів до оптимізації споживання енергії. Наприклад, може знадобитися оцінити вигоди від додавання того чи іншого пристрою і порівняти їх з ціною придбання та встановлення, або ж вибрати з кількох алгоритмів, що розробляються, найкращий.

З точки зору кінцевого користувача до системи можуть пред'являтися такі вимоги, як доступ до історичних даних та тенденцій, можливість віддаленого керування пристроями, попередження про небажані ситуації [18].

Конфіденційність даних, надійність та ефективність системи також відіграють роль в оцінці її якості. Подібні вимоги скоріш характерні для продуктів практичної спрямованості, призначених для прямого використання за призначенням, а чи не дослідницьких прототипів.

2.3 Можливі підходи щодо забезпечення ефективності використання енергії

Перш ніж перейти до аналізу конкретних підходів, зазначимо кілька загальних міркувань.

Поняття ефективності використання енергії залежить від обраних критеріїв оцінки. Можливі критерії було наведено у попередньому розділі. Вибір конкретних вимог залежить від цілей користувача, тому поза таким контекстом прийняти рішення про те, які підходи кращі, буває неможливо, хоча деякі фактори відіграють роль для будь-яких завдань. При виборі варто орієнтуватися на ситуацію. Необхідно проаналізувати, в яких сферах можливе отримання найбільших переваг, а де використання енергії і так близько до мінімального. У деяких випадках достатньо невеликих змін особливо неефективних алгоритмів роботи пристроїв, щоб значно зменшити споживання енергії.

Не варто забувати і про ефективність підконтрольних пристроїв. Нерідко простим способом знизити енергоспоживання буде використання найкращих матеріалів та апаратури. При цьому система домашньої автоматизації обмежена можливостями пристроїв.

Разом з тим вартість впроваджуваного рішення може виявитися вищою за пропоновану ним економію, або ж від користувачів потрібно змиритися зі значним втручанням у їх спосіб життя. У такому разі зниження енергоспоживання навряд чи виявиться пріоритетом.

При аналізі ефективності потрібно як теоретичний розгляд, а й проведення експериментів, оскільки можуть виявитися непередбачені поєднання чинників, які призводять до несподіваним результатам.

Найбільш простим підходом до зниження енергоспоживання є включення та відключення пристроїв залежно від знаходження людей у приміщенні. Такий метод дозволяє забезпечити значне зменшення використання енергії порівняно з ситуацією, коли побутова техніка залишається включеною, навіть коли її роботи немає необхідності [7]. Тим не менш, даний підхід обмежений тим, що дає лише невеликі переваги зручності в тому випадку, коли користувач і так робить щось подібне вручну.

Зазначимо, що використання подібних алгоритмів у контексті ОВіК, особливо у поєднанні зі здатністю системи до самонавчання та передбачення розташування користувачів, призводять до значного зниження енергоспоживання на підтримку комфортної температури [20]. У той же час використання програмованих термостатів вручну пов'язане з частими помилками, що призводять до збільшення енергоспоживання [11], а використання інтелектуальних термостатів без здатності до передбачення поведінки користувачів призводить до втрат в комфорті [20].

Окрім перевірки знаходження людей у приміщенні, існує потенціал для використання сезонних та денних варіацій показників довкілля для управління ОВіК [7]. Таким чином, як щодо доступного та простого рішення у сфері оптимізації енергоспоживання можна запропонувати систему, що ґрунтується на комбінації датчиків для оцінки навколишнього середовища та знаходження користувачів, і керуючу пристроями виходячи з цього.

Таку систему рекомендується включити максимальну кількість пристроїв, які не повинні бути доступними на вимогу користувача, а пов'язані із забезпеченням необхідного рівня комфорту. Насамперед, до них відносяться пристрої ОВіК та освітлення.

Для зменшення залежності від централізованих виробників енергії та пов'язаних із цим грошових витрат пропонується використання поновлюваних

джерел енергії (ВІЕ). Їхня ефективність залежить від географічних умов, так що необхідні дослідження на основі погодних даних обраної території, щоб оцінити перспективи застосування сонячних батарей, вітроелектричних установок та інших пристроїв.

Однією з основних проблем ВДЕ є їхня нестабільність. Для зменшення дії цього фактора система може бути доповнена засобами зберігання енергії [18]. Коли умови довкілля дозволяють виробництво енергії, акумулятори заряджаються, після чого вона може бути використана, коли є потреба.

Часто пропонується підхід, що ґрунтується на плануванні використання гнучкої побутової техніки, з чого випливає вирівнювання попиту, у тому числі зниження пікового навантаження, завдяки чому загальна ефективність збільшується [13]. Проте такий ефект буде помітним лише на рівні великих груп домогосподарств, що перешкоджатиме його впровадженню. Для індивідуальних користувачів метод має сенс у разі, якщо використовується динамічне ціноутворення лише на рівні енергосистеми. Тоді слід призначати використання побутових приладів на періоди, коли вартість енергії мінімальна. Зауважимо, що подібний механізм також може бути корисним у поєднанні з локальними ВІЕ, коли використання деяких пристроїв може бути заплановане на періоди їх високої продуктивності. Таким чином, планування використання побутової техніки може виступати як альтернатива установці пристроїв зберігання енергії.

Крім встановлення спеціалізованих батарей, існує можливість використовувати електро-або гібридні автомобілі як засіб зберігання енергії [9]. Зрозуміло, мобільний характер такого пристрою призводить до того, що він не завжди буде доступним. Крім того, користувачі нерідко очікують, що автомобіль повинен бути заряджений, коли вони бажають його використовувати додатково обмежуючи потенціал такого засобу зберігання.

Всі описані підходи виграють від прогнозування таких факторів, як продуктивність ВІЕ, розподіл попиту за часом і пристроям, що використовуються, ціна на енергію у разі її мінливості.

Перш ніж розпочати планування та проведення експерименту, необхідно сформулювати тестовану гіпотезу чи підхід, і навіть визначити базовий випадок. Наприклад, може порівнюватися система управління енергією, що використовує сонячні панелі, із вбудованими батареями та без.

Від вибору базової конфігурації системи залежатимуть розраховані переваги та недоліки, тому важливо проаналізувати ситуацію, щоб визначити її коректно.

Як наступний крок потрібно виділити особливості функціонування обраного підходу з допомогою аналізу предметної області.

Зокрема, необхідно визначити, які фактори сприяють та перешкоджають його роботі. Як приклад можна навести ВДЕ: для конкретної технології слід визначити, за яких умов довкілля вона здатна виробляти енергію.

На основі виділених факторів необхідно створити імітаційну модель, що генерує потрібні показники.

У тому числі слід визначити конфігурацію експерименту, зокрема, вхідні дані про навколишнє середовище, такі як температура, вологість, хмарність, швидкість вітру та інші залежно від необхідних розрахунку показників.

Також потрібно вибрати тривалість проведення експерименту та інтервал між двома кроками моделювання.

Зрештою, необхідно підключити до моделі прототип системи, провести імітаційний експеримент та проаналізувати його результати.

Висновки. У другому розділі було вирішено такі завдання:

- охарактеризовано завдання забезпечення енергоефективності розумного будинку, у тому числі середовище її роботи та основні компоненти;
- виділено вимоги до таких систем, їх показники якості;
- розроблено можливі підходи щодо забезпечення енергоефективності, а також структура експериментів для виділення найкращих.

3 ВИКОРИСТАННЯ AWS IOT CORE І NODE-RED ДЛЯ ТЕСТУВАННЯ СИСТЕМИ

3.1 Імітаційна модель довкілля

Було запропонований підхід до моделювання та тестування систем інтернету речей, при якому вона відокремлюється від програми, що відповідає за генерацію та постачання даних. У той час як завданням тестованої програми є коректна реакція на дані, що надходять від датчиків, про навколишнє середовище, від тестуючої системи потрібно генерація і передача цих даних, а також збір та аналіз інформації про реакцію на них. Під моделюванням розуміється дослідження поведінки системи непрямим шляхом, тобто з допомогою аналізу допоміжних об'єктів [2].

Хоча повне поділ двох програм неможливе, оскільки система тестування повинна знати про те, які дані і в якій формі надходять на входи і виходи, система, що тестується, може бути підключена таким чином, що з її точки зору надходять згенеровані дані не відрізняються від реальних. Скористаємося тим фактом, що будь-яка система інтернету речей за визначенням повинна з'єднуватися з фізичними пристроями, тобто в ній є входи та виходи, через які передаються повідомлення з використанням певного протоколу. Крім того, при розумінні галузі використання систем, які необхідно протестувати або досліджувати, можливо звузити кількість моделюваних показників.

Розглянемо ключові деталі реалізації програми, що тестує. Важливою частиною є модель навколишнього середовища - набір правил, за якими генеруються показання датчиків з урахуванням як даних про зовнішнє середовище, так і станів пристроїв системи, причому необхідно досягти реалізму в певному сенсі, що зазвичай означає, що результати роботи моделі повинні збігатися з аналогічним сценарієм реальному світі із заданим ступенем точності. Таким чином, завдання побудови імітаційної моделі полягає у знаходженні наближених до реальності правил, якими формуються дані. Вибір правил залежить від предметної

області, доступних обчислювальних ресурсів та необхідної точності моделювання. Точність даних, що генеруються, залежить не тільки від правил, але і від даних, до яких вони застосовуються. Тому необхідно вирішити питання про джерело даних довкілля. Можливим варіантом використання історичних даних. Крім того, можливо відомий очікуваний розподіл даних, тоді можна використовувати методи генерації випадкових величин. У той самий час нерідко буває так, що потрібно перевірити систему у заданих умовах: наприклад, потрібно протестувати реакцію системи на різке зміна умов довкілля, виникнення небажаної ситуації тощо. У такому випадку залишається тільки використовувати необхідні формули для створення даних.

Підхід до моделювання умов довкілля ґрунтується на загальних засадах імітаційного моделювання. Для деяких показників має сенс використовувати методи системної динаміки, що особливо носять безперервний характер, таких як температура. Для дискретних показників є більш відповідним агентне або подієве моделювання. Моделювання на основі використання агентів особливо корисне, коли в системі присутні дійові особи, що характеризуються своєю поведінкою. Крім того, найчастіше виявляється вигідно використовувати гібридний підхід, що поєднує два чи більше «чистих» методів імітаційного моделювання.

Особливим показником у тих імітаційного моделювання систем інтернету речей є внутрішній час. Важливою властивістю систем інтернету речей і даних, з якими вони працюють, є тривалість оброблюваних і керованих процесів. Наприклад, нагрівання кімнати може тривати декілька годин, а значні зміни температури навколишнього середовища спостерігаються в масштабі від дня до року в залежності від необхідної амплітуди. Тому потрібне моделювання внутрішньосистемного часу із значним прискоренням щодо зовнішнього. Під зовнішнім часом розумітимемо час, який займає моделювання з погляду зовнішнього спостерігача. Зокрема, зручним буде вимірювати час одного кроку симуляції: може бути одна секунда, одна десята чи сота секунди тощо. Внутрішній, чи системний, час — це час, що проходить усередині моделі, інакше кажучи, з погляду системи, що тестується. Для моделювання найчастіше необхідно

«прискорити» системний час, тобто якщо за один крок симуляції для зовнішнього спостерігача проходить одна секунда, то для системи може проходити одна хвилина, година і так далі. При цьому прискорення обмежене як можливостями використовуваного середовища, доступними в ньому ресурсами, так і обсягом трудомісткістю обчислень на кожному кроці.

Для того, щоб уникнути обчислень з числами з точкою, що плаває, можливо також вести облік кроків від нуля до заданого загального числа. У такому випадку час у годинах або інших одиницях виражається через кількість кроків за таку одиницю часу, яка є окремим параметром моделі.

Вибір інтервалу внутрішньосистемного часу, який займає один крок, залежить від багатьох факторів. З одного боку, підвищення точності моделювання необхідно у випадку крок зменшити. Іноді буває можливість розрахувати необхідну довжину необхідної точності. На реалістичність моделювання також впливає відповідність часового проміжку частоті прийняття рішень системою: якщо деякі події занадто короткі за часом, то вони можуть бути не зафіксовані при моделюванні.

З іншого боку, існують чинники, регулюючі мінімально можливу довжину кроку, насамперед пов'язані з доступними щодо моделювання ресурсами. Якщо моделювання одного кроку досить трудомістке, досягти досить довгих періодів моделювання загалом можливо лише збільшуючи загальний зовнішній час. Крім того, ступінь, в якому система, що тестується, може збільшити швидкість реагування на зовнішні по відношенню до неї події, обмежена, що встановлює верхню межу «прискорення» внутрішнього часу по відношенню до зовнішнього.

Системи управління енергоспоживанням вимагають даних як про стан та роботу підконтрольних їм пристроїв, так і про навколишнє середовище. Для коректного моделювання таких показників потрібно використовувати правильні способи їх розрахунку, а й задати правильні константні параметри. Такі параметри можуть бути отримані шляхом вивчення статистичних даних про навколишнє середовище у заданій місцевості. У той же час, нерідко потрібно вивчити реакцію

системи на особливу ситуацію. У такому разі параметри можуть бути вхідними даними тесту чи експерименту.

В контексті забезпечення ефективного використання енергії для обігріву приміщення найбільш важливим показником зовнішнього середовища є її температура. Зауважимо, що він є внутрішнім по відношенню до системи, що тестує (імітаційної моделі), на роботу тестованої системи він впливає опосередковано, зокрема, через розрахунок внутрішньої температури контрольованого приміщення, вплив на значення якої він надає, збільшуючи або зменшуючи втрати тепла.

Враховуючи, що проміжок внутрішньосистемного часу, протягом якого проводитиметься експеримент, може значно відрізнятися залежно від завдання, моделювання температури навколишнього середовища набуває різних форм. У разі коротких проміжків часу, які застосовуються, зокрема, для перевірки нормальної роботи системи, оцінки якості роботи окремих її компонентів і так далі, простим способом моделювання є використання середньої температури, складеної з добовим коливанням.

Для розрахунку витрат на опалення будинку необхідно визначити деякі параметри, зокрема, його геометрію та використані матеріали. Під геометрією розуміється форма, вимірювання стін та даху, кількість та розміри вікон тощо параметри житла. Зокрема, задаються ширина та довжина удома, а також його висота без даху. Для даху задається кут її нахилу. Для вікон задається ширина та висота. Крім того, додатковим параметром вікон та стін є їх товщина. Використані матеріали задаються через їх теплові властивості, звідки можна розрахувати тепловий опір будинку, знаючи товщину стін та вікон.

Для розрахунку теплових втрат необхідно знати площу поверхні, що стикається із зовнішнім середовищем. У будинку є два типи поверхонь: поверхня стіни та поверхня вікна.

Площа зовнішньої поверхні будинку, що залишилася, є площею стін. Як спрощення вважатимемо, що дахи будинку має ту ж товщину і складається з того самого матеріалу, що і його стіни.

Швидкість нагрівання повітря залежить від його маси, тому для кінцевого розрахунку внутрішньої температури при використанні нагрівача потрібно буде розрахувати масу повітря в будинку.

З іншого боку, необхідно задати характеристики використовуваного обігрівача, щоб визначити кількість теплової енергії, що подається. Співвідношення між подачею тепла від обігрівача та втратою тепла через довкілля визначає напрямок та швидкість зміни температури всередині приміщення. Простою моделлю буде пристрій, що визначається температурою гарячого повітря, що подається, і швидкістю його подачі.

У контексті управління споживанням енергії, особливо, коли йдеться про пристрої ОВіК, має сенс виділити три можливі стани, пов'язані з поведінкою людей: «будинок порожній» відноситься до випадку, коли всередині немає жодної людини; «жителі активні», коли хоча б одна людина перебуває в будинку і не спить; «жителі сплять», коли немає жодної пильної людини в будинку.

Для визначення наявності хоча б однієї активної людини у підконтрольному приміщенні можна використовувати датчики руху чи присутності [20]. Поширеним механізмом їхньої роботи є аналіз інфрачервоного випромінювання. Зокрема, до цього типу належать пасивні інфрачервоні датчики. Зауважимо, що для використання в системах домашньої автоматизації варто вибирати більш дешеві, прості в установці, найменш нав'язливі пристрої, навіть якщо це означає певні втрати в точності вигляді хибнопозитивних чи хибнонегативних результатів, адже ця сфера відрізняється відносно невисоким ступенем відповідальності.

Для моделювання поведінки такого датчика потрібна розробка правила, яким визначається активність користувачів. Як найпростіша модель можна запропонувати відправлення повідомлень в рамках заданого проміжку часу. У межах цього проміжку емуляція датчика полягає в тому, щоб надсилати сигнал на кожен крок моделювання, інакше датчик мовчить. Більш складна модель може включати ймовірнісні відхилення від такого заданого розкладу, проте це вимагає додаткових досліджень поведінки людей.

Для відстеження роботи системи необхідно збирати статистику про її поведінку: наприклад, як часто і в які моменти вона включала та відключала опалення. З цих даних можна розрахувати такі показники, як споживання електроенергії. Крім інформації про поведінку системи також доступні згенеровані дані та параметри системи, такі як час або температура в приміщенні. Наприклад, комфорт користувачів може визначатися як здатність системи підтримувати температуру в межах заданих значень.

Для цього використовується показання датчиків, до яких система також має доступ.

Враховуючи, що з точки зору користувача метою системи управління енергоспоживанням є мінімізація витрат на енергію, як основна оцінка якості роботи таких систем запропонуємо загальну вартість використаної енергії.

Для спрощення розрахунків вважатимемо, що відсутні втрати при перетворення одного виду енергії в іншу. Зокрема, для обчислення вартості спожитої енергії використовуємо константу — вартість однієї кіловат-години.

Node-RED пристрої, що споживають енергію, були підключені до вузла розрахунку сумарної вартості.

Менш очевидною оцінка комфорту користувачів системи. У цій роботі вимоги щодо підтримки того чи іншого температурного режиму тощо запити були інкорпоровані в систему як такі, що обмежують умови.

Тим не менш, за вихідними даними проведеного експерименту можна розрахувати складніші оцінки, якщо виникає така необхідність.

Зокрема, найчастіше після використання вимог як обмежень у системі пропонується оцінка, заснована на лінійному відхиленні від цільових. значень [14]. В якості альтернативи можна запропонувати розрахунок частки часу, проведеної в стані, відмінному від бажаного.

Така оцінка є трохи простішою для розрахунку, оскільки на кожному тимчасовому кроці вона має двійкове значення.

3.2 Оцінка роботи тестованої системи

Для проведення експериментів над системою, що тестується, необхідно підключити систему таким чином, щоб вона отримувала необхідні їй згенеровані дані. Відповідно, набір показників, що моделюються, частково визначається областю застосування тестованої системи, частково — реалізацією конкретної системи або прототипу. Таким же чином можна міркувати щодо обчислення оцінок якості роботи системи, з тією відмінністю, що на них впливають вихідні дані.

Доповнення до питання підключення і передачі даних між середовищем моделювання і системою, що тестується, слід торкнутися розробки користувальницького інтерфейсу, через який буде проводитися управління процесом імітаційного дослідження. Зокрема, такий інтерфейс має дві основні компоненти: елементи налаштування та управління експериментом, а також уявлення вихідних даних процесу симуляції, включаючи можливості експортувати результати для аналізу.

Використання Node-RED у поєднанні з AWS IoT Core дозволяє вирішити всі вищезазначені питання, від моделювання даних до видачі результатів. Середовище імітаційного моделювання, що включає генерацію датчиків, підключення системи, візуалізацію інформації і інтерфейс користувача, була реалізована за допомогою Node-RED. Для збору та передачі даних було підключено AWS IoT Core. Далі опишемо ключові деталі проектування та реалізації.

Однією з важливих переваг Node-RED є те, що за допомогою цього інструменту можливо створити простий прототип системи в тому ж середовищі, в якому ведеться або вестиметься основна розробка. Засоби візуального програмування Node-RED у поєднанні з використанням хмарних технологій,

Зокрема, AWS IoT Core дозволяють створювати прототипи систем інтернету речей без використання реальних пристроїв, що дозволяє пропрацювати архітектуру системи до її фізичної реалізації. Крім того, Node-RED також надає засоби для простої візуалізації системи, що вийшла, що дозволяє створити

прототип графічного інтерфейсу у межах цього середовища. Зауважимо, що ці властивості роблять особливо зручним тестування такої системи в рамках імітаційних експериментів.

Для виконання середовища Node-RED було створено віртуальну машину у хмарному сервісі AWS Compute Cloud. Цей сервіс є частиною AWS Cloud і надає обчислювальні потужності, що масштабуються, для створення віртуальних машин і управління ними [4]. Даний сервіс пропонує широкий вибір налаштувань віртуальних машин, від різних операційних систем до тонкого настроювання ресурсів, що використовуються. Також був створений реєстр системи в AWS IoT Core, що включає кілька пристроїв. З'єднання з цими об'єктами було встановлено за допомогою вузлів MQTT Input та MQTT Output.

Вихідні дані для генерації можуть бути поміщені в кожен окремий вузол, однак у цій роботі для цього було виділено окремий вузол для конфігурації. Незмінні дані задаються при розгортанні потоку, при цьому є можливість встановити частину даних у початковий для експерименту стан через графічний інтерфейс.

Перш ніж перейти до опису процесу для датчиків, торкнемося реалізації моделювання часу за допомогою Node-RED, хоча час у представленому в даній роботі вигляді і цікавить насамперед при проведенні імітаційних експериментів. Інтервал зовнішнього часу підтримується вузлом Inject. Оскільки потік весь час активний, а хотілося б вміти запускати експеримент і переглядати його результати, було додано вузол, що вважає кроки симуляції в межах від нуля до заданої загальної кількості. Після досягнення останнього кроку вузол видає сигнал про закінчення експерименту окремий вихід. Для зручності демонстрації також використовувався вузол, відповідальний отримання внутрішньосистемного часу в годинах. Реалізація вузлів, які відповідають за керування віртуальним часом, показані у додатку А.

Що ж до емуляції роботи датчиків, реалізація цього процесу залежить від характеру його показань. Зокрема, їх можна класифікувати на дискретні та безперервні. До дискретних віднесемо ті датчики, які передають дані, що належать

до кінцевої множини можливих повідомлень. Якщо ж показання датчиків вибираються з деякого числового проміжку значень, такий датчик назвемо безперервним.

Зазвичай дискретні датчики виникають у тому випадку, коли потрібно реагувати на певну подію. Найбільш простим варіантом є датчик, множина значень якого складається з одного елемента, що означає, що подія виникла. Прикладом такого датчика може бути датчик присутності. У Node-RED такий датчик моделюється шляхом перевірки умови, причому вузол видає повідомлення, якщо умова виявляється істинною. Аналогічно можна діяти для датчиків, що мають більше одного значення, тільки необхідно перевірити більше умов.

Безперервні датчики, навпаки, рідше реагують на події, а часто повинні постійно передавати значення, що вимірюються ним, через деякий заданий проміжок часу. Прикладом такого датчика може бути датчик температури. У Node-RED розрахунок температури виконується у кількох вузлах, оскільки її значення впливають багато чинників. Деякі з цих вузлів відносяться до обробки даних, отриманих від системи, що тестується. Тим не менш, окремий вузол був виділений для передачі даних про температуру на кожному кроці симуляції, адже передбачається, що система повинна регулярно отримувати нові дані про температуру всередині підконтрольного приміщення. Набір вузлів, призначених для генерації показань датчиків та керування часом, показаний на рисунку 3.1. Їхній внутрішній пристрій знаходиться в додатку А.

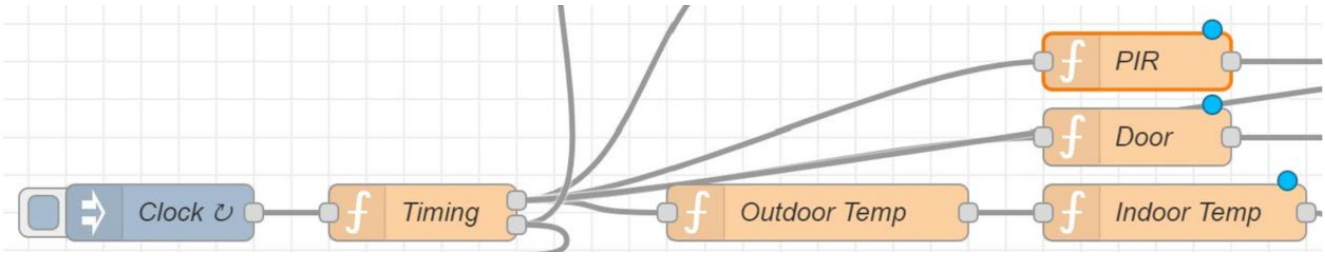

Рисунок 3.1 - Основні вузли генерації показань датчиків

Модель навколишнього світу може бути будь-якої миті доповнена, якщо виникла потреба у моделюванні нового показника. В систему додається новий

вузол, який з'єднується з вузлом, що генерує час. Найбільш відповідним вузлом, особливо у разі складних правил отримання даних, є Function. Усередині цього вузла записується алгоритм. При цьому можуть бути додані інші вузли для розрахунку, якщо потрібно облік будь-яких нових факторів.

На відміну від випадку, коли тестується готовий програмний продукт, при тестуванні теоретично розроблених підходів потрібно спочатку створити прототип системи, яка використовує обраний підхід. Оскільки реалізація сильно пов'язана з конкретним дослідженням, то торкнемося лише загальних міркувань. При використанні у сфері інтернету речей Node-RED пропонує кілька переваг, порівняно з іншими варіантами. По-перше, Node-RED є інструментом для візуального програмування, що дозволяє не тільки зробити просту автоматизацію навіть тим, хто не знайомий з розробкою, але полегшити створення прототипів, інтегрувати усі стадії розробки системи. При цьому Node-RED досить гнучкий, щоб вирішувати більшість речей завдань, що постають в області інтернету. По-друге, Node-RED можна запускати на великій кількості платформ. Сюди відносяться як Raspberry Pi, так і хмарні сервери. Таким чином, розроблений прототип можна використовувати як основу подальшої розробки продукту. По-третє, розроблені сценарії та правила не залежать від фізичної реалізації системи, оскільки Node-RED надає можливості отримання даних та управління абстрактними пристроями, які можуть бути пов'язані з фізичними окремо.

Що стосується підключення системи або прототипу до середовища моделювання, потрібна інтеграція з вузлами або потоками як генерації даних, так і аналізу оцінки якості, що приблизно відповідає входам та виходам такої системи. Спочатку розберемо загальний принцип роботи із датчиками.

Основним датчиком для роботи з пристроями ОВіК у контексті керування енергоспоживанням є датчик температури усередині контрольованого приміщення. Типовим використанням такого датчика є включення та відключення обігрівача залежно від поточної температури у приміщенні та вимог за допустимими її межами. Можливі вузли для обробки температурних показань показано рисунку 3.2. Їх внутрішній пристрій показано у додатку Б.

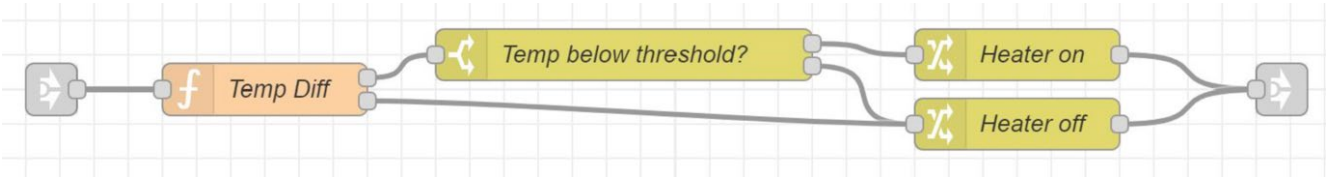

Рисунок 3.2 — Вузли для обробки показань датчика температури

Для уточнення необхідності використання енергії має сенс використання датчиків присутності. Наприклад, система може підтримувати кілька станів як кінцевого автомата, переходи між якими здійснюються за повідомленнями від датчиків. Управління пристроями в такому разі залежатиме від встановленого стану. Вузли системи, призначені для реагування повідомлення від таких датчиків, зображені рисунку 3.3. Їх внутрішній пристрій показано у додатку Б.

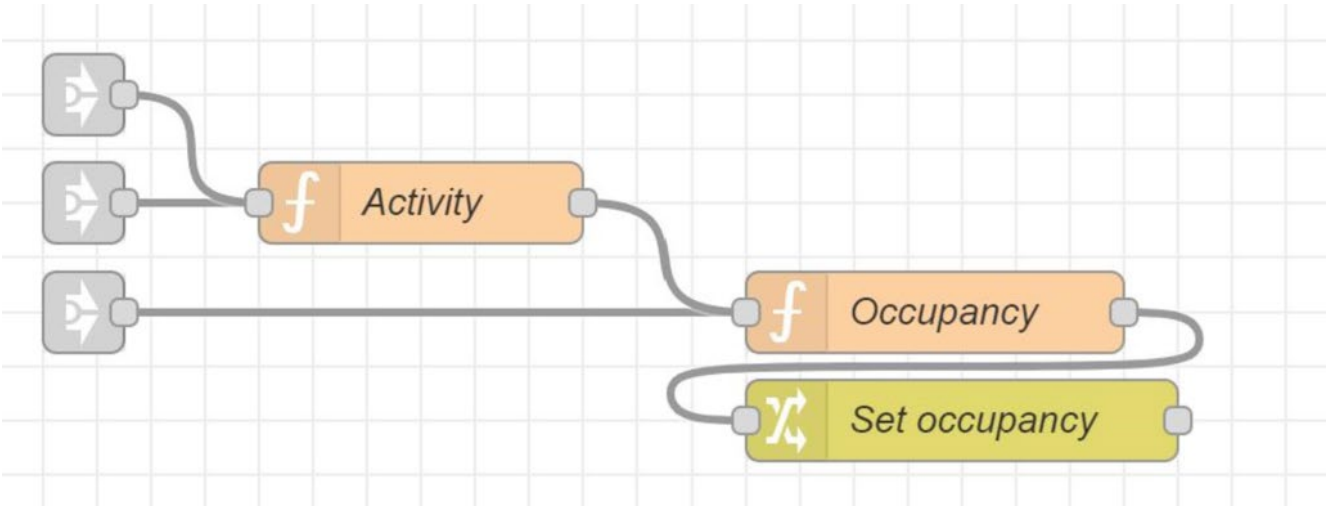

Рисунок 3.3 — Вузли обробки даних від датчиків руху

Важливим завданням комунікативної частини системи тестування є також отримання даних від системи, що тестується. Існують два напрями їх використання: по-перше, це аналіз якості роботи системи; по-друге, це генерація показників датчиків. Реалізація вузлів для обробки даних показана у додатку, а візуальне подання — на рисунку 3.4.

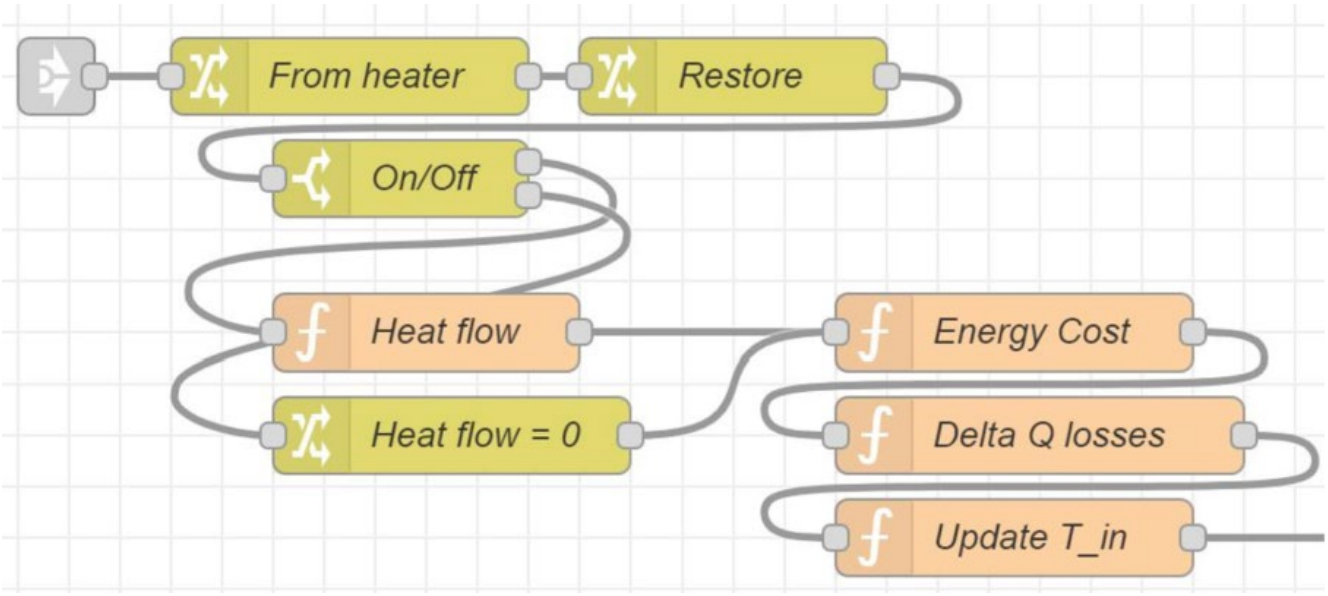

Рисунок 3.4 - Вузли розрахунку внутрішньої температури

Для того, щоб система була наочнішою або інтуїтивніше зрозумілою, бажана візуалізація процесу її роботи. Зокрема, можливо демонструвати генеровані показання датчиків або умови навколишнього середовища середовища, такі як температура навколишнього середовища, температура в приміщенні, або результати роботи системи, наприклад, спожита енергія або її вартість.

Щоб візуалізувати дані, Node-RED пропонує спеціальні графічні вузли, що дозволяють створити елементи інтерфейсу, такі як текстові поля, графіки, так званої «панелі управління».

Наприклад, на графіці можна показати зміну температури або накопичення вартості енергії, а за допомогою тексту - вартість енергії або кількість кроків моделювання.

Для збору даних використовувалися MQTT-вузли з підключенням до AWS IoT Core. Вузли, що відповідають за збирання та передачу даних, а також за їх відображення на графіку, показані на рисунку 3.5. Їх внутрішній пристрій показано у додатку Г.

Крім візуалізації, зібрані дані також зберігаються у файл, який також доступний через інтерфейс користувача. Цей файл створюється під час запуску експерименту та доповнюється при надходженні нових даних.

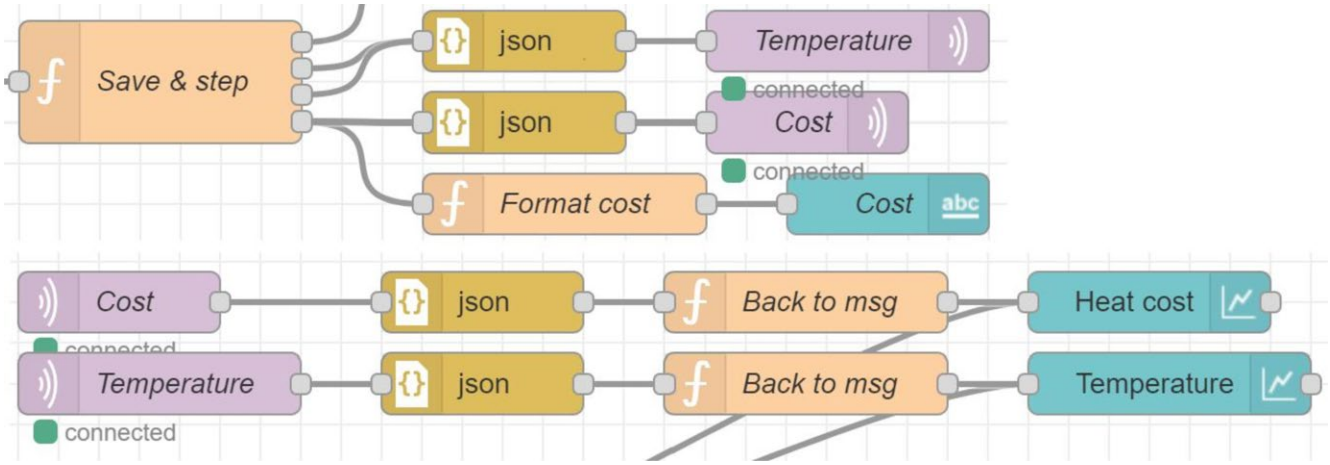

Рисунок 3.5 — Вузли для збирання, передачі, візуалізації даних

Тут же можна вивантажити параметри системи, щоб врахувати їх при подальшому аналізі. Вузли для створення, оновлення та завантаження файлів показані на рисунку 3.6.

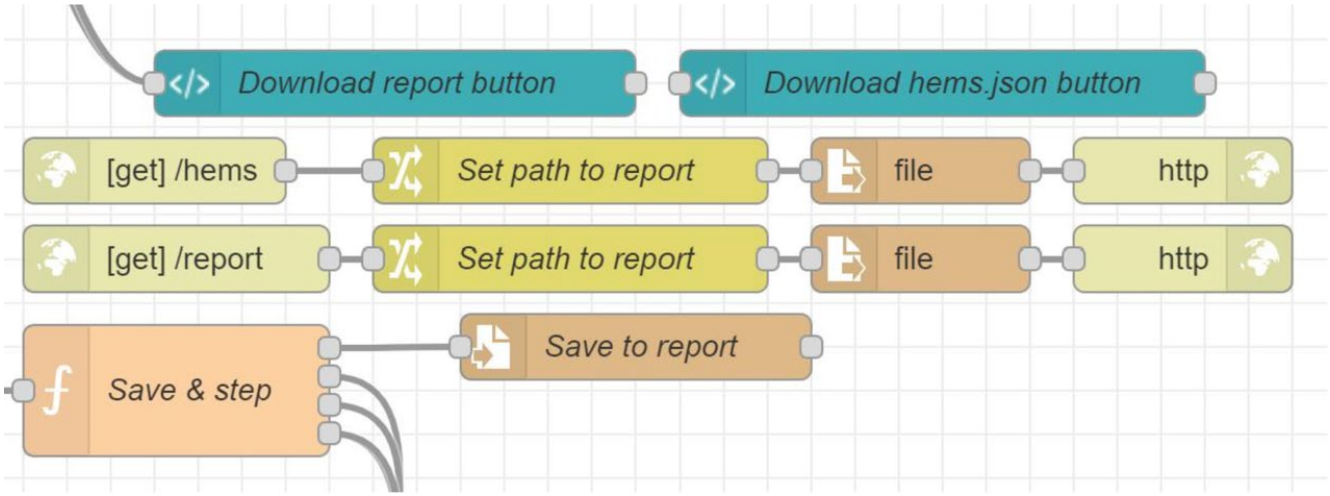

Рисунок 3.6 — Вузли для роботи з файлами

На цю вкладку панелі управління була поміщена кнопка для запуску моделювання. Таким чином, при запуску експерименту користувач може бачити динамічну зміну температури, споживання енергії, кількість пройдених кроків, а після закінчення - вивантажити згенеровані дані, які можна проаналізувати окремо. На верхньому графіці показано накопичення загальної вартості витраченої енергії в процесі симуляції, а на нижньому - температура всередині та поза домом. Зазначимо, що дані для цих графіків відправлялися кожну годину віртуального

часу або кожні десять кроків. Це було зроблено для того, щоб не навантажувати графічну частину, яка потребує великої кількості ресурсів. Вкладка з елементами графічного інтерфейсу, описаними вище, показано рисунку 3.7.

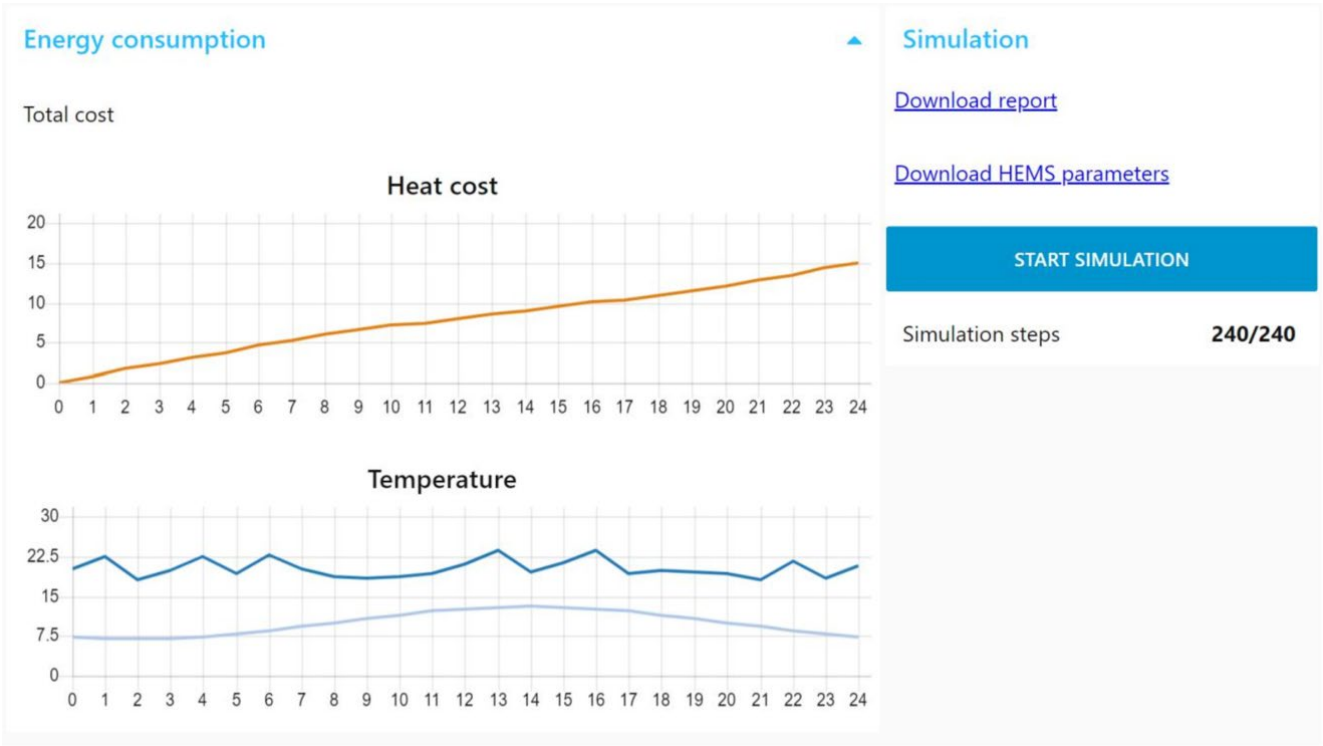

Рисунок 3.7 - Вкладка "Симуляція"

Користувачеві також доступні деякі можливості конфігурації системи. Користувач може налаштувати дві групи параметрів: параметри моделювання та параметри системи. До першої групи відносяться такі параметри, як загальна кількість кроків, а також їх частота, виражена як кількість кроків за одну годину за внутрішньосистемним часом. Що стосується другої групи, то вона, у свою чергу, може бути розділена на параметри системи, що тестується, такі як необхідна температура, і настройки модулів, що включаються. Зокрема, користувач може відключати ті чи інші функції системи, щоб перевірити реакцію системи, порівняти різні стратегії керування енергоспоживанням.

Під частину інтерфейсу користувача, призначену для конфігурації, була виділена окрема вкладка панелі управління, показана на рисунку 3.8, недоступна в той час, коли йде процес моделювання. Це було зроблено для того, щоб користувач

не міг випадково змінити параметри в ході експерименту і таким чином порушити його виконання.

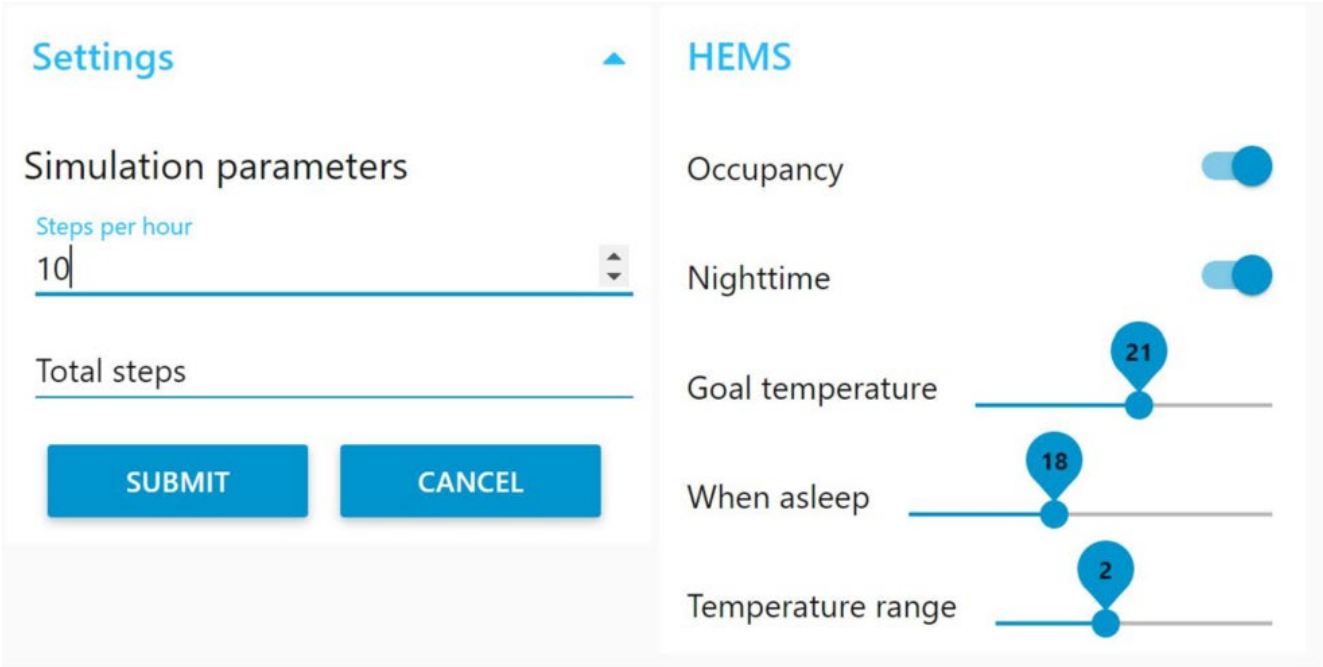

Рисунок 3.8 — Вкладка «Налаштування»

3.3 Проведення експериментів

Для демонстрації можливості проведення імітаційних експериментів, а також аналізу та оцінки використання для цього Node-RED та AWS IoT Core було поставлено наступне підзавдання: порівняти кілька алгоритмів управління пристроями ОВіК, що відрізняються ступенем та способом використання поведінки користувачів системи.

Базовий варіант системи управління енергоспоживанням була розроблена система, що включає і вимикає опалення, ґрунтуючись на температурі всередині підконтрольного їй приміщення. Система відправляє команду на включення, коли різниця між цільової та реальної температури досягає заданого порога, і на вимикання, коли температура знову знаходиться в потрібних межах.

Температурний режим, що виходить, протягом однієї доби показаний на рисунку 3.9, де синя лінія показує температуру в приміщенні, помаранчева — температуру зовні, а червоним відзначені бажані межі.

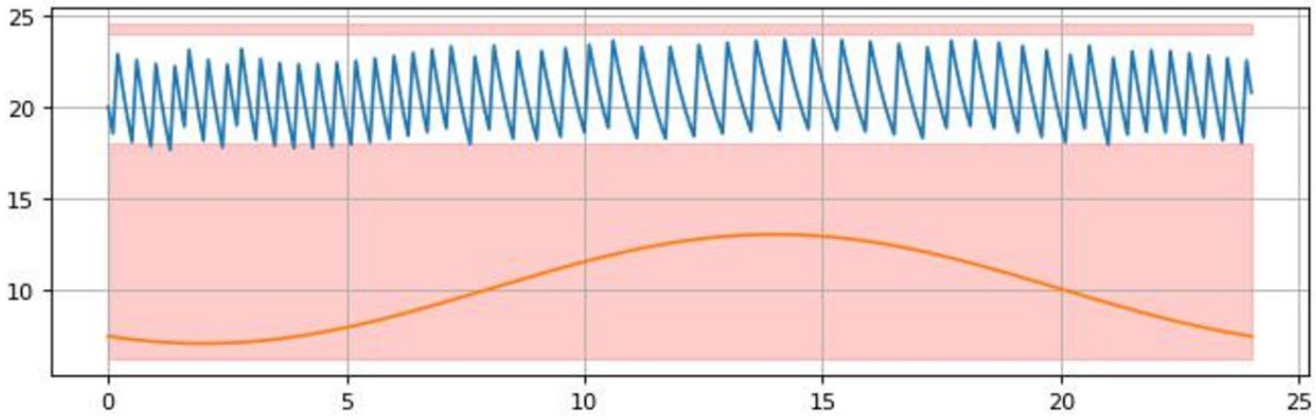

Рисунок 3.9 – Температурний режим базової системи

Для порівняння було обрано дві можливі стратегії зменшення енергоспоживання, що ґрунтуються на відстеженні поведінки користувачів. Перша з них полягає в тому, щоб спостерігати за активністю людей, які перебувають будинку з використанням датчиків руху. Використання такої стратегії виправдане, оскільки вплив холоду не впливає на якість сну, тоді як надмірно висока температура його знижує [10]. Друга використовує датчик на вхідних дверях, щоб відстежувати моменти, коли люди покидають приміщення чи входять до нього. У такому випадку можна відключати пристрої ОВіК, коли вони не потрібні.

Стратегія, заснована на активності, працює в такий спосіб. До системи підключається датчик руху. Поки він посилає сигнали, система перебуває в «активному» стані та підтримує один заданий температурний рівень. Після кожного сигналу система чекає певний час, який скидає після приходу кожного нового сигналу. Однак якщо за цей час не з'явилося нових повідомлень, система переходить у «неактивний» стан, в якому система підтримує іншу, меншу температуру, як показано на рисунку 3.10.

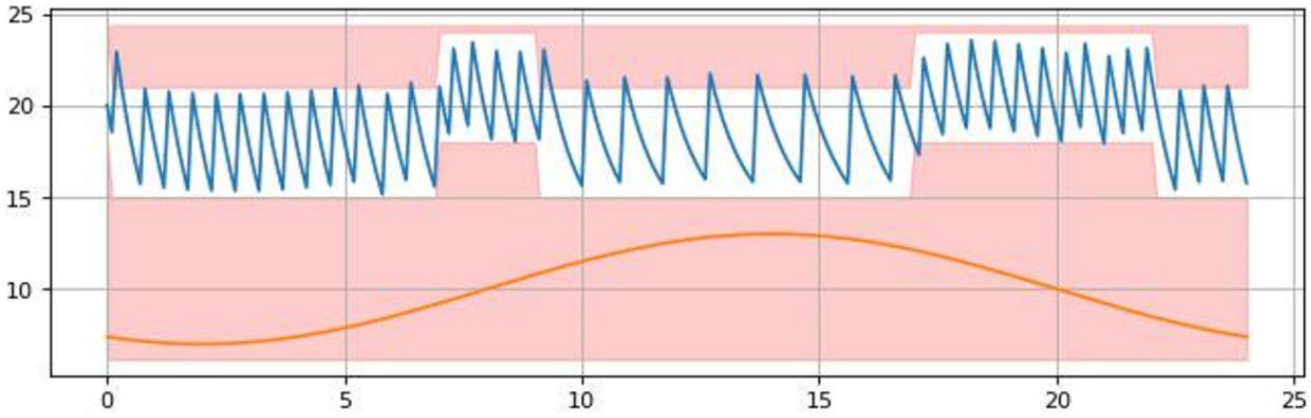

Рисунок 3.10 — Температурний режим з урахуванням активності користувачів

Інша запропонована стратегія дозволяє відстежувати стан будинку порожній», але не активність користувачів. Для цього до системи підключається датчик, що реагує використання вхідних дверей. Таким чином система відключатиме опалення в ті моменти, коли в приміщенні нікого немає. Можливий температурний режим зображено рисунку 3.11.

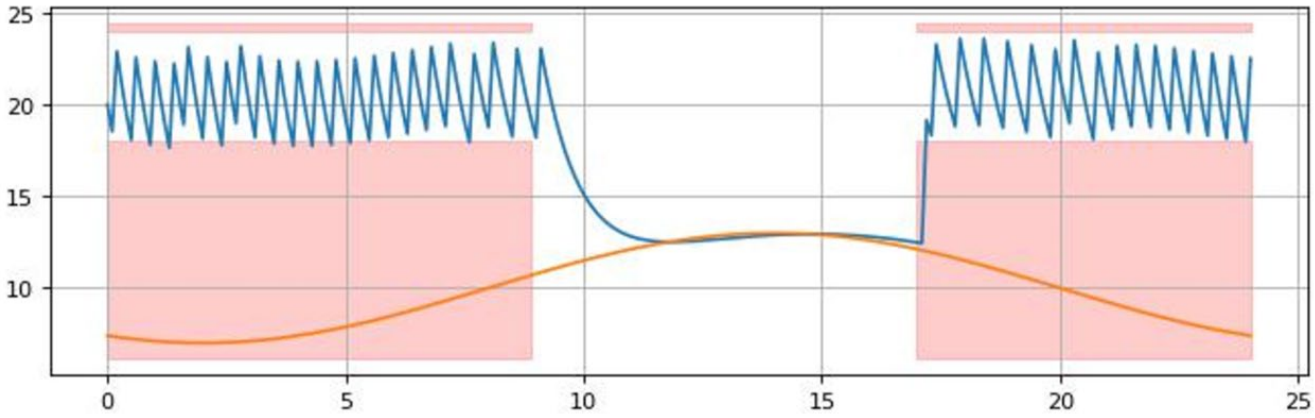

Рисунок 3.11 - Температурний режим з урахуванням наявності людей у приміщенні

Як можна помітити, запропоновані методи не суперечать один одному, тому є сенс також перевірити їхню спільну роботу. У такому разі система визначає стан «будинок порожній» аналогічно попередньому сценарієві, а в протилежному випадку діє залежно від показань датчиків руху. Відповідний температурний режим протягом доби показано рисунку 3.12.

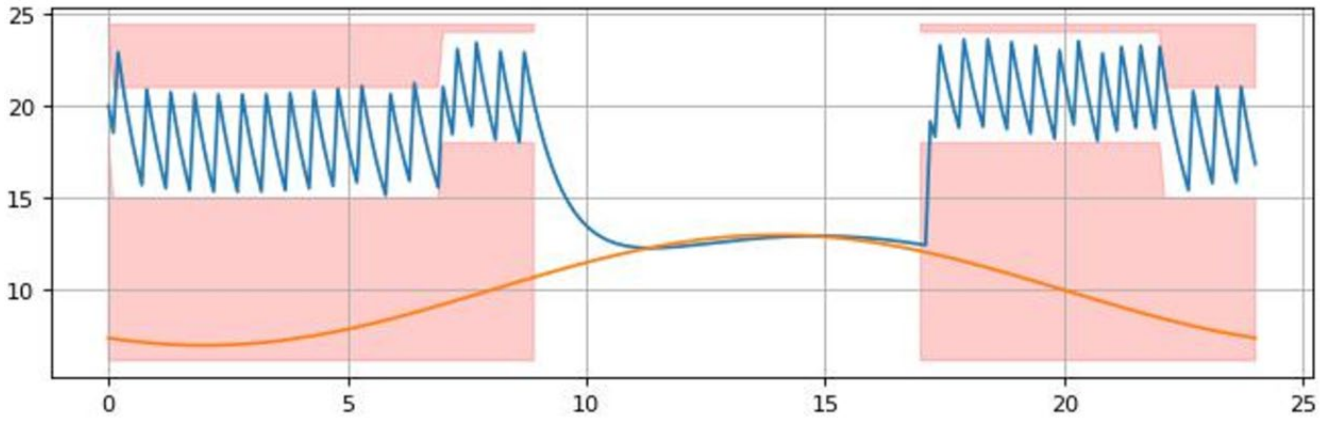

Рисунок 3.12 – Температурний режим при використанні обох стратегій

Основною метрикою функціональності систем керування енергоспоживанням є загальна вартість енергії. Таким чином, щоб оцінити якість та перспективність виділених підходів, потрібно порівняти системи за підсумковою кількістю спожитої енергії. Для цього поставимо такий експеримент: система працює протягом 24 годин за віртуальним часом.

При цьому для моделювання було обрано розмір кроку 6 хвилин, тобто 10 кроків на годину. Формули, що включають час, використовують саме внутрішньосистемний час, не кількість кроків, якщо не зазначено інакше.

Поведінка користувача моделюється наступним чином: вважається, що користувач спить з 22:00 до 7:00, активний з 7:00 до 9:00 та з 17:00 до 22:00, решта часу знаходиться поза контрольованим приміщенням.

Метою користувача є мінімізація енергоспоживання пристроїв ОВіК. Разом з тим, користувач має вимоги щодо комфорту, показані в таблиці 3.1, пов'язані бажаною температурою. Зазначимо, що для деяких підходів, зокрема, пов'язаних з розпізнаванням активності жителів, точнішою була б метрика з вужчими рамками, однак такий показник не був би універсальним і не дозволив би порівняти всі системи між собою.

Гіпотеза полягає в тому, що базова система виявиться найменш ефективною.

Таблиця 3.1 - Вимоги щодо комфорту

Стан	Мінімальна температура	Максимальна температура
Будинок порожній	-	-
Мешканці активні	18	24
Мешканці сплять	15	24

З цієї точки зору, найефективнішою виявиться система, що використовує обидві стратегії. В результаті проведеного імітаційного експерименту було отримано такі результати. Загальна вартість витраченої енергії склала 14,87 умовних одиниць для базової системи, для стратегії з урахуванням активності – 12,64, для стратегії з урахуванням наявності людей у приміщенні – 11,52, для об'єднаної стратегії – 9,88.

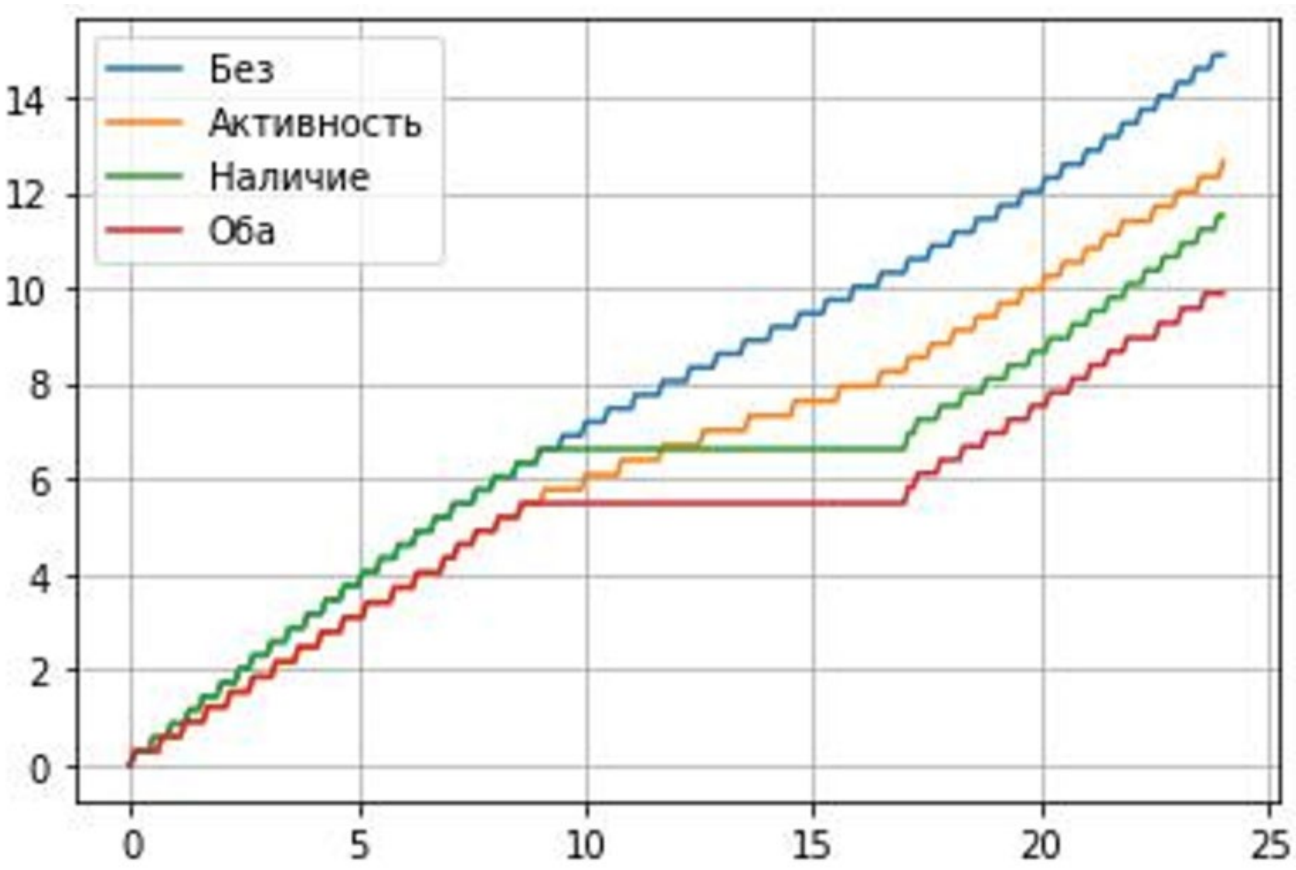

Рисунок 3.13 - Накопичення споживання енергії

На графіку, зображеному рисунку 3.13, можна також бачити динамічне накопичення загальної вартості. Зокрема, зауважимо, що стратегії, що включають

відстеження наявності людей у приміщенні, виграють за рахунок тривалих проміжків без споживання енергії. Отже, висунута гіпотеза підтвердилася.

Отримані результати наведено в таблиці 3.2. Можна бачити, що найкращі результати показав базовий варіант, проте третя та четверта системи трохи від нього відрізняються. Зазначимо, що значних відхилень від необхідних значень немає, що підтверджує можливість підтримки достатнього рівня комфорту всіма системами. Також варто зауважити, що найчастіше порушення спостерігаються при переході між станами, як можна бачити на рисунках вище. Це з тим, що система не встигає досить швидко нагрітися чи охолонути. Можливим рішенням буде уточнення умов керування обігрівачем, щоб система змінювала температуру плавніше.

Таблиця 3.2 - Оцінка комфорту користувача для кожної системи

Система	Частка часу по за даними меж	Середнє відхилення від заданих меж
1	0,0415	0,0072
2	0,1112	0,0472
3	0,0456	0,009
4	0,0539	0,0196

Висновок попереднього аналізу експерименту відзначимо, що до всіх систем пред'являлися однакові вимоги щодо комфорту, щоб їх можна було порівняти, через що в обчисленнях використовувалися досить м'які межі.

Таблиця 3.3 - Результати тривалого експерименту

Система	Споживання ерегії		Комфорт	
	Загальне	Середнє	По часу	Відхилення
1	104,13	14,87	0,0464	0,009
2	87,62	12,52	0,1201	0,0573
3	78,89	11,27	0,05	0,0136
4	70,99	10,14	0,0577	0,0197

У зв'язку з цим зазначимо, що перша та третя системи підтримують досить високу температуру у нічний час, що може бути небажаним для деяких користувачів.

Для перевірки отриманих результатів також було проведено більш тривалий експеримент тривалості сім внутрішньосистемних днів. Показання датчиків моделювалися аналогічним чином. Отримані дані подано у таблиці 3.3.

Показники змінилися незначно, причому якщо абсолютні значення і змінилися, то відносини між моделями, що розглядаються, — ні, з чого можна зробити висновок про справедливість першого експерименту, а також підтвердити його результати.

3.4 Імітаційні експерименти

Аналіз розроблених систем проведемо у два етапи. По-перше, розглянемо результати експериментів з погляду узагальнених показників якості, призначених для оцінки програмних продуктів у будь-яких сферах. Наприклад, тут можна використовувати показники якості зі стандарту ISO/IEC TR 9126-2:2003: функціональність, надійність, зручність використання, ефективність, можливість супроводу та переносимість. Особливу увагу звернемо на функціональність, надійність та ефективність. Зрозуміло, однією з важливих відмінностей програмного продукту з його прототипу є конкретизація вимог щодо нього, тому під час тестування готових систем аналіз відрізнятиметься. Тим не менш, розглянемо чотири розроблені прототипи на виконання перерахованих категорій метрик якості.

Основне місце займає перевірка програмного забезпечення на відповідність вимогам функціональності, проте при цьому критерії часто потрібно розробляти відповідно до конкретної предметної області та очікуваного застосування системи, що розробляється. У розглянутому прикладі вимоги щодо функціональності

полягали в тому, щоб, по-перше, знизити енергоспоживання, а по-друге, контролювати знаходження певних показників у заданих межах. Докладніше висновки з цієї точки зору будуть описані нижче, проте можна відзначити, що всі прототипи виконують певні вимоги функціональності.

У категоріях ефективності та надійності можна судити за появою деяких небажаних подій. Прикладом може бути постійне знаходження обігрівача у включеному стані, внаслідок чого температура приміщенні вийде межі бажаних користувачем значень. Крім того, така система також була б неефективною у використанні енергії. З іншого боку, прикладом відмови системи можна назвати стан, коли система не посилає жодних команд, що також призвело б до виходу за межі необхідного режиму. У жодному з експериментів немає значного виходу межі необхідних значень, тобто небажаних подій був, що свідчить користь аналізованих прототипів.

Інші категорії метрик досить важко оцінити програмним чином, тобто без участі людини, проте можна провести аналіз зручності використання. Можна бачити, що система практично не вимагає участі людини для своєї роботи, за винятком встановлення необхідної температури. Тим не менш, існує ризик неправильного використання такої системи, наприклад, спроби ручного налаштування, що призведе до зниження її ефективності.

Із аналізу загальних категорій метрик якості програмного забезпечення робимо висновок, що всі системи, що розглядаються, задовольняють базовим вимогам, однак, слід провести більш глибокий аналіз, щоб сформулювати рекомендації щодо вибору між стратегіями.

Безпосередньо з проведених експериментів і розрахованих метрик споживання енергії і комфорту можна дійти невтішного висновку у тому, що найкращою є четверта система, використовує як чинники прийняття рішень активність і присутність людей у приміщенні. У таблиці 3.4 показано, наскільки підвищилося чи знизилося вживання енергії між кожною парою систем, причому частка вважається щодо енергоспоживання системи, зазначеної у стовпці. Показники були розраховані за даними тижневого експерименту. Бачимо, що,

порівняно з найменш ефективною системою, споживання енергії вдалося знизити на 32%. У той же час використання наступної ефективності стратегії оптимізації енергоспоживання підвищує його на 11%. Втрати в комфорті незначні і, швидше за все, залишаться непоміченими для кінцевих користувачів, причому зниження температури під час сну може навіть підвищувати комфорт, хоча цей фактор у обчисленнях не враховувався. Таким чином, з точки зору функціональності слід порекомендувати систему, яка використовує дані про активність та присутність людей у приміщенні.

Таблиця 3.4 - Відносне використання енергії

Система	1	2	3	4
1	100%	119%	132%	147%
2	84%	100%	111%	124%
3	76%	90%	100%	111%
4	68%	81%	90%	100%

Тієї ж точки зору не рекомендується використовувати другу систему, яка враховує лише активність користувачів. Хоча вона дозволяє знизити споживання енергії щодо найгіршого варіанту на 16%, її використання пов'язане із відносно великими втратами в комфорті. З практичної точки зору також варто відзначити, що використання датчиків руху може призводити до помилково-негативних результатів, знижуючи зручність використання системи.

Незважаючи на економію, що спостерігається, варто відзначити, що четверта система вимагає відносно більшого втручання в особисте життя користувачів, а також витрат на установку. З цієї точки зору варто відзначити, що навіть базова система потенційно зменшить витрати енергії в порівнянні з випадком, коли опалення включено весь холодний сезон року, і при цьому підвищить комфорт мешканців.

Зрештою, зазначимо, що в деяких випадках встановлення подібної системи домашньої автоматизації хоч і призведе до зниження енергоспоживання, проте меншою мірою, ніж краща теплоізоляція, ефективніші прилади та інші інженерно-

технічні рішення. З іншого боку, енергоспоживання також можна знизити, якщо користувач готовий терпіти деяке зниження комфорту, наприклад, встановлення нижчої цільової температури.

Було продемонстровано, яким чином можна використовувати Node-RED та AWS IoT Core в імітаційному моделюванні з метою тестування та дослідження систем інтернету речей, зокрема – систем управління енергоспоживанням. Проведення експериментів, підтвердило гіпотезу про можливість використання таких інструментів. Тим не менш, потрібний більш глибокий аналіз процесу розробки застосування середовища імітаційного моделювання, щоб зробити висновки про переваги та недоліки такого підходу порівняно з іншими.

Під час теоретичного розгляду питання про тестування систем інтернету речей шляхом імітаційного моделювання було проаналізовано існуюче рішення у цій галузі — так званий Simulink, що є частиною пакету прикладних програм MATLAB. Цей продукт призначений в першу чергу для досліджень різних фізичних систем, у тому числі, може бути застосований і для моделювання житла разом із системою управління енергоспоживанням. Середовище Simulink дає змогу досягти високої точності за рахунок використання складних чисельних алгоритмів розрахунку. Разом про те поріг входження щодо високий з погляду як навичок, і коштів. Тому альтернативою було запропоновано підхід, заснований на інтеграції Node-RED і AWS IoT Core.

Основною перевагою розробленого у цій роботі підходу і те, що він інтуїтивніший, особливо для людей з досвідом розробки програмного забезпечення. Для більш складних завдань у Node-RED є вузол Function, який дозволяє створити невелику програму JavaScript. Такі вузли зручні для моделювання показників, оскільки дозволяють записати алгоритм звичному форматі, причому розроблена програма запускається в браузері при активації цього вузла повідомленням. При цьому Node-RED, як і Simulink, дозволяє зв'язувати компоненти в мережу візуально, що спрощує розуміння будови системи. Повідомлення, що передаються між вузлами, є об'єктами JavaScript, які описуються форматом JSON, що дозволяє легко задавати необхідні дані.

Візуальна складова та широка бібліотека компонентів також відіграють роль у забезпеченні більш легкого входження. Завдяки великій кількості доступних вузлів завдання, що часто зустрічаються, можна автоматизувати. Особливо це корисно для створення графічного інтерфейсу, оскільки від користувача потрібно лише налаштування кількох параметрів, на відміну повної реалізації зовнішнього вигляду з використанням HTML або сторонніх засобів розробки. Якщо необхідно автоматизувати виконання певної операції, для якої не доступні раніше створені вузли, Node-RED також пропонує можливість додати власний вузол.

Node-RED також відноситься до програмних продуктів з відкритим вихідним кодом і не потребує грошових витрат для використання, що може бути значною перевагою для невеликих команд. Крім того, доступність вихідного тексту Node-RED означає можливість як створення своїх вузлів, а й індивідуальної налаштування всього середовища.

Що стосується інтеграції Node-RED з AWS IoT Core, вона дозволяє пересилати дані між різними компонентами, у тому числі на хмарний сервер. У цій роботі використовувалася лише мала частина всіх можливостей - звичайний збір даних для подальшої візуалізації, а також використання віртуальної машини для запуску Node-RED. У перспективі використання хмарних технологій дозволяє зберігати результати до бази даних, а також проводити їх аналіз для виявлення тенденцій та складніших залежностей. Використання віртуальної машини також дає можливість віддаленого доступу, в тому числі спрощуючи роботу та демонстрацію результатів між кількома учасниками. AWS IoT Core також можна зв'язати з сервісом AWS Cloud Functions, що дозволяє виконувати ті чи інші операції при отриманні даних від моделі або системи, що тестується.

За всіх переваг розробленого підходу він також не позбавлений і недоліків, що обмежують його застосування. У порівнянні з Simulink описана модель програє точно через недоступність складних чисельних методів. Зрозуміло, деякого поліпшення можна досягти, реалізувавши їх самостійно, проте один цей факт значно ускладнює розробку настільки, що було б простіше використовувати Simulink. Проте варто зазначити, що потенційно така можливість існує, оскільки

Node-RED пропонує широкі можливості з персоналізації та індивідуального настроювання середовища роботи.

Node-RED також неможлива для реалізації моделей, що потребують трудомістких обчислень, зі значним ступенем прискорення внутрішнього часу. Цей недолік може виявлятися у випадках, коли необхідна невелика довжина кроку, зокрема, менше однієї хвилини за внутрішньосистемним часом. Графічний інтерфейс особливо страждає від цієї проблеми, оскільки він важко справляється вже з потоком повідомлень частіше ніж раз на секунду, хоча цей ефект більш помітний на складних візуальних об'єктах, таких як графіки. Однак для інших елементів важко досягти частоти повідомлень в секунду більше 10, тобто мінімальна довжина кроку симуляції за зовнішнім часом становить 0,1 с. Відповідно, моделювання протягом одного дня при довжині кроку в 1 хвилину за віртуальним часом займає 2,4 хвилини, одного місяця або 30 днів – 72 хвилини, одного року – 14,6 години. Таким чином, необхідні тимчасові ресурси для моделювання виявляються значними навіть частоти в 1 хвилину.

Незважаючи на виділені недоліки, підхід, заснований на інтеграції Node-RED та AWS IoT Core, показав себе добре і може бути рекомендований до використання.

У третій главі було вирішено такі задачи:

Розроблено модель навколишнього середовища.

Реалізовано взаємодію з системою, що тестується, для оцінки її роботи.

Створено інтерфейс користувача для управління проведенням експериментів.

Проведено імітаційні експерименти.

Проаналізовано результати та їх значення з погляду порівняння методів забезпечення енергоефективності.

Виділено переваги та недоліки використання розробленого методу тестування.

ВИСНОВКИ

В цій роботі ставилося таке завдання: дослідити можливість застосування інструменту розробки Node-RED та хмарного сервісу AWS IoT Core у сфері інтернету речей. Зокрема, метою було розроблення підходу до моделювання та тестування систем інтернету речей з їх використанням.

Як підготовка була проаналізована концепція інтернету речей, технологічні передумови її появи і складності, що виникають при впровадженні систем інтернету речей. Були розглянуті основні вимоги до таких програмних продуктів, особливості їх тестування. Як основний підхід було запропоновано використання імітаційного моделювання за допомогою Node-RED та AWS IoT Core.

Пробну сферу для перевірки запропонованого підходу було обрано домашню автоматизацію, зокрема системи управління енергоспоживанням. Було вивчено пропоновані в літературі підходи до вирішення цього завдання, виділено найперспективніші на основі використання статистичних даних про енергоспоживання домогосподарств. Було виділено основні компоненти таких систем, на основі яких було розроблено математичну модель будинку, що включає показники навколишнього середовища, необхідні для вирішення цього завдання.

Розроблена модель була реалізована з використанням Node-RED таAWS IoT Core, після чого вона була випробувана для порівняння кількох підходів

Розроблена модель дозволила оцінити переваги, одержувані при доповненні найпростішої системи, яка базується на контролі температури, деякими поведінковими компонентами.

В ході виконання даної роботи було розглянуто такі питання та вирішено такі завдання:

- вивчено концепцію інтернету речей та основні особливості цієї сфери, включаючи аналіз якості програмних продуктів;

- розглянуто проблеми тестування систем інтернету речей із особливим фокусом на використання імітаційного моделювання, а також можливість використання Node-RED та AWS IoT Core;
- проведено аналіз використання домашньої автоматизації для управління енергоспоживанням, включаючи середовище їхньої роботи, основні компоненти;
- описано основні підходи щодо забезпечення ефективності використання енергії;
- на основі виділених компонентів створено математичну модель показників навколишнього середовища, що використовуються для керування споживанням енергії.
- hозроблено систему тестування та моделювання систем інтернету речей з використанням Node-RED та AWS IoT Core.
- проведено імітаційний експеримент у сфері управління енергоспоживанням із аналізом його результатів.

Таким чином, проведене в рамках даної роботи дослідження дозволяє зробити висновок про можливість використання інструменту розробки Node-RED у поєднанні з хмарними технологіями, що надаються сервісом AWS IoT Core, та переваги такого використання, до яких можна віднести легкість входження, зручність реалізації, можливість перенесення розробленого прототипу середовище застосування. Підхід рекомендується використовувати при оцінці систем інтернету речей або їх прототипів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Johnston, WM Advances in Dataflow Programming Languages / WM Johnston, JRP Hanna, RJ Millar // ACM Computing Surveys. - 2004. - Vol. 36. - P. 1-34.

2. Pan, J. An Internet of Things Framework для Smart Energy в Buildings: Designs, Prototype, and Experiments / J. Pan [et al.] // IEEE Internet of Things Journal. - IEEE, 2015. - Vol. 2. - P. 527-537.

3. Stojkoska, BLR Перегляд Інтернету Things для smart home: Challenges and solutions / BLR Stojkoska, KV Trivodaliev // Journal of Cleaner Production. - Elsevier, 2017. - Vol. 140. - P. 1454-1464.

4. Pedrasa, MAA Coordinated Scheduling of Residential Distributed Energy Resources to Optimize Smart Home Energy Services / MAA Pedrasa, TD Spooner, IF MacGill // IEEE Transactions on Smart Grid. - IEEE, 2010. - Vol. 1. - P. 134-143.

5. Pritoni, M. Energy efficiency and misus of programmable thermostats: The effectiveness of crowdsourcing for understanding household behavior / M. Pritoni [et al.] // Energy Research & Social Science. - Elsevier, 2015. - Vol. 8 - P. 190-197.

6. Shrouf, F. Energy management базується на Інтернеті Things: практики і framework для adoption in production management / F. Shrouf, G. Miragliotta // Journal of Cleaner Production. - Elsevier, 2015. - Vol. 100. - P. 235-246.

7. Asare-Bediako, B. Integrated Energy Optimization with Smart Home Energy Management Systems / B. Asare-Bediako, PF Ribeiro, WL Kling // 3rdIEEE PES Innovative Smart Grid Technologies Europe, Berlin, 14-17 Oct. 2012/IEEE. - New York, 2013. - P. 1-8.

8. Beaudin, M. Home energy management systems: Review of modelling and complexity / M. Beaudin, H. Zareipour // Renewable і Sustainable Energy

Reviews. - Elsevier, 2015. - Vol. 45 - P. 318-335.

9. Node-RED [Electronic resource] - Mode of access: https://nodered.org/ - Date of access: 13.10.2023.

10. Adua, L. Reviewing the complexity of energy behavior: Technology, analytical traditions, and household energy consumption data in United States / L. Adua // Energy Research & Social Science. - Elsevier, 2020. - Vol. 59.

11. Simulink [Electronic resource] - Mode of access: mathworks.com/help/simulink/ - Date of access: 10.10.2023.

12. Zhou, B. Smart home energy management systems: Concept, configurations, and scheduling strategies / B. Zhou [et al.] // Renewable i Sustainable Energy Reviews. - Elsevier, 2016. - Vol. 61. - P. 30-40.

13. Software engineering – Product quality – Part 2: External metrics: ISO/IEC TR 9126-2:2003. - Intr. 07.2003. - Geneva: International Organization for Standardization, 2003. - 86 c.

14. Liu, J. The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes

15. Liu J. [et al.] // Proceedings of the 8thACM Conference on Embedded Networked Sensor Systems, Zurich, Nov. 2010, / Association for Computing Machinery. - New York, 2010. - P. 211-224.

16. Blackstock, M. Toward a Distributed Data Flow Platform для Web Things (Distributed Node-RED) / M. Blackstock, R. Lea // Proceedings of the 5thInternational Workshop on Web of Things. - 2014. - P. 34-39.

ДОДАТОК А

Реалізація вузлів, які відповідають за керування віртуальним часом

```
function timing(msg) {
      let t = global.get("simtime");
      let total_steps = global.get("total_steps");
      if (t > total_steps) {
          msg.payload = "END";
          return [null, msg];
    } else if (t >= 0) {
       msg.payload = t;
       return [msg,
       null];
          }
}
function outdoorTemp(msg) {let t = global.get("simtime")/global.get("steps_per_hour"); let ampl = global.get("T_out_ampl");
      let T_var = ampl * Math.cos (2 * Math.PI / 24 * (t - 14));

      msg.T_out = global.get("T_out") + T_var;
      return msg;
}
function indoorTemp(msg) {
      msg.T_in = global.get("T_in");
      return msg;
}

function PIR(msg) {
      let h = (global.get("simtime") / global.get("steps_per_hour")) % 24;

    if ((h >= 7 && h < 9) || (h >= 17 && h < 22))
       { msg.payload = "active";
            return msg;
      }
}
function door(msg) {
      let h = (global.get("simtime") / global.get("steps_per_hour")) % 24;

      if (h == 9) {
          msg.payload = "leave";
          return msg;
    } else if (h == 17) {
       msg.payload =
       "enter";     return
       msg;
          }
}
```

ДОДАТОК Б

Вузли системи, призначені для реагування повідомлення від датчиків

```
function tempDiff(msg) {
      let occupancy = global.get("hems.occupancy");

 if (occupancy === "AWAY") {
          return [null, msg];
      } else {
          let currT = msg.payload;
          let goalT = global.get("hems.goal_temp") [occupancy];

          msg.diff_T = goalT - currT;
          return [msg, null];
      }
}

function activity(msg) {
      let curr_time = global.get("simtime");

      if (msg.payload === "RESET") {
          context.set("last_active", -1);
} else if (msg.payload === "active") {
   context.set("last_active",
   curr_time); msg.payload = "on";
          return msg;
      } else { // timing signal
          let          last_active          =
   context.get("last_active"); if (curr_time -
   last_active > 1) {
              msg.payload = "off";
              return msg;
          }
      }
}

function occupancy(msg) {
      var state = context.get("state");

      if (state === "ACTIVE") {
          if (msg.payload === "off") {
              state = "ASLEEP";
   } else if (msg.payload === "leave")
       { state = "AWAY";
          }
      } else if (state === "AWAY") {
          if (msg.payload === "enter") {
              state = "ACTIVE";
          }
```

```
} else if (state === "ASLEEP")
    { if (msg.payload === "on")
    {
                state = "ACTIVE";
            }
       }

       context.set("state", state);
       msg.payload = state;
       return msg;

 }
```

ДОДАТОК В

Вузли для обробки температурних показань

```
 function heatFlow(msg) {
       let T_heater = global.get("T_heater");
       let T_indoor = msg.T_in;
       let Mdot = global.get("Mdot");
       let c = global.get("c");

       let heat_flow = (T_heater - T_indoor) *
Mdot  *  c;  msg.delta_Q_heater  =  heat_flow;
return msg;
 }
 function energyCost(msg) {
       let time = 1/global.get("steps_per_hour");
       let heat_flow = msg.delta_Q_heater;
       let cost_kWh = global.get("cost_kWh");

let  cumulative_cost  =  global.get("total_cost")  +
                      heat_flow    *    time    *
                      cost_kWh;
       global.set("total_cost", cumulative_cost);
       msg.cost = cumulative_cost;
       return msg;
 }
 function deltaQLosses(msg) {
       T_in = msg.T_in;
       T_out = msg.T_out;
       R_eq = global.get("R_eq");
       delta_Q_losses = (T_in - T_out) / R_eq;
       msg.delta_Q_losses = delta_Q_losses;
       return msg;
 }
 function updateTin(msg) {
       var T_in = global.get("T_in");
       let delta_Q_heater = msg.delta_Q_heater;
       let delta_Q_losses = msg.delta_Q_losses;
       let M_air = global.get("M_air");
       let c = global.get("c");
       delta_T_in  =  (delta_Q_heater  -  delta_Q_losses)  /
(M_air        *        c);        T_in        +        =
delta_T_in/global.get("steps_per_hour");
       global.set("T_in", T_in);
       msg.payload = delta_T_in;
       return msg;
 }
```

ДОДАТОК Г

Вузли, що відповідають за збирання та передачу даних

```
function saveAndStep(msg) {
      let            steps_per_hour            =
global.get("steps_per_hour");    let    steps    =
global.get("simtime");      let      hours      =
steps/steps_per_hour;
      let occupancy = global.get("hems.occupancy");
      let simdata = `${hours}, ${msg.T_in}, ${msg.T_out}, ${msg.cost},
${occupancy}` msg.payload = simdata;
      global.set("simtime", steps + 1);

      if (steps % steps_per_hour === 0) {
          indoors = {
              "payload": {
                  "payload": msg.T_in,
                  "timestamp": hours,
                  "topic": "in"
              }
          };
          outdoors = {
              "payload": {
                  "payload": msg.T_out,
                  "timestamp": hours,
                  "topic": "out"
              }
          };
          cost = {
              "payload": {
                  "payload": msg.cost,
                  "timestamp": hours,
                  "topic": "cost"
              }
          }
          return [msg, indoors, outdoors, cost];
      } else {
          return [msg, null, null, null];
      }
}

function formatCost(msg) {
      let cost = msg.payload.payload.toFixed(2) + "крб.";
      msg.payload = cost;
      return msg;
}

function backToMsg(msg) {
      let message = msg.payload;
```

```
for(const [key, value] of Object.entries(message))
    { msg[key] = value;
       }
       return msg;
 }
```