Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2024 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей »

Захищено на засіданні
Атестаційної комісії № 44
протокол № __ від __.06.2024 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**
(підпис) (прізвище та ініціали)

Виконав:
студент 4 курсу, групи КУ– 41
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – «Автоматизація та комп'ютерно-інтегровані технології»

ОКРУТНИЙ Роман Ярославович_____

Керівник: проф., д.т.н, професор ЗВО кафедри ТПС
АГЕЄВ Дмитро Володимирович_____

Рецензент: к.ф.-м.н., доцент ЗВО кафедри моделювання систем і технологій
КАРАСЬ Ірина Вячеславівна _____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і двох додатків. Загальний обсяг роботи складає 63 сторінок, з яких 49 сторінок основної частини з 21 рисунками, 19 найменуваннями списку використаних джерел та трьома додатками.

Тема дослідження: "Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей"

Метою кваліфікаційної роботи є розробка IoT мережі у Node-RED та Java-додатка за архітектурою MVC для збору та обробки даних з цієї мережі.

Об'єкт дослідження – процеси інтеграції IoT пристроїв в єдину мережу та взаємодії між ними і програмним забезпеченням.

Предмет дослідження – методи налаштування та керування IoT мережами в Node-RED, а також принципи побудови Java-додатків за архітектурою MVC для збору та обробки даних.

Методи дослідження складаються з аналізу існуючих методів інтеграції IoT пристроїв, експериментального налаштування IoT мережі в Node-RED, проектування та розробки Java-додатка за архітектурою MVC та тестування IoT мережі та Java-додатка для оцінки їх функціональності та ефективності.

Область застосування – використання є важливим для багатьох галузей, включаючи домашню автоматизацію, промисловий моніторинг та системи "розумного міста".

Актуальність роботи полягає в зростаючому значенні IoT технологій у сучасному світі. Інтеграція IoT пристроїв в єдину мережу та створення програмного забезпечення для збору та обробки даних стають критично важливими для багатьох галузей, включаючи домашню автоматизацію, промисловий моніторинг та системи "розумного міста".

Ключові слова: IoT, Node-RED, Java, MVC, сенсори, програмне забезпечення, моніторинг, автоматизація, обробка даних.

ABSTRACT

The explanatory note to the bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references and two appendices. The total volume of the work is 63 pages, of which 49 pages are the main part with 21 figures, 19 references and three appendices.

Research topic: "Model of an information system for monitoring the environment using Internet of Things technologies"

The explanatory note to the bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references and two appendices. The total volume of the work is 55 pages, including 47 pages of the main part with 21 figures, 19 references and two appendices.

The purpose of the qualification work is to develop an IoT network in Node-RED and a Java application based on MVC architecture for collecting and processing data from this network.

The object of research is the processes of integrating IoT devices into a single network and the interaction between them and software.

The subject of the study is the methods of setting up and managing IoT networks in Node-RED, as well as the principles of building Java applications based on the MVC architecture for data collection and processing.

The research methods consist of analyzing existing methods of integrating IoT devices, experimentally setting up an IoT network in Node-RED, designing and developing a Java application using MVC architecture, and testing the IoT network and Java application to evaluate their functionality and efficiency.

Scope - the use is important for many industries, including home automation, industrial monitoring and smart city systems.

The relevance of the work lies in the growing importance of IoT technologies in the modern world. The integration of IoT devices into a single network and the creation of software for data collection and processing are becoming critical for many industries, including home automation, industrial monitoring, and smart city systems.

Keywords: IoT, Node-RED, Java, MVC, sensors, software, monitoring, automation, data processing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 7

ВСТУП 8

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ ІНФОРМАЦІЙНИХ МОДЕЛЕЙ. ОГЛЯД АРХІТЕКТУРИ МЕРЕЖІ ІНТЕРНЕТУ РЕЧЕЙ 10

1.1 Огляд існуючих інформаційних систем 10

1.1.1 Управлінські інформаційні системи (MIS) 10

1.1.2 Системи підтримки прийняття рішень (DSS) 11

1.1.3 Інформаційні системи для управління знаннями (KMS) 11

1.1.4 Системи електронного документообігу (DMS) 12

1.1.5 Інформаційні системи для управління ланцюгами постачання (SCM) 14

1.1.6 Аналітичні системи та системи великих даних 15

1.1.7 Системи управління контентом (CMS) 16

1.2 Інтернет речей (IoT) 17

1.3 Аналіз технологій, які використовуються в IoT 19

1.3.1 Типи мереж IoT 19

1.3.2. Аналіз протоколів мережі IoT 20

1.4. Аналіз пристроїв мережі IoT 21

1.5. Аналіз архітектури IoT 22

1.5.1. Основні складові архітектури IoT 22

1.5.2. Сенсори та контролери 23

1.5.3. Шлюзи та системи збору даних 24

1.5.4 Центр обробки даних 25

1.6 Огляд завдань моніторингу у мережі IoT 26

Висновки до розділу 27
РОЗДІЛ 2 ОПИС ТА АНАЛІЗ ІНСТРУМЕНТІВ ДЛЯ МОНІТОРИНГУ IOT ... 29
2.1 Опис вибраної платформи Node-RED 29
2.2 Установка та налаштування Node-RED 30
2.3 Створення та налаштування IoT мережі. 31
2.3.1 Створення топології мережі. 31
2.3.2 Встановлення та налаштування MySQL 34
2.3.3 Налаштування компонентів мережі 34
Висновки до розділу 37
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАСОБІВ МОНІТОРИНГУ ЕЛЕМЕНТАМИ IOT 38
3.1 Опис архітектури та інструментів розробки 38
3.2 Архітектура серверного додатку 38
3.2.1 Опис мови програмування Java та фреймворку Spring Boot 39
3.1.3 MySQL та Hibernate 41
3.1.4 Freemarker 42
3.1.5 Thymeleaf 43
3.2 Реалізація серверної частини 43
3.2.1 Реалізація бізнес-логіки та моделей даних 43
3.2.2 Реалізація Бізнес-Логіки додатку 45
3.3 Реалізація інтерфейсу користувача 46
3.3.1 Налаштування системи 46
3.3.2 Налаштування серверної частини 47
3.4 Демонстрація роботи 47
3.4.1 Встановлення та запуск 47

3.4.2 Демонстрація роботи 47
Висновки до розділу 48
ВИСНОВКИ 50
СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ 53
Додаток А 55
Додаток Б 57
Додаток В 60

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

IoT - Internet of Things

API – Application Programming Interfac

СКБД – Система керування базами даних

ВСТУП

У сучасному світі інформаційні технології відіграють вирішальну роль у багатьох сферах людської діяльності, включаючи моніторинг. Однією з найбільш інноваційних технологій, що сприяє розвитку цієї галузі, є Інтернет речей (IoT). Технології IoT дозволяють створювати розумні системи, які збирають, обробляють і аналізують дані з різних датчиків у режимі реального часу, забезпечуючи тим самим оперативне та точне відстеження стану навколишнього середовища.

Актуальність теми обумовлена необхідністю створення ефективних інформаційних систем моніторингу, які дозволяють вчасно виявляти та реагувати на зміни у навколишньому середовищі. Зокрема, це важливо для контролю рівня забруднення повітря та води, метеорологічних умов, стану грунту, шумового забруднення та інших критичних показників. Використання технологій IoT у цій сфері відкриває нові можливості для автоматизації процесів збору даних та їх аналізу, що значно підвищує точність і швидкість реагування на екологічні загрози.

Метою цієї дипломної роботи є розробка моделі інформаційної системи моніторингу середовища з використанням технологій Інтернету речей. В рамках роботи буде проведено аналіз існуючих рішень у цій сфері, визначено основні вимоги до системи, розроблено архітектуру та реалізовано прототип системи, який дозволяє збирати дані з датчиків, обробляти їх та передавати на центральний сервер для подальшого аналізу.

Наукова новизна роботи полягає у розробці інтегрованої системи моніторингу, яка поєднує різні типи датчиків і використовує сучасні технології обробки та аналізу даних для забезпечення оперативного контролю стану навколишнього середовища.

Практична значущість роботи полягає в можливості використання розробленої системи для різних прикладних задач, таких як моніторинг міського середовища, контроль якості повітря у промислових зонах, відстеження погодних умов у сільськогосподарських регіонах тощо.

Таким чином, розробка моделі інформаційної системи моніторингу середовища на основі технологій Інтернету речей є важливим кроком до створення більш безпечного та здорового середовища для життя людини, сприяючи збереженню природних ресурсів і підвищенню якості життя.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ ІНФОРМАЦІЙНИХ МОДЕЛЕЙ. ОГЛЯД АРХІТЕКТУРИ МЕРЕЖІ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Огляд існуючих інформаційних систем

Інформаці́йна систе́ма (англ. Information system) — сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів. Вони стали невід'ємною частиною сучасних організацій, забезпечуючи ефективне управління ресурсами, підтримку прийняття рішень та операційні вдосконалення. Інформаційні системи є фундаментом для сучасних організацій, забезпечуючи ефективність, прозорість та інновації у бізнес-процесах. Вибір конкретної системи залежить від потреб організації, її розміру та галузі діяльності[1].

1.1.1 Управлінські інформаційні системи (MIS)

Управлінські інформаційні системи (MIS) — це комплекс апаратних і програмних засобів, призначених для збирання, зберігання, аналізу та представлення інформації, яка необхідна для управління організацією. MIS забезпечують своєчасний доступ до точних даних, що допомагає керівникам приймати обґрунтовані рішення. Вони інтегрують різні бізнес-процеси, полегшуючи координацію та управління ресурсами[2, 3]. Приклади MIS:

1. SAP ERP. Включає модулі для управління фінансами, людськими ресурсами, логістикою, виробництвом та іншими бізнес-процесами. Використовується великими підприємствами у різних галузях.
2. Oracle ERP Cloud. Це хмарне рішення, що пропонує модулі для фінансового управління, управління проектами, закупівель, управління ризиками тощо.

3. Microsoft Dynamics 365. Інтегрує функції ERP та CRM, дозволяючи організаціям ефективно управляти своїми операціями та взаєминами з клієнтами.

1.1.2 Системи підтримки прийняття рішень (DSS)

Система підтримки прийняття рішень (DSS) — це інтерактивна програмна система, яка збирає, обробляє та аналізує інформацію, допомагаючи користувачам приймати інформовані рішення. DSS використовуються для аналізу великої кількості даних, моделювання сценаріїв та прогнозування результатів. Системи підтримки прийняття рішень (DSS) є важливим інструментом для менеджерів та керівників, допомагаючи їм приймати обґрунтовані рішення на основі аналізу даних[4]. Ось деякі з існучих DSS-систем:

1. IBM Cognos Analytics. Ця система надає інструменти для звітності, аналізу даних та підтримки прийняття рішень. Включає можливості для візуалізації даних та прогнозного аналізу.
2. Microsoft Power BI. Забезпечує інтерактивну візуалізацію та бізнес-аналітику, інтегруючись з іншими продуктами Microsoft для зручного доступу до даних.
3. QlikView та Qlik Sense. Пропонують рішення для самообслуговування в аналізі даних, дозволяючи користувачам легко створювати візуалізації та аналізувати інформацію.
4. Tableau. Це інструмент для візуалізації даних, який дозволяє створювати інтерактивні та наочні звіти. Використовується для аналізу даних та прийняття рішень на основі отриманих інсайтів.

1.1.3 Інформаційні системи для управління знаннями (KMS)

Інформаційні системи для управління знаннями (Knowledge Management Systems, KMS) — це системи, що спрямовані на підтримку процесів створення, обміну, використання та збереження знань в організації. Вони включають технології та інструменти для збору, зберігання, пошуку та поширення знань, що дозволяє працівникам отримувати доступ до потрібної інформації в потрібний час. Це сприяє підвищенню ефективності роботи, інноваційності та конкурентоспроможності компаній[5]. Одним з найпопулярніших KMS систем вважаються SharePoint та Notion. Але частіше зустрічається SharePoint, адже це система від Microsoft, що інтегрується з Office 365, забезпечуючи управління документами, спільну роботу та збереження знань.

Система SharePoint дозволяє створювати спільні сайти, документи та інші ресурси, які легко доступні для спільної роботи та обміну знаннями серед співробітників. Вона також забезпечує можливість створення баз даних знань, де користувачі можуть зберігати та здійснювати пошук інформації за допомогою ключових слів та метаданих. Багато компаній використовують SharePoint для створення корпоративних порталів знань, де співробітники можуть отримати доступ до важливих даних, процедур та інструкцій.

Notion — це інша популярна KMS система, яка відрізняється своєю простотою використання та гнучкістю. Вона дозволяє створювати різні типи документів, списків, баз даних та інших ресурсів, організовувати їх у структуровані рубрики та легко ділитися ними з іншими користувачами. Notion надає можливість встановлювати зв'язки між різними елементами, що допомагає створювати зв'язані бази знань та процеси роботи.

1.1.4 Системи електронного документообігу (DMS)

Системи електронного документообігу (Document Management Systems, DMS) є важливими інструментами для організацій у цифровій епохі, де ефективне управління документами та інформацією має вирішальне значення.

Основними функціями таких систем є зберігання та організація документів, керування доступом, колаборація, версіонування, пошук та індексація, архівація та знищення. Системи електронного документообігу відіграють критичну роль у сучасному бізнес-середовищі, де обсяги інформації зростають експоненційно. Завдяки цим системам організації можуть легко керувати своїми документами, від початкового створення до кінцевого знищення. Вони забезпечують ефективне зберігання та структурування інформації, що робить доступ до неї простим і швидким.

Однією з ключових переваг таких систем є можливість керування доступом, що дозволяє точно визначати, хто і коли має доступ до конкретних документів. Це забезпечує конфіденційність та захист важливої інформації від несанкціонованого доступу. Крім того, можливість спільної роботи над документами у реальному часі полегшує комунікацію та співпрацю між співробітниками, навіть якщо вони знаходяться в різних частинах світу[6].

Завдяки функціям версіонування, системи електронного документообігу дозволяють зберігати історію змін кожного документа, що робить можливим відстеження розвитку інформації та відновлення попередніх версій у разі потреби. Пошукові та індексаційні інструменти спрощують процес знаходження необхідної інформації серед великого обсягу документів, що значно економить час та зусилля співробітників.

Наприклад, в банківській сфері системи електронного документообігу допомагають забезпечити дотримання регулятивних вимог щодо збереження документації та забезпечити швидкий доступ до важливих даних для ефективного вирішення клієнтських запитів. Виробничі підприємства використовують ці системи для керування технічною документацією, планування виробництва та ведення журналів якості, що допомагає забезпечити високу якість продукції та оптимізувати виробничі процеси.

1.1.5 Інформаційні системи для управління ланцюгами постачання (SCM)

Системи управління ланцюгами постачання (Supply Chain Management, SCM) — це комплексні інформаційні системи, що дозволяють ефективно керувати всіма аспектами логістики, виробництва, дистрибуції та інших етапів постачального ланцюга (Рис.1.1). Вони є ключовим інструментом для координації та оптимізації всіх процесів від постачальника до кінцевого споживача.

Ці системи допомагають управляти різними аспектами ланцюга постачання, включаючи планування запасів, прогнозування попиту, керування замовленнями, виробництво, транспортування товарів, складське управління та управління відносинами з постачальниками. Вони дозволяють підприємствам забезпечувати постачання товарів та послуг вчасно, знижувати витрати на логістику, оптимізувати запаси та підвищувати рівень обслуговування клієнтів.

SCM системи інтегрують дані та процеси з усіх ланцюга постачання в єдину платформу, що дозволяє отримувати цілісний погляд на всю логістичну мережу. Це сприяє покращенню координації між різними відділами компанії, зниженню запасів та скороченню часу доставки товарів до кінцевих клієнтів. Крім того, вони дозволяють виявляти та усувати проблеми в ланцюгу постачання, що допомагає забезпечити його ефективну роботу в умовах змінного бізнес-середовища[7].

Рисунок 1.1 - SCM системи

1.1.6 Аналітичні системи та системи великих даних

Аналітичні системи — це комплексні програмні рішення, спрямовані на аналіз даних для винесення висновків та прийняття рішень. Вони використовують різні методи аналізу, включаючи статистичний аналіз, машинне навчання та штучний інтелект, щоб витягти корисну інформацію з даних.

Системи великих даних (Big Data Systems) — це інфраструктурні рішення, призначені для зберігання, обробки та аналізу великих обсягів даних, які не можна ефективно обробити за допомогою традиційних методів. Вони включають в себе різні технології та інструменти для роботи з даними, включаючи розподілені системи зберігання, обробки та аналізу даних.

Аналітичні системи та системи великих даних є ключовими компонентами в сучасному світі, де величезні обсяги даних генеруються кожну секунду. Ці системи допомагають організаціям аналізувати, розуміти та використовувати ці дані для прийняття бізнес-рішень та виявлення трендів.

1.1.7 Системи управління контентом (CMS)

CMS — це система управління контентом. По-простому, її ще називають «движком» сайту. CMS дозволяє наповнити сайт контентом, а також трансформувати його зовнішній вигляд, і часто це можна зробити, не знаючи коду, а просто скориставшись графічним інтерфейсом. Загалом можна виокремити наступні різновиди таких систем:

1. Web content management systems для управління вебсайтами (наприклад, енциклопедіями, подібними до Вікіпедії, онлайн-виданнями, блогами, форумами, корпоративними чи персональними вебсторінками та ін.)
2. Транзакційні СКВ для забезпечення транзакцій у електронній комерції.Інтегровані СКВ для роботи з документацією на підприємствах.
3. Електронні бібліотеки (Digital Asset Management) для забезпечення циклу життя файлів електронних медіа (відео, графічн., презентації тощо).
4. Системи для забезпечення циклу життя документації (інструкції, довідники, описи).
5. Освітні СКВ — системи для організації Інтернет курсів та відповідного циклу життя документації.

Інформаційні системи є фундаментом для сучасних організацій, забезпечуючи ефективність, прозорість та інновації у бізнес-процесах. Вибір конкретної системи залежить від потреб організації, її розміру та галузі діяльності[7].

Рисунок 1.2 - Системи управління контентом (CMS)

1.2 Інтернет речей (IoT)

Інтернет речей (IoT) – це відносно новий підхід, коли Інтернет перестає бути просто мережею комп'ютерів і людей, а стає мережею розумних об'єктів і речей. Завдяки швидкому розвитку технологій, Інтернет речей перетворюється на глобальну систему, де всі пристрої з'єднані між собою через Інтернет (Рис.1.1). Це динамічний напрямок, який активно досліджують сьогодні. Завдяки постійному вдосконаленню сенсорів, мікроконтролерів та зв'язку, тепер можливо підключити до Інтернету навіть найзвичайніші предмети: від холодильників та світильників до автомобілів та медичних пристроїв. Ця нова глобальна система створює безліч можливостей для зручності, ефективності та автоматизації в усіх сферах життя. Зв'язок через Інтернет дозволяє цим об'єктам обмінюватися даними, отримувати віддалені команди та взаємодіяти один з одним, що відкриває нові можливості для побудови інтелектуальних систем, які працюють автономно та оптимізують свою роботу в реальному часі[8].

Загалом, Інтернет речей перетворюється на глобальну мережу, де кожен розумний об'єкт може бути підключений до всього світу, створюючи нову еру взаємодії та технологічного прогресу.

Рисунок 1.3 - Приклад мережі IoT

IoT полягає в об'єднанні даних з різних джерел на віртуальній платформі або існуючій Інтернет-інфраструктурі. Концепція Інтернету речей з'явилася в 1982 році, коли до Інтернету підключили автомат з газованою водою, який повідомляв про наявність напоїв і їх температуру. У 1991 році Марк Вайзер вперше дав сучасну оцінку цієї концепції. У 1999 році Білл Джой запропонував ідею зв'язку між пристроями, а Кевін Ештон ввів термін «Інтернет речей» для позначення пов'язаних пристроїв.

Основна ідея IoT - це можливість обміну корисною інформацією між унікально ідентифікованими пристроями реального світу, оснащеними сучасними технологіями, такими як радіочастотна ідентифікація (RFID) і бездротові сенсорні мережі (WSNs). Ці пристрої можуть самостійно приймати рішення і виконувати автоматизовані дії. У 2005 році Міжнародний союз електрозв'язку (ITU) оголосив про настання ери всепроникаючих мереж, де мережі взаємодіють між собою. Головна ідея IoT - створення середовища, в якому речі можуть виконувати команди і обробляти дані для досягнення бажаних результатів за допомогою машинного навчання. Практичне втілення IoT можна побачити в Twine, компактному і малопотужному пристрої, який працює з мережевим програмним забезпеченням в реальному часі, перетворюючи цю концепцію на реальність. Проте різні люди та організації можуть мати свої уявлення про Інтернет речей [9].

1.3 Аналіз технологій, які використовуються в IoT

В Інтернеті речей основними елементами є глибоко вбудовані пристрої з такими характеристиками, як вузька смуга пропускання, регулярний збір даних та невеликий обсяг передаваних даних. Ці пристрої обмінюються інформацією між собою та надають її через різні інтерфейси. Деякі з них, наприклад, охоронні відеокамери високої роздільної здатності або VoIP-відеотелефони, потребують широкосмугового зв'язку для передачі даних у режимі реального часу. Однак, більшість пристроїв IoT обмінюються даними лише час від часу, передаючи невеликі пакети інформації [8].

1.3.1 Типи мереж IoT

Для розгляду мережі IoT варто розпочати з розгляду типів мереж, що використовуються в її побудові.

1. Бездротові мережі (Wireless Networks). Вони є основою для багатьох IoT застосувань, оскільки використовують стандарти, такі як Wi-Fi, Bluetooth, Zigbee, Z-Wave та інші. Це дає можливість простішого підключення та розгалуження мережі.
2. Дротові мережі (Wired Networks). Хоча багато IoT пристроїв бездротові, Існують і дротові з’єднання, такі як Ethernet або Powerline, для підключення до Інтернету або до центральних систем.
3. Mesh Networks. Це мережі, де кожен вузол може бути як точка доступу для інших пристроїв мережі. Це створює гнучкі та надійні мережі, де можливі з'єднання із запасом та безпеки.
4. LPWAN (Low-Power Wide-Area Network). Ці мережі призначені для підключення великої кількості пристроїв на великі відстані, використовуючи малу потужність. Наприклад, Sigfox та LoRaWAN є популярними протоколами LPWAN.

5. Супутникові мережі (Satellite Networks). Вони можуть бути використані для підключення IoT пристроїв в областях, де доступність звичайних мереж обмежена, наприклад, віддалені райони або морські об'єкти.

Це лише декілька типів мереж, і вони можуть комбінуватися для створення складних інфраструктур для різноманітних IoT застосувань[8].

1.3.2. Аналіз протоколів мережі IoT

Розглянемо тепер протоколи, які використовуються в мережі IoT. Пристрої IoT обмінюються даними за допомогою протоколів IoT. Ось декілька основних протоколів та їхніх особливостей:

1. AMQP (Advanced Message Queuing Protocol). Стандартизує обмін повідомленнями між пристроями на промисловому рівні.
2. CoAP (Constrained Application Protocol). Мережевий протокол з обмеженою пропускною здатністю для пристроїв з обмеженою потужністю.
3. DDS (Data Distribution Service). Універсальний протокол для мереж з високою продуктивністю, що спрощує розгортання та зменшує складність.
4. MQTT (Message Queuing Telemetry Transport). Протокол обміну повідомленнями для легких з'єднань з низькою пропускною здатністю.

На рівні транспорту мережеві дані захищаються та передаються за допомогою протоколів TCP та UDP:

1. TCP (Transmission Control Protocol). Забезпечує обмін даними між вузлами, розбиваючи дані на пакети.
2. UDP (User Datagram Protocol). Забезпечує швидку передачу даних без гарантії надійності.

На рівні мережі встановлюється комунікаційний канал:

1. 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks). Версія IPv6 для бездротових мереж з меншою потужністю.
2. IPv6 (Internet Protocol version 6). Остання версія протоколу IP для ідентифікації пристроїв в мережі.

На рівні каналу передачі дані передаються за допомогою стандартів, таких як IEEE 802.15.4, для створення бездротових мереж. Ці протоколи та стандарти забезпечують надійну комунікацію та обмін даними між пристроями IoT у різних середовищах.

1.4. Аналіз пристроїв мережі IoT

У мережі Інтернету речей використовуються різноманітні пристрої залежно від їхніх функцій та можливостей:

1. Вбудовані системи. Ці системи включають як обладнання, так і програмне забезпечення і використовуються для управління певними функціями, пов'язаними з більшою системою. Вони базуються на мікропроцесорах або мікроконтролерах.
2. Інтелектуальні системи. Ці пристрої можуть виконувати обчислення і часто оснащені мікроконтролером.
3. Мікроконтролери (MCU). Це невеликі комп'ютери, вбудовані в мікросхеми і обладнані центральним процесором, оперативною та постійною пам'яттю. Вони призначені для виконання простих завдань.
4. Мікропроцесори (MPU). Це також центральні процесори, але вони можуть бути розміщені на одній або декількох інтегральних мікросхемах. Вони мають більше обчислювальну потужність, ніж мікроконтролери.
5. Пристрої, які не призначені для обчислень. Ці пристрої лише встановлюють підключення і передають дані, без можливості виконання обчислень.

6. Перетворювачі. Це фізичні пристрої, які перетворюють один вид енергії в інший. Вони можуть включати внутрішні датчики та виконавчі пристрої, що передають дані про взаємодію об'єктів з їхнім середовищем[10].

1.5. Аналіз архітектури IoT

Архітектура Інтернету речей визначається великою різноманітністю пристроїв та систем, які її створюють. Фрагментація є однією з основних проблем IoT через різноманітний характер підключених речей. Для досягнення успішної реалізації будь-якої системи IoT необхідно інтегрувати всі ресурси, обладнання та програмне забезпечення в єдину структуру, що дозволить створити інтегроване, надійне та економічно ефективне рішення. Головна задача архітектури IoT полягає у забезпеченні можливості виконувати завдання системи з ефективним використанням ресурсів та максимальною придатністю.

1.5.1. Основні складові архітектури IoT

Архітектура Інтернету речей (IoT) має кілька основних складових:

1. Речі, підключені до Інтернету. Це об'єкти, які мають вбудовані датчики та виконавчі механізми і можуть взаємодіяти з навколишнім середовищем, а також збирати та передавати дані через IoT шлюзи.
2. Системи збору даних IoT та шлюзи. Ці системи збирають, фільтрують та попередньо обробляють велику кількість необроблених даних, перетворюючи їх у цифрові потоки готові для аналізу.
3. Крайові пристрої (Edge devices). Вони відповідають за подальшу обробку та аналіз даних. Тут також можуть використовуватися технології візуалізації та машинного навчання для розуміння отриманих даних.

4. Центри обробки даних. Дані передаються в центри обробки даних, які можуть бути розміщені хмарно або локально. Тут дані зберігаються, управляються та аналізуються для отримання практичної інформації.

1.5.2. Сенсори та контролери

Сенсори та контролери є важливими компонентами будь-якої системи Інтернету речей (IoT), відповідальними за збір даних, які становлять основу IoT. Для отримання інформації про фізичні параметри навколишнього середовища або в середині об'єктів використовуються різноманітні датчики (Рис.1.4). Ці датчики можуть бути вбудованими у пристрої або функціонувати автономно для вимірювання та збору даних. Наприклад, в промисловості використовуються датчики тиску та температури для моніторингу умов виробництва, а у медицині - датчики пульсу та кров'яного тиску для контролю стану пацієнтів. Важливим елементом цього рівня є виконавчі механізми, які разом із сенсорами можуть перетворювати зібрані дані на конкретні дії. Наприклад, в автоматизованих системах управління будинком, на основі даних від датчиків про рівень освітленості та температуру можуть автоматично регулюватися жалюзі та обігрівачі. Усі ці процеси відбуваються автоматично, без необхідності втручання людини. Також важливо, що підключені пристрої мають здатність співпрацювати не лише з шлюзами або системами збору даних, але й між собою для обміну інформацією та взаємодії в реальному часі. Це забезпечує ефективне використання ресурсів у всій системі. Проте, у випадку обмежених ресурсів та пристроїв, що працюють від акумуляторів, це завдання ускладнюється, оскільки така взаємодія вимагає значної обчислювальної потужності та споживає енергію та пропускну здатність. Тому надійна архітектура повинна забезпечувати ефективне управління пристроями, використовуючи спеціальні, безпечні та легкі протоколи зв'язку, наприклад такі як Lightweight M2M[11].

Рисунок 1.4 - Апаратні складові IoT

1.5.3. Шлюзи та системи збору даних

Шлюзи та системи для збору даних грають ключову роль у структурі IoT, оскільки вони функціонують у тісному зв'язку з датчиками та виконавчими механізмами. Вони забезпечують збір, фільтрацію та передачу даних до локальної інфраструктури та хмарових платформ. Шлюзи та системи збору даних діють як посередники між підключеними пристроями та хмарою, забезпечуючи зв'язок між різними рівнями системи (Рис.1.5). Вони знаходяться між операційними та інформаційними технологіями, забезпечуючи зв'язок між датчиками та іншими системами, перетворюючи дані датчиків у формати, придатні для передачі та використання іншими компонентами системи. Крім того, вони можуть керувати, обробляти та відфільтровувати дані, що сприяє зменшенню обсягу інформації, яка передається до центральних систем. Це допомагає ефективно використовувати мережевий трафік та скорочує час очікування відповіді. Таким чином, порти можуть проводити передпочаткову обробку даних, упаковуючи їх у компактні пакети, готові для подальшого використання. Забезпечення безпеки є ще однією ключовою функцією їх роботи. Оскільки вони контролюють потік інформації у двох напрямках, вони можуть використовувати шифрування та інші заходи захисту, щоб убезпечити дані IoT від витоку до центральних систем та знизити ризик кібератак на пристрої IoT[11].

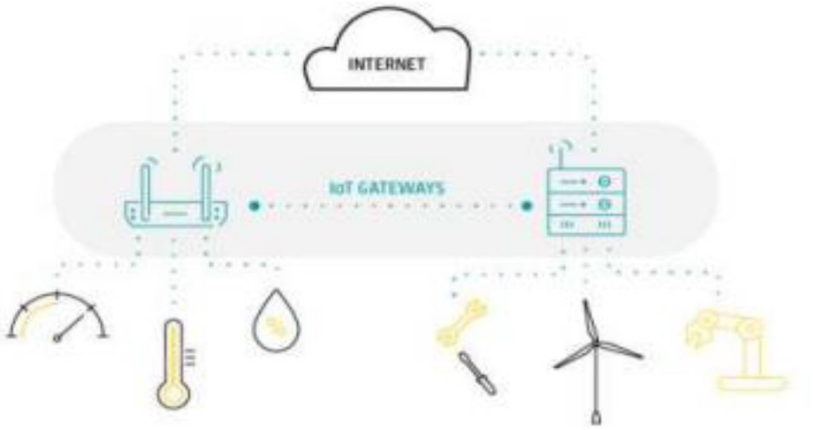

Рисунок 1.5 - Шлюзи та системи збору даних

1.5.4 Центр обробки даних

Центр обробки даних, або ж хмарна система, можна уявити як мозок у тілі Інтернету речей, якщо датчики є нейронами, а шлюзи - основою IoT. У протилежність крайовим рішенням, центр обробки даних або хмарна система призначені для зберігання, обробки та аналізу великих обсягів даних за допомогою потужних механізмів аналізу даних та машинного навчання, які крайові системи не можуть підтримати (Рис.1.6). Спостерігаючи зростання впровадження, особливо в промисловій архітектурі IoT, хмарні обчислення сприяють підвищенню продуктивності, скороченню незапланованих простоїв та зменшенню витрат на енергію, а також приносять багато інших переваг для бізнесу. У випадку використання хмарних обчислень разом із відповідними програмними рішеннями, вони можуть забезпечувати бізнес-аналітику та варіанти презентації, які допомагають людям взаємодіяти з системою, контролювати та керувати нею, а також приймати обґрунтовані рішення на основі звітів, інформаційних панелей та даних, що можна переглядати в режимі реального часу[12].

Рисунок 1.6 - Центр обробки даних

1.6 Огляд завдань моніторингу у мережі IoT

Моніторинг у мережі Інтернет речей (IoT) представляє собою складний процес, який включає в себе широкий спектр аспектів та вимог. Починаючи зі збору даних та контролю за системою, моніторинг простежує за функціонуванням як пристроїв самого IoT, так і їхнього зовнішнього середовища, а також процесів, які з ними пов'язані.

У різних сценаріях систем IoT, які потребують моніторингу, можуть бути задіяні не лише пристрої, але й зовнішнє середовище та процеси, що з ними пов'язані. Наприклад, розглянемо моніторинг велосипедиста, який подорожує по дорозі. Тут моніторинг може охоплювати різні параметри, від внутрішніх, таких як фізичні показники велосипедиста, до зовнішніх, таких як нахил дороги чи погодні умови.

Поняття моніторингу також включає в себе як апаратні, так і програмні аспекти. Показники апаратного забезпечення можуть охоплювати такі вимірювання, як напруга акумулятора чи сила радіосигналу. З програмного ж точку зору, моніторинг може включати як застосункове, так і системне програмне забезпечення, включаючи операційну систему чи мережевий стек.

Розуміння взаємодії між програмними компонентами, які працюють на пристрої та в хмарі, також є ключовим аспектом моніторингу. Це передбачає розуміння доступу до інформації для обох систем та їхню здатність поєднувати цю інформацію. Наприклад, для вирішення питання про час доставки

повідомлення до хмари можуть бути використані метрики, які включають час публікації пристрою та час прибуття повідомлення до хмари.

Повне рішення моніторингу передбачає контроль як основних, так і допоміжних компонентів. Наприклад, моніторинг коду програми на пристрої може включати в себе моніторинг білих скриньок, який дозволяє бачити, як саме працює програма. Однак можуть виникати ситуації, коли необхідно застосовувати моніторинг чорних ящиків. Це може статися, наприклад, у випадку перевірки API та інших хмарних сервісів, які впливають на роботу системи.

У сфері IoT широко використовуються датчики для спостереження за навколишнім середовищем. Фізичні датчики, такі як датчики температури, можуть бути прямо пов'язані з логічними метриками, але їхні дані часто потребують певної обробки. Наприклад, датчик акселерометра може постійно вимірювати силу G, але метрика, яка відображає тремтіння, вимагає спеціального алгоритму для її визначення.

Після визначення систем, які підлягають моніторингу, необхідно встановити, для чого саме здійснюється цей моніторинг. Моніторинг повинен гарантувати належне функціонування послуги чи функції системи. Наприклад, при моніторингу фізичних процесів, таких як обігрів приміщення, важливо вимірювати параметри, такі як години роботи двигуна, для забезпечення ефективної роботи системи[13].

Висновки до розділу

Моніторинг у мережі Інтернет речей (IoT) є ключовим елементом для забезпечення належної функціональності, ефективності та безпеки систем. Він включає в себе різноманітні аспекти, починаючи зі збору даних через контроль за системою, якою користується людина. У різних сценаріях системою IoT

можуть бути не лише пристрої, але й зовнішнє середовище та процеси, пов'язані з пристроєм.

Моніторинг може охоплювати апаратні та програмні аспекти, включаючи як застосункове, так і системне програмне забезпечення. Важливо забезпечити взаємодію між програмними компонентами, що працюють на пристрої та в хмарі. Контроль як основних, так і допоміжних компонентів є ключовим для повного рішення моніторингу.

У сфері IoT широко використовуються фізичні датчики для спостереження за навколишнім середовищем. Моніторинг передбачає отримання загального уявлення про те, як змінюються показники з часом, відмінно від реєстрації пристроїв, яка фокусується на деталях конкретних подій.

Загалом, ефективний моніторинг у мережі IoT допомагає забезпечити надійність, безпеку та ефективність систем, що використовуються, і є важливою складовою успішного функціонування цих систем, включаючи інформаційні системи, які дозволяють структурувати та аналізувати дані, збирані з різних джерел, і забезпечують зрозумілість та легкість взаємодії зі зібраними даними.

РОЗДІЛ 2
ОПИС ТА АНАЛІЗ ІНСТРУМЕНТІВ ДЛЯ МОНІТОРИНГУ ІОТ

2.1 Опис вибраної платформи Node-RED

Node-RED - це інноваційний візуальний інструмент, який розробляється відкритою спільнотою для створення розумних інтернет-рішень. Він надає можливість створення потужних аплікацій шляхом використання графічного інтерфейсу, що дозволяє спрощувати розробку та інтеграцію різноманітних компонентів. Node-RED дозволяє користувачам створювати потокові процеси, в яких взаємодіють різні вузли. При цьому вузли можуть виконувати різні функції, включаючи обробку даних, взаємодію зі зовнішніми сервісами та пристроями, відправку та отримання повідомлень тощо[14].

У порівнянні з Cisco Packet Tracer, Node-RED також надає можливість експериментувати та тестувати різноманітні сценарії без необхідності реального обладнання. Він є ідеальним інструментом для навчання, розробки та прототипування IoT (Internet of Things) рішень, оскільки дозволяє швидко створювати складні системи за допомогою інтуїтивного візуального інтерфейсу.

Використовуючи Node-RED, користувачі можуть легко і швидко створювати різноманітні застосунки, включаючи системи моніторингу, керування домашніми пристроями, збір та аналіз даних, автоматизацію процесів та багато іншого (Рис. 2.1). Він є потужним інструментом для розробки розумних та інноваційних рішень, які використовують переваги підключеного світу.

Node-RED працює за принципом потокового програмування, де вузли (nodes) розміщуються на робочій області та з'єднуються лініями, що представляють потоки даних. Кожен вузол виконує окрему функцію, і вузли можуть бути сконфігуровані для виконання специфічних завдань, таких як збирання даних з сенсорів, обробка інформації, взаємодія з API або керування пристроями.

Node-RED має величезну бібліотеку готових вузлів, які можна легко додавати та використовувати у проєктах. Наприклад, вузли для роботи з MQTT, HTTP, WebSockets, базами даних та різноманітними протоколами дозволяють створювати інтегровані рішення без необхідності писати багато коду.

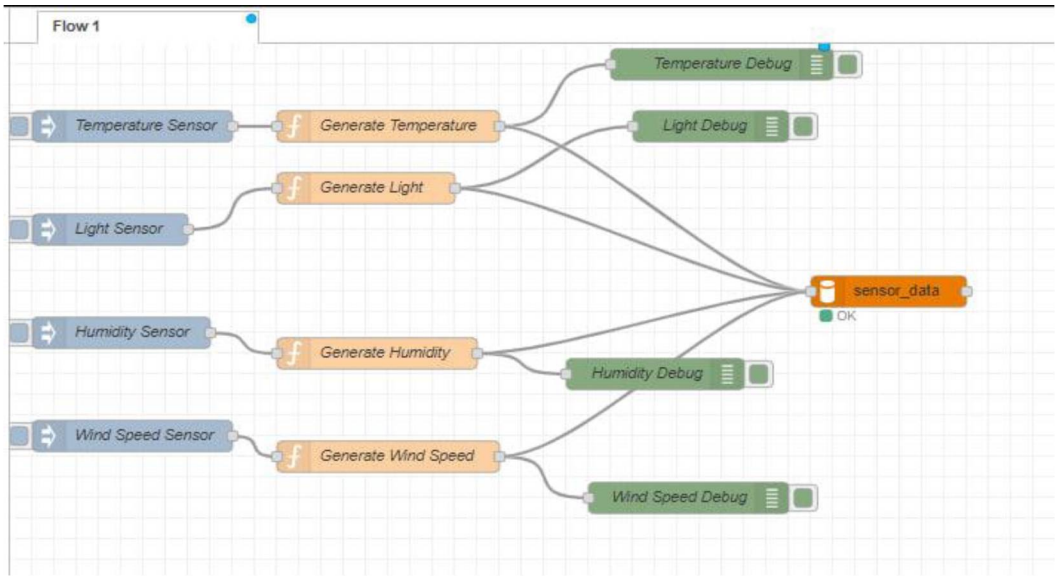

Рисунок 2.1 – Приклад побудованої мережі у Node-RED

2.2 Установка та налаштування Node-RED

Для установки Node-RED не потрібно зареєструватися на офіційному сайті, оскільки Node-RED розробляється відкритою спільнотою та надається безкоштовно в рамках відкритої ліцензії.

Отже, ось кроки необхідні для установки та запуску Node-RED:

1. Необхідно завантажити встановлювач програми з офіційного сайту Node-RED.
2. Запустити встановлювач та дотриматись інструкцій на екрані для завершення процесу установки.
3. Після завершення установки, необхідно запустити програму Node-RED за допомогою команди node-red (Рис.2.2).

По замовчуванню, Node-RED працює на порті 1880. Таким чином, після успішного запуску програми ви можете звертатися до інтерфейсу Node-RED через веб-браузер за адресою: 'http://localhost:1880'.

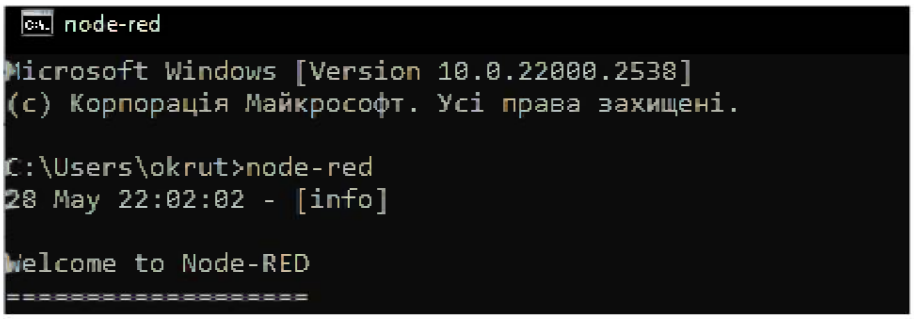

Рисунок 2.2 - запуск програми Node-RED

2.3 Створення та налаштування IoT мережі.

Перед створенням необхідно визначити які датчики необхідні для створюванної моделі. Для проекту достатнім буде використання 4 датчиків: датчиків температури, вологості, рівня освітлення та швидкості вітру. Ці датчики надають великий спектр можливостей. Наприклад вони можуть бути використанні на умовній метеостанції. Саме показники цих основних сенсорів є важливішими для передбачення погоди та моніторингу кліматичних змін.

2.3.1 Створення топології мережі.

На панелі Node-RED кожен об'єкт має своє місцезнаходження та функціональність, яка спрощує навігацію та редагування в потоках даних.

Основою мережі є Inject вузли, які розміщуються у верхній частині панелі, оскільки вони відповідають за введення початкових даних або подій у потік (Рис.2.3). Вони мають значок, що виглядає як стрілка, що виходить з області вводу, щоб показати, що вони початкові точки потоку.

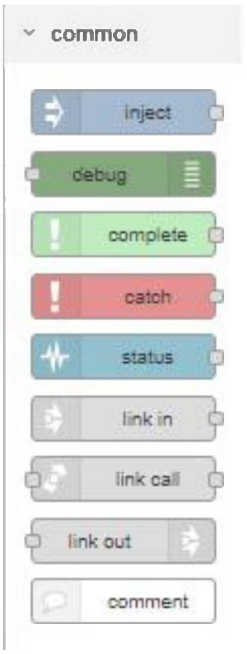

Рисунок 2.3 – Inject та Function вузли

Вузли функції необхідні для рандомізації та запису в сгенерованних данних до MySQL. Ці вузли розташовані після Inject вузлів у середній частині панелі. Вони мають іконку з квадратом та кружечком, щоб вказати, що вони виконують функції обробки даних (Рис2.4).

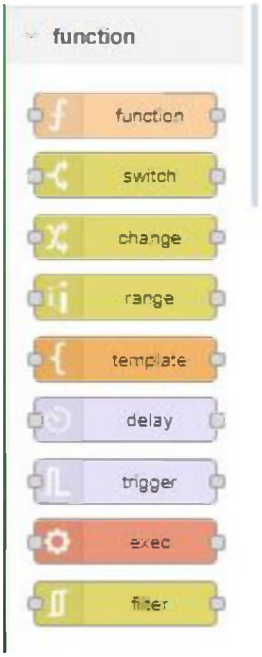

Рисунок 2.4 – Function вузли

Правильність генерації даних можна перевірити, під’єднавши кожну функцію з Debug вузлом відповідно (Рис.2.3).

Після перевірок коректності генерації даних, необхідно додати точку запису в MySQL. Але за замовчуванням її немає на доступній панелі. Щоб розширити можливості Node-RED для роботи з MySQL, треба встановити додаткові пакети або доповнення. Для цього необхідно відкрити панель керування Node-RED, натиснувши на іконку у верхньому правому куті, а потім вибрати "Manage palette". У вікні керування палітрою треба знайти та встановити пакети, пов'язані з MySQL, наприклад, "node-red-node-mysql", який дозволяє взаємодіяти з базою даних MySQL безпосередньо з Node-RED (Рис.2.5). Після встановлення пакету, нові вузли та функції для роботи з MySQL з'являться у панелі Node-RED (Рис.2.6).

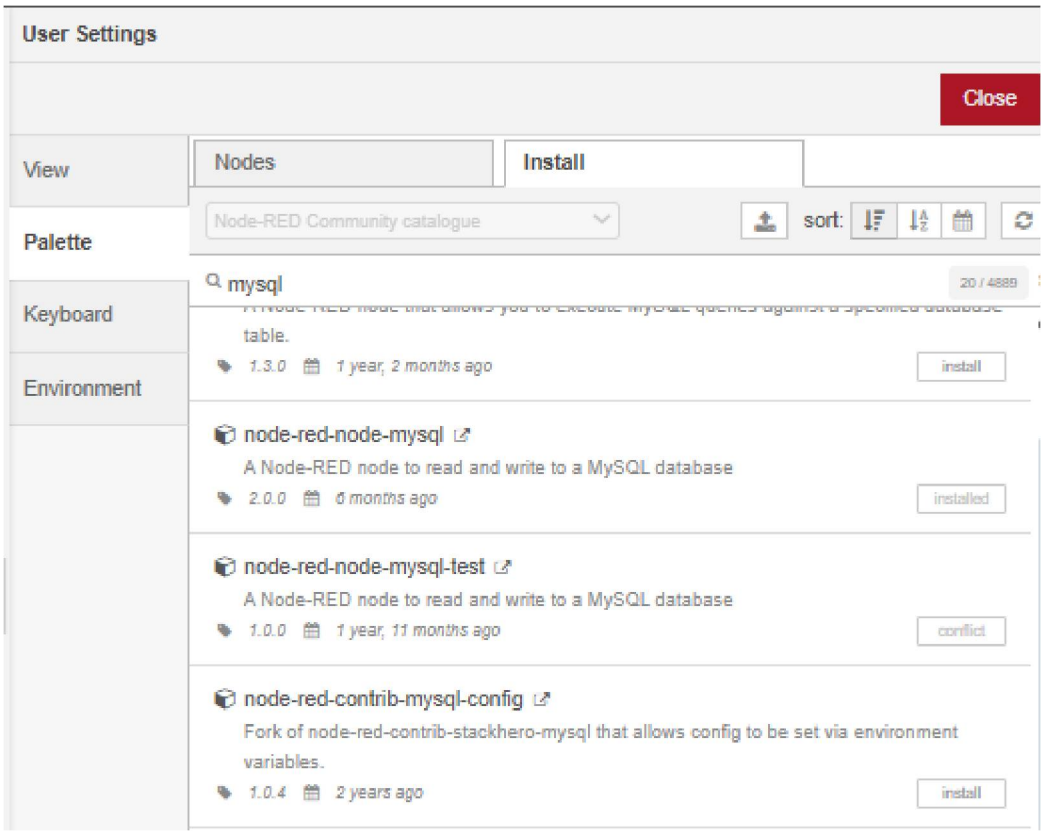

Рисунок 2.5 - Пакети для роботи з MySQL

Рисунок 2.6 - Точку запису в MySQL

2.3.2 Встановлення та налаштування MySQL

Для встановлення MySQL необхідно завантажити встановлювач програми з офіційного сайту та виконати кроки по встановленню. Після чого треба створити користувача та пароль. За замовчуванням MySQL працює на порту 3306, це знадобиться надалі. Для роботи з базою даних можна використати MySQL Workbench. MySQL Workbench — інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безкоштовне оточення для системи баз даних MySQL. Використовуючи мову запитів SQL треба створити базу даних з назвою sensor_data. База даних буде мати 4 таблиці. Кожна таблиця буде мати унікальний ідентифікатор (id), значення, яке вимірюється (value), і позначку часу (timestamp), що відображає момент, коли дані були зібрані. Ці таблиці дозволять ефективно зберігати та керувати даними, що надходять від різних сенсорів, у вашій базі даних MySQL[15].

2.3.3 Налаштування компонентів мережі

Після додавання вузлів, їх необхідно налаштувати. Для вузлів inject необхідно вказати ім’я вузла, тип передаванних даних та частоту з якою вузол буде їх

генерувати. Для цього треба зробити 2 кліки по відповідному вузлу та заповнити поля (Рис.2.8).

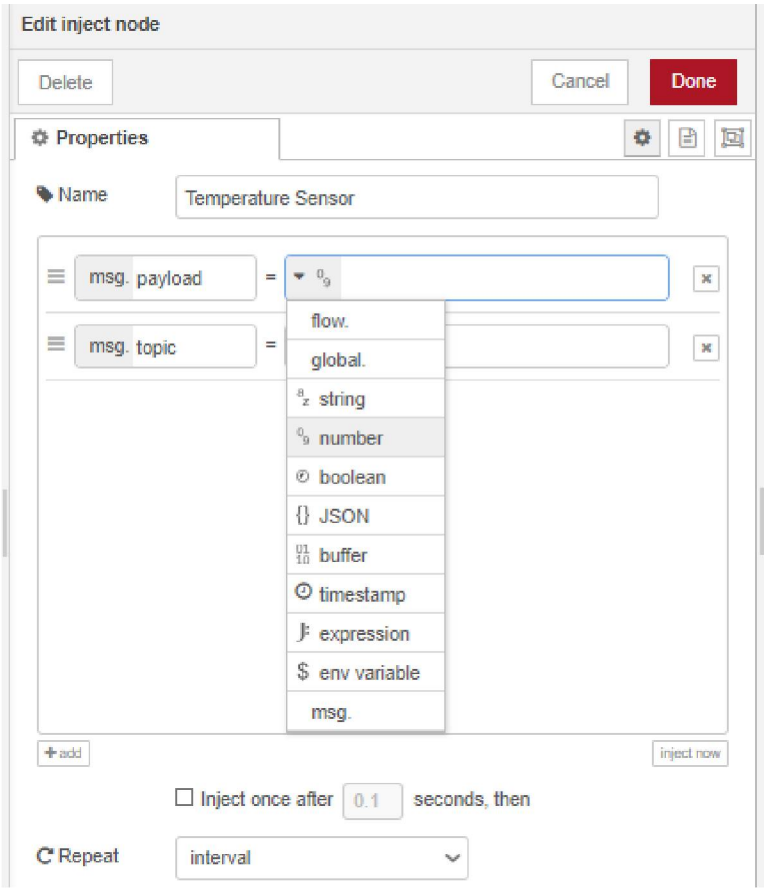

Рисунок 2.8 – налаштувати для вузлів inject

Далі необхідно створити функції генерації даних та передачу їх до MySQL. Функції можна сворити на мові JavaScript, яка використовується в Node-RED. Ці функції створюють SQL-запити для вставки випадкового значення датчику та поточного часу в таблицю відповідну бази даних MySQL (Рис.2.9). Перевірити коректність даних можна натиснувши на кнопку у кожного Debug вузла та відкривши вікно Debug у правому верхньому куті.

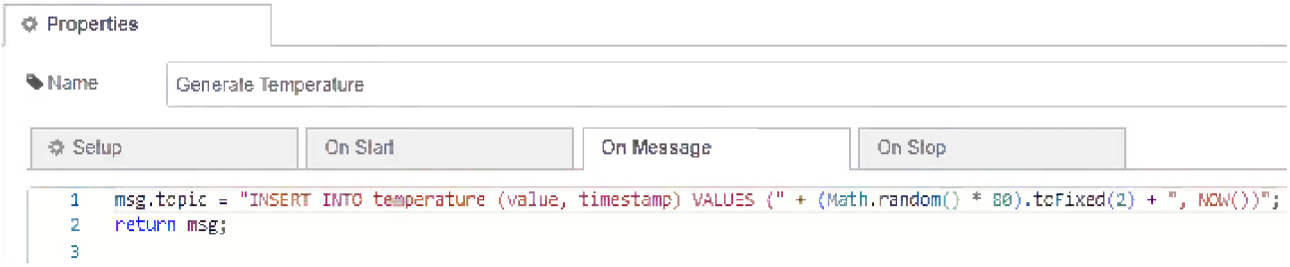

Рисунок 2.9 – Функція для датчика температури

Далі необхідно налаштувати точку запису в MySQL. Для цього необхідно два рази клацнути на доданий вузол та заповнити поля доступу до бази даних (Рис.2.10). Після налаштувань можна зробити остаточне з'єднання між функціями та точкою запису. І зберегти робочий проект натиснувши на червону кнопку Deploy у правому верхньому куті.

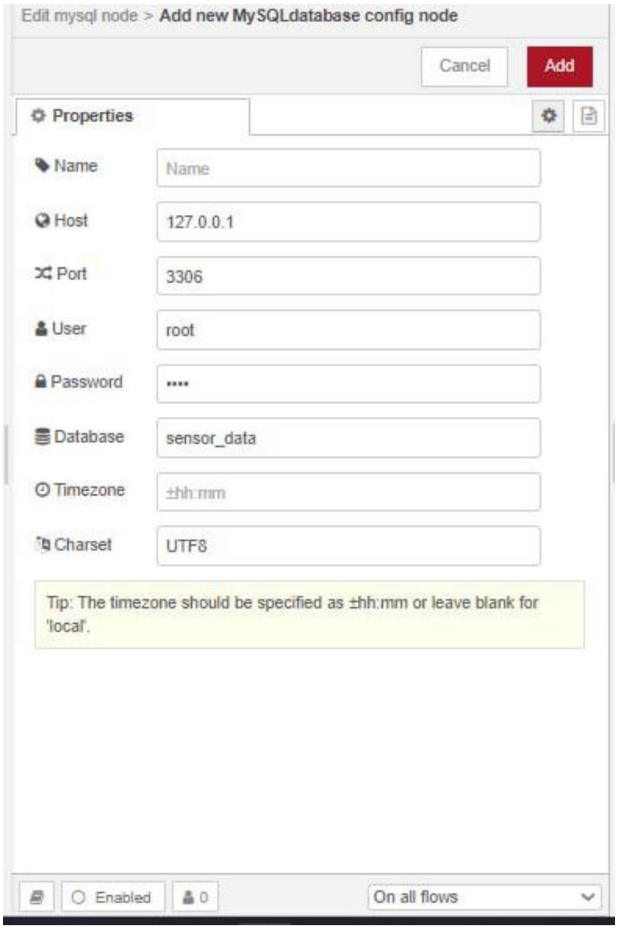

Рисунок 2.10 – налаштування з'єднання з MySQL

Система буде працювати автоматично навіть після закриття веб додатку, головною умовою буде лише запушений процес Node-RED.

Висновки до розділу

Розділ описує Node-RED як потужний інструмент для створення розумних інтернет-рішень. Використовуючи графічний інтерфейс, користувачі можуть легко створювати потужні додатки, які взаємодіють з різними компонентами. Це ідеальний інструмент для навчання, розробки та прототипування рішень IoT, забезпечуючи швидку розробку через інтуїтивний інтерфейс. Установка Node-RED є простою і доступною, не вимагаючи реєстрації або вартісних оплат. Крім того, розділ розглядає налаштування IoT мережі з використанням Node-RED, вказуючи на кроки встановлення та налаштування компонентів, таких як MySQL. Наприкінці розділу надається опис процесу створення та налаштування компонентів мережі, зокрема вузлів та функцій для генерації даних та їх запису до бази даних. Розділ зробив обґрунтовану демонстрацію можливостей Node-RED як інструменту для створення розумних інтернет-рішень, надаючи огляд процесу установки, налаштування та роботи з ним.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАСОБІВ МОНІТОРИНГУ ЕЛЕМЕНТАМИ IOT

3.1 Опис архітектури та інструментів розробки

Для розробки було обрано архітектуру клієнт-сервер, що забезпечує легку масштабованість системи. Клієнт-серверна архітектура дозволяє оновлювати сервер або додавати нові функції без необхідності зміни клієнтської частини. Сервер відповідає за зберігання та обробку даних, що спрощує управління даними, забезпечує безпеку та визначає політику доступу. Його можна розмістити у захищеному середовищі з відповідними заходами безпеки, тоді як клієнтські програми можуть працювати на менш захищених пристроях. Сервер обробляє складні обчислювальні завдання та взаємодіє з базою даних, що відсутнє в клієнтських пристроях.

Додаток розроблений мовою програмування Java версії 8 з використанням фреймворку Spring Boot для спрощення розробки. Дані зберігаються за допомогою системи керування базами даних (СКБД) MySQL. Для роботи з даними використовується ORM-фреймворк Hibernate, який дозволяє виконувати запити до бази даних за допомогою мови запитів HQL, спрощуючи процес розробки. Веб-інтерфейс розроблений за допомогою шаблонізатора Freemarker, що спрощує створення динамічних сторінок. Для розробки використано середовище IntelliJ IDEA від компанії JetBrains, що забезпечує зручну та продуктивну роботу завдяки потужним плагінам та інструментам.

3.2 Архітектура серверного додатку

Додаток розроблено за архітектурою MVC (model, view, controller). Така аріхтектура дозволяє слабо зв’язувати між собою модулі, що дає можливість легкої підтримки та заміни модулів (Рис.3.1). Детальніше про модулі:

1. Model, це робота з базою даних, обробка та валідація даних;

2. View, частина яка приймає вхідні дані, та надсилає оброблені;

3. Controller, основна бізнес-логіка, обробка, та виконання операції з даними.

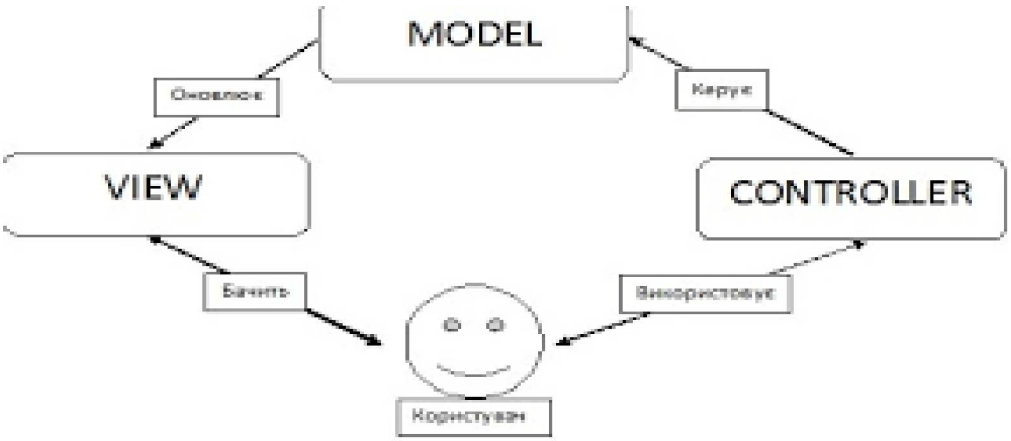

Рисунок 3.1 Архітектура MVC

Модель (Model) представляє дані та бізнес-логіку програми. Вона відповідає за доступ до даних, їх обробку та маніпулювання. Модель не залежить від користувацького інтерфейсу, тому вона може бути повторно використана у різних частинах додатку. Вид (View) в свою чергу відповідає за відображення даних користувачеві. Він представляє собою користувацький інтерфейс, який може бути веб-сторінкою, графічним елементом і т. д. Вид не має права вносити зміни в дані, він просто відображає їх у відповідному форматі. Контролер (Controller) відповідає за обробку вхідних даних від користувача, таких як кліки миші, натискання кнопок і т. д. Він інтерпретує ці дії та взаємодіє з моделлю, змінюючи її стан за необхідності. Контролер також оновлює вигляд, щоб відобразити зміни в даних користувачеві[16].

3.2.1 Опис мови програмування Java та фреймворку Spring Boot

Java — це об'єктно-орієнтована мова програмування, яка була випущена у 1995 році. Однією з головних особливостей Java є її схожість з мовою програмування C у синтаксисі. Проте, вона відрізняється від C завдяки використанню великої кількості об'єктно-орієнтованих концепцій[17].

Найважливішою рисою Java є Java Virtual Machine (JVM), яка є віртуальною машиною. JVM дозволяє розробникам писати код один раз, а потім виконувати його на різних платформах без необхідності перекомпіляції. Це значно полегшує розробку програмного забезпечення та знижує витрати на неї.

Одним з ключових компонентів JVM є байт-код Java. Вихідний код програми на Java компілюється в байт-код, який потім виконується JVM. Байт-код Java є переносним та може запускатися на будь-якому пристрої, який підтримує JVM. Це дозволяє програмістам писати програми, які працюють на різних операційних системах та пристроях, не залежно від їх архітектури. Основні причини обрання Java, як мови для розробки даного додатку:

1. Широке використання. Java є однією з найпопулярніших мов програмування з великою кількістю бібліотек та інструментів.
2. Сумісність. Java добре підходить для створення веб-додатків, особливо у корпоративному середовищі.
3. Покращена продуктивність. Java 8 включає лямбда-вирази та Stream API, що покращує продуктивність та читабельність коду.

Spring Boot - це потужний фреймворк для розробки додатків на мові Java, який значно спрощує процес створення програм. Він надає реалізацію багатьох функцій, які в традиційній розробці на Java доводиться реалізовувати самостійно. Одним з ключових переваг Spring Boot є те, що він дозволяє легко створювати додатки, які можна запустити з мінімальними зусиллями[18]. Він поєднує в собі Spring Framework та інші сторонні бібліотеки, що дозволяє розробникам швидко та ефективно створювати програми без необхідності вручну конфігурувати кожен компонент. Більшість Spring Boot додатків

вимагають лише невеликої конфігурації, що робить процес розробки ще більш приємним та продуктивним. Основні причини обрання Spring Boot:

1. Швидка розробка. Spring Boot забезпечує вбудовані функції для швидкого розгортання додатків з мінімальною конфігурацією.
2. Модульність. Можливість легко додавати та видаляти модулі та бібліотеки.
3. Спільнота. Велика спільнота розробників, документація та підтримка.

Також для Spring Boot потрібно підключити залежність для роботи з веб-інтерфейсами. Завдяки Maven, підключення цілої бібліотеки реалізується у декілька строчок коду у файлі pom.xml (Рис.3.2).

```
<!-- Spring Boot Starter Web for building web applications -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- Spring Boot Starter Data JPA for working with databases -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Рисунок 3.2 - Підключення залежностей для web розробки для Spring Boot

3.1.3 MySQL та Hibernate

MySQL - це система управління реляційними базами даних (RDBMS), яка широко використовується для різних баз даних. Вона є однією з найпопулярніших баз даних у світі, і її часто використовують для веб-додатків, онлайн-магазинів, аналітики даних та багато іншого. Переваги MySQL:

1. Надійність. MySQL є одним з найпопулярніших СУБД з високою надійністю та продуктивністю.

2. Широка підтримка. Підтримка широкого спектру інструментів для управління та моніторингу баз даних.
3. Масштабованість. Підтримка масштабованих рішень для великих обсягів даних.

Hibernate є засобом перетворення класів у Java на таблиці у СКБД, та надає зручні інтерфейси для керування даними та базою. Метою Hibernate є скорочення часу розробки завдяки вже реалізованим в ньому запитам. Він автоматично контролює зв’язок між класами та відповідними таблицями, забезпечуючи простий спосіб їх з'єднання. Hibernate також має свою мову запитів, HQL, яка є аналогом SQL, але спрощена для роботи з об'єктами замість таблиць.

3.1.4 Freemarker

Freemarker — це простий шаблонізатор, що дозволяє розробникам створювати гнучкі та зручні інтерфейси користувача для веб-додатків без необхідності глибокого розуміння мов програмування, таких як JavaScript або TypeScript. Основною перевагою цього інструмента є макроси, які дають змогу значно зменшити обсяг коду завдяки можливості повторного використання вже створених шаблонів.

Freemarker дозволяє розробникам створювати шаблони HTML-сторінок з можливістю вбудовування змінних, логічних конструкцій та інших елементів динамічного контенту. Він легко інтегрується з різними веб-фреймворками та додатками, що робить його популярним вибором для швидкої розробки веб-інтерфейсів.

Завдяки простому та зрозумілому синтаксису, Freemarker дозволяє розробникам швидко створювати та модифікувати шаблони з мінімальними

зусиллями. Крім того, він підтримує розширення через власні директиви та функції, що дозволяє розширити його можливості за потреби.

3.1.5 Thymeleaf

Thymeleaf має потужну інтеграцію зі Spring Boot, підтримуючи Spring MVC та дозволяючи легко обробляти моделі даних і форми, що робить його ідеальним вибором для проектів на базі Spring Boot. Thymeleaf оптимізований для швидкої генерації HTML і добре масштабується для великих веб-додатків. Це особливо важливо для проектів, які потребують швидкого рендерингу сторінок та ефективного використання ресурсів.

Thymeleaf також підтримує створення власних розширень і діалектів, що дозволяє додавати нові можливості та адаптувати його під специфічні потреби проекту. Завдяки своїй популярності, Thymeleaf має велику і активну спільноту користувачів, що забезпечує наявність великої кількості прикладів, бібліотек і документації. Це значно полегшує вирішення проблем і покращує загальну якість розробки.

Вибір Thymeleaf для цього проєкту обумовлений його сумісністю з HTML, легкістю у використанні, потужною інтеграцією зі Spring Boot, високою продуктивністю, широкими можливостями розширення та великою спільнотою користувачів[19]. Ці переваги роблять Thymeleaf ідеальним вибором для створення фронтенду для проєкту моніторингу IoT, забезпечуючи зручність розробки та високу якість кінцевого продукту.

3.2 Реалізація серверної частини

3.2.1 Реалізація бізнес-логіки та моделей даних

У проекті реалізовано систему збору даних з різних типів сенсорів, таких як вологість, освітлення, температура та швидкість вітру. Створено 3 основні частини відповідно до архітектури:

1. Моделі даних. Наприклад, модель Humidity містить поля, які відображають значення вологості, час зчитування та унікальний ідентифікатор (Рис.3.3). Аналогічно, моделі Light, Temperature та WindSpeed мають відповідні поля для даних, які зчитуються з відповідних сенсорів.
2. Репозиторії. Для кожного типу сенсора створено відповідний репозиторій, наприклад HumidityRepository, який реалізує стандартні CRUD операції за допомогою JpaRepository (Рис.3.4).
3. Сервіси. Кожен тип сенсора має свій власний сервіс, наприклад HumidityService, який інкапсулює бізнес-логіку. Сервіси надають методи для отримання існуючих даних (getAllHumidities) (Рис.3.3).

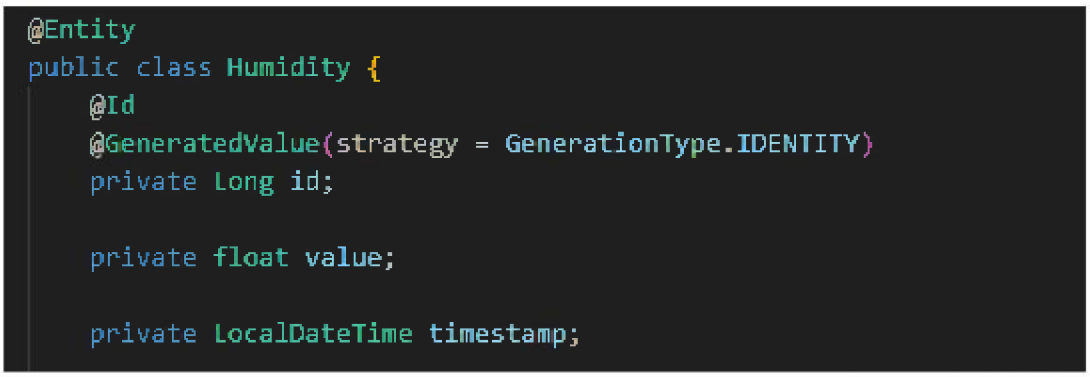

Рисунок 3.3- Модель Humidity

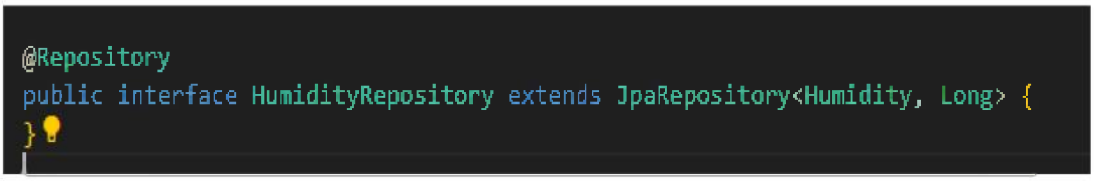

Рисунок 3.4 – Репозиторій HumidityRepository

```java
@Service
public class HumidityService {

    @Autowired
    private HumidityRepository humidityRepository;

    public List<Humidity> getAllHumidities() {
        return humidityRepository.findAll();
    }
```

Рисунок 3.5 – Сервіс HumidityService

Ця архітектура дозволяє легко додавати нові типи сенсорів, не змінюючи основну структуру додатку. Наприклад, якщо потрібно додати сенсор для вимірювання рівня CO2 в повітрі, достатньо створити нову модель, репозиторій та сервіс для цього типу сенсора, а основна логіка додатку залишиться незмінною.

3.2.2 Реалізація Бізнес-Логіки додатку

Бізнес-логіка додатку реалізована у вигляді контролерів (Рис.3.6), які викликають сервіси для виконання необхідних дій. Контролери для кожного типу сенсора забезпечують REST API для додавання нових показників і отримання існуючих даних. Для забезпечення коректності даних, що зберігаються, додано валідацію вхідних даних. Контролери використовують анотації Spring для обробки запитів:

1. GetMapping - для обробки GET-запитів.
2. PostMapping - для обробки POST-запитів.

Варто зазначити що також можливе додавання інших типів анотацій, однак розроблена система передбачає лише отримання даних.

```java
@RestController
@RequestMapping("/api/humidities")
public class HumidityController {

    @Autowired
    private HumidityService humidityService;

    @GetMapping
    public List<Humidity> getAllHumidities() {
        return humidityService.getAllHumidities();
    }

    @PostMapping
    public Humidity createHumidity(@RequestBody Humidity humidity) {
        return humidityService.createHumidity(humidity);
    }
}
```

Рисунок 3.6 – HumidityController

3.3 Реалізація інтерфейсу користувача

Інтерфейс користувача реалізований з використанням Thymeleaf. Створено окремі HTML-шаблони для кожного типу сенсора, які використовують Thymeleaf для динамічного рендерингу даних. Контролери обробляють HTTP-запити і повертають відповідні шаблони Thymeleaf з даними. Це дозволяє користувачам переглядати останні показники, показники за день.

3.3.1 Налаштування системи

Налаштування системи включає конфігурацію Spring Boot додатку, налаштування підключення до бази даних MySQL та налаштування Thymeleaf для рендерингу сторінок. Файл конфігурації application.properties містить налаштування для підключення до бази даних, такі як URL, ім'я користувача та пароль (Рис.3.6). Також створено скрипт SQL для створення бази даних і таблиць, що містять дані сенсорів. Цей скрипт включає команди для створення бази даних і таблиць для кожного типу сенсора.

3.3.2 Налаштування серверної частини

Налаштування доступу до бази даних здійснюється за допомогою файлу application.properties (Рис.3.7). Трьома основними пунктами налаштування є:

1. spring.datasource.url - шлях до бази даних.
2. spring.datasource.username - ім’я користувача бази даних.
3. spring.datasource.password - пароль для підключення до бази даних.

```
spring.application.name=SmartHomeowner

# MySQL Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/sensor_data
spring.datasource.username=root
spring.datasource.password=1111
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Рисунок 3.7 - Налаштування application.properties

3.4 Демонстрація роботи

3.4.1 Встановлення та запуск

Для встановлення системи потрібно, щоб на машині була встановлена JVM не нижче 8-мої версії. Для запуску додатка потрібна встановлена база даних з створеним користувачем, запущенний MySQl , активний сеанс у Node-RED, у разі якщо необхідно отримувати поточні данні. Після чого потрібно скопіювати файл SmartHome.jar та запустити командою java -jar.

3.4.2 Демонстрація роботи

Системою моніторингу можна користуватися як з персональних комп'ютерів та ноутбуків, так і з мобільних пристроїв. Для початку роботи

потрібно перейти за адресою http://localhost:8080 в браузері, так як 8080 – це порт на якому працює Spring, після чого відкриється головне вікно системи.

За основною адресою знаходиться домашня сторінку, також можна перейти в детальніший огляд пристроїв натиснувши на відповідне поле. При переході на один з пристроїв відкривається вікно з детальною інформацією про пристрій: його назва, поточний показники, та (якщо це, наприклад, датчик температури) графік коливань теператури за останню добу (Рис.3.8).

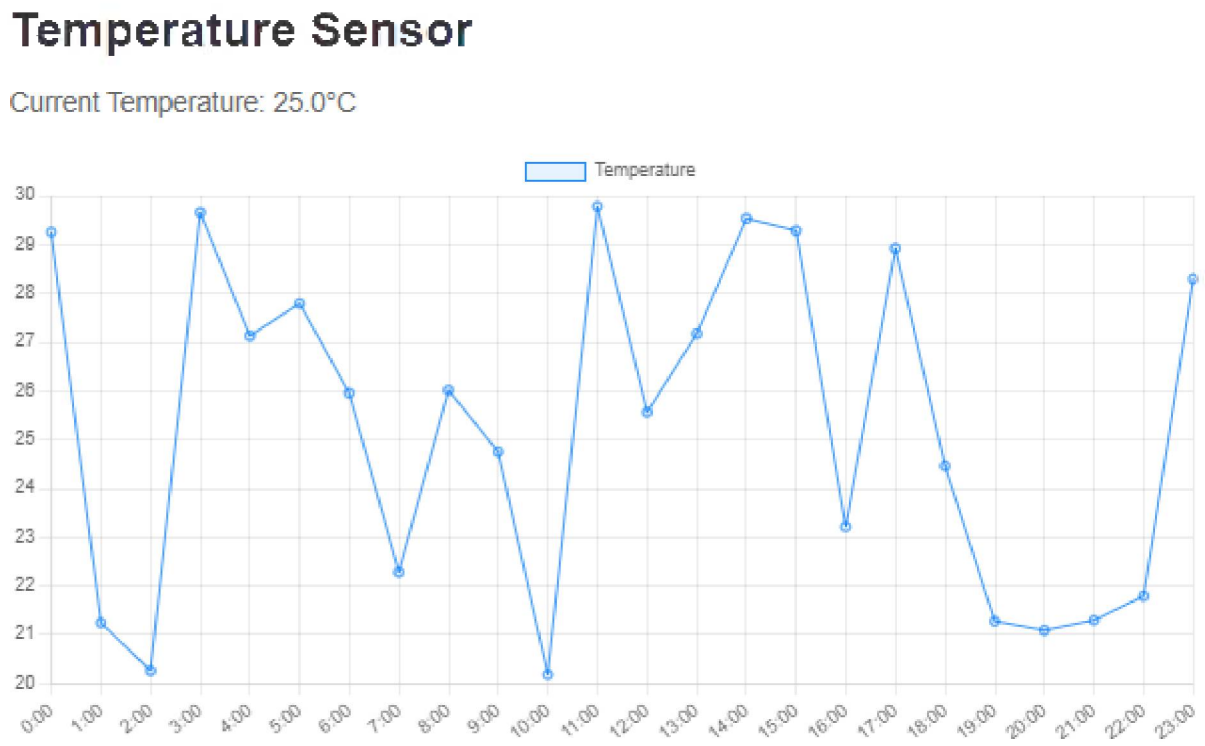

Рисунок 3.8 – Інформація температурного датчика

Висновки до розділу

У розділі надано огляд архітектури та реалізації серверної частини інтегрованої системи моніторингу різних параметрів середовища. Ця система призначена для збору та аналізу даних з різних типів сенсорів, таких як вологість, освітлення, температура та швидкість вітру. Описано структуру

системи, включаючи моделі даних для різних типів сенсорів, репозиторії для зберігання даних та сервіси для інкапсуляції бізнес-логіки. Ця архітектура дозволяє легко додавати нові типи сенсорів, не змінюючи основну структуру додатку. Описано реалізацію бізнес-логіки у вигляді контролерів, які викликають сервіси для виконання необхідних дій. Контролери забезпечують REST API для взаємодії з додатком, а також використовують валідацію вхідних даних для забезпечення коректності інформації.

Пояснено, як реалізований інтерфейс користувача з використанням Thymeleaf, що дозволяє динамічно рендерити дані на сторінках. Описано використання HTML-шаблонів для відображення даних різних типів сенсорів. Наведено інформацію про налаштування системи, включаючи конфігурацію бази даних та Thymeleaf. Описано структуру конфігураційних файлів та встановлення додатку. Описано процес встановлення та запуску системи, а також можливості користування нею з різних пристроїв. Зазначено, що система дозволяє переглядати останні показники, статистику за різними періодами та графіки зміни значень сенсорів.

Цей розділ надає загальний огляд ключових аспектів архітектури та реалізації серверної частини системи моніторингу, а також процесу встановлення та користування нею.

ВИСНОВКИ

Тема роботи була обрана через те, що в сучасному світі, з урахуванням швидкого розвитку технологій та великої кількості інформації, є необхідність у автоматизації процесів, пов'язаних з її обробкою. Існує практична потреба в методах контролю та моніторингу за допомогою різних пристроїв. Для зручності роботи з такими пристроями мають існувати інтерфейси моніторингу, що є інформаційними моделями представлення даних. Це може бути застосовано у багатьох сферах діяльності, наприклад, у побуті для моніторингу різних показників приміщення тощо.

Інтернет речей (IoT) є основною концепцією для оцифровування даних, де люди мають змогу встановити контакт з пристроями через комунікаційні мережі та отримати інформацію про стан навколишнього середовища. На даний час використання цієї технології вже призвело до збільшення обсягу даних. Інтернет речей вже став тим кроком, що призвів до оцифровування суспільства, де люди і предмети пов'язані між собою. Розробка таких програм з концепцією Інтернету речей є актуальною та конкурентоспроможною.

Моніторинг за приладами є важливою складовою, що використовується у поточному часі для безпеки та комфорту користувачів. Розвиток Інтернету речей дав поштовх для створення і розвитку повсякденних розумних об'єктів. Такі системи надають великий спектр можливостей, зокрема, зниження витрат на експлуатацію та обслуговування завдяки моніторингу, діагностиці і іншим можливостям у віддаленому форматі. Зручність і безпека, які також надаються з можливістю віддаленого контролю, є додатковими перевагами.

Інтернет речей змінює спосіб розвитку промислово-споживчого ринку. Робототехнічні пристрої, дрони, автономні транспортні засоби, блокчейн, розширена та віртуальна реальність, цифрові помічники та машинне навчання (штучний інтелект) – це технології, які вже призвели до нового етапу розробки

додатків. Тому ідеї автоматизації та віддалених сервісів будуть ще довго актуальними.

У даній роботі було досліджено існуючі інформаційні системи та засоби моніторингу і управління за концепцією IoT, для розуміння необхідних елементів для роботи таких систем. Проведені дослідження надають змогу зрозуміти, які програмні засоби необхідні для побудови системи моніторингу елементів IoT, і як це зробити з мінімальними фінансовими витратами.

Дослідження є ефективним, бо наводить конкретні програмні засоби і кроки для побудови програмних і апаратних забезпечень, що дозволяє створити власну повну мережу Інтернету речей. У роботі було проаналізовано, які протоколи та типи мереж існують, а також складові таких систем. Проаналізовано, які задачі у моніторингу і навіщо необхідне його застосування. Це дозволяє обрати необхідні технології, на яких базується система.

У роботі запропонована система моніторингу, побудована на основі Node-RED. Система моделює роботу мережі, подібну до реальної компактної мережі. Вона може контролювати стан сенсорів завдяки Інтернету. Користувач може входити до системи та взаємодіяти з пристроями в реальному часі. Система має можливість для емуляції роботи будь-яких сенсорів з відповідними інтерфейсами.

Node-RED надає можливість емулювати роботу реальних датчиків, тому ідеально підходить для роботи. Адже Інтернет речей - це не тільки велика кількість різних приладів і датчиків, об'єднаних між собою каналами, а й тісна інтеграція реального та віртуального світів, у середовищі якого спілкування здійснюється між людьми і пристроями.

У даному дослідженні проведено аналіз, що в підсумку дозволяє зрозуміти, яка архітектура застосунків є необхідною. Найбільше підходить архітектура клієнт-сервер. Спираючись на вибрану архітектуру додатку, який забезпечує керування системою, обрано мову Java, що має об’єктно-орієнтований

синтаксис, безпечну роботу з пам’яттю, незалежність від платформи та архітектури та високу продуктивність. Серверний додаток створено за архітектурним шаблоном MVC, а клієнтський - за монолітною архітектурою і запропоновані засоби, за допомогою яких є можливість розробки користувацького інтерфейсу.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Лаудон, К. К., Лаудон, Д. П. Менеджмент інформаційних систем: управління цифровою фірмою. 16-те вид. Лондон: Pearson, 2020. С. 9-12.

2. Турбан, Е., Волоніно, Л. Інформаційні технології для управління: розвиток стійкого, прибуткового бізнесу. 10-те вид. Нью-Йорк: Wiley, 2019. С. 227-230.

3. Clavius Solutions. Supply Chain Management Solutions. [Електронний ресурс]. – Режим доступу: https://www.claviussolutions.com/scm/ (дата звернення: 10.02.2024).

4. Турбан, Е., Аронсон, Д. Є., Ліанг, Т.-П. Системи підтримки прийняття рішень та інтелектуальні системи. 8-те вид. Лондон: Pearson, 2019. С. 3-7.

5. Alavi, M., Leidner, D. E. Knowledge Management Systems: Issues, Challenges, and Benefits. Communications of the AIS, 2001, 7(1), С. 1-37.

6. McInerney, C. R., Day, R. L. Knowledge Management Systems: Information and Communication Technologies for Knowledge Management. 4-те вид. Нью-Йорк: Routledge, 2019. С. 93-98.)

7. Шалабодін, С. Що таке CMS? [Електронний ресурс]. – Режим доступу: https://shalabodin.com/shcho-take-cms/ (дата звернення: 27.02.2024).

8. Atzori, L., Iera, A., Morabito, G. The Internet of Things: A Survey. Computer Networks, 2010, 54(15), С. 2787-2805.

9. I ITU-T. ITU Internet Reports 2005: The Internet of Things. ITU Telecommunication Standardization Sector, International Telecommunication Union, 2005. С. 5-10.

10. Гільфанд, Б. П. Проектування вбудованих систем управління. Київ: Наукова думка, 2004. С. 27-35.

11. Hui, T., Yu, R., Song, H. Zigbee based intelligent hepta-copter with infrared and temperature sensing capabilities. IEEE Transactions on Industrial Informatics, 2013, 9(1), С. 289-297.

12. Mukherjee, S., Misra, P., Mondal, S. Architecture of IoT gateway for smart city: A review. В: 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU). Нью-Йорк: IEEE, 2019. С. 1-6

13. Li, W., Li, L., Zhang, Y., Yang, L. T. IoT-based smart rehabilitation system. IEEE Transactions on Industrial Informatics, 2015, 11(2), С. 403-412.

14. Node-RED. Офіційний сайт. [Електронний ресурс]. – Режим доступу: https://nodered.org/ (дата звернення: 21.03.2024).

15. MySQL. MySQL :: MySQL Installer. [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/mysql-installation-excerpt/5.7/en/installing.html (дата звернення: 22.03.2024).

16. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Лондон: Addison-Wesley, 1995. ISBN 0-201-63361-2.

17. Oracle. (2024). Java Language and Virtual Machine Specifications. [Електронний ресурс]. Режим доступу: https://docs.oracle.com/en/java/javase/index.html (дата звернення: 27.03.2024).

18. Wall, M., Hoye, D., Overdick, S. Spring Boot: Up and Running: Building Cloud Native Java and Kotlin Applications. Сан-Франциско: O'Reilly Media, 2016. ISBN: 978-1491938053.

19. Legler, A. Learning Thymeleaf: A practical guide to building feature-rich web pages with Thymeleaf. Бірмінгем: Packt Publishing, 2016. ISBN: 978-1785882744.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
галузь знань:**15 – Автоматизація та приладобудування**
спеціальність:**151 – Автоматизація та комп'ютерно-інтегровані технології**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної
та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.
«21» грудня 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Окрутний Роман Ярославович

(прізвища, ім'я, по батькові студента)

1. Тема роботи «***Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей***»

керівник роботи Агеєв Дмитро Володимирович, д.т.н., професор.
(прізвища, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» травня 2024року № 4101-5/909

2.Строк подання студентом роботи 31 травня 2024року

3. Перелік питань, які потрібно розробити

1) Визначення параметрів моніторингу середовища.
2) Розробка алгоритмів емуляції роботи сенсорів та пристроїв для збору даних.
3) Розробка алгоритмів збору даних.
4) Проектування інтерфейсу користувача для візуалізації та аналізу даних моніторингу.
5) Розробка системи зберігання та обробки даних.
6) Тестування роботи інформаційної системи.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Пошук та аналіз наукової літератури	21.12.2023 - 25.01.2024
2	Аналіз існуючих моделей моніторингу середовища з використанням технологій інтернету речей	19.12.2023 - 2.01.2024
3	Аналіз принципів роботи моделей моніторингу середовища	19.12.2023 - 2.01.2024
4	Проектування архітектури системи	2.01.2024 - 2.02.2024
5	Розробка алгоритмів емуляції роботи сенсорів та пристроїв для збору даних	12.01.2024 - 2.02.2024
6	Реалізація прототипу інформаційної системи	2.01.2024 - 2.02.2024
7	Тестування роботи інформаційної системи	3.02.2024 - 30.03.2024
8	Підготовка тексту кваліфікаційної роботи	3.03.2024 - 30.04.2024
9	Оформлення пояснювальної записки.	31.03.2024 - 27.05.2024

5. Дата видачі завдання 21.12.2023

Студент Р. Я. Окрутний _____
ініціали, прізвище підпис

Керівник роботи Д. В. Агеєв _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання на розробку програмного виробу «Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного вироби – Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей. 1.2. Галузь застосування – моніторинг середовища.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології. 2.2. Завдання на дипломну роботу бакалавра, затверджено наказом ХНУ імені В. Н. Каразіна № 4101-5/909 від 03.05.2024 р. (представить як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3. Призначення розробки	3.1. Мета розробки програмного виробу – розробка IoT мережі у Node-RED та Java-додатка за архітектурою MVC для збору та обробки даних з цієї мережі. 3.2. Призначення програмного виробу – моніторинг середовища. 3.3. Початкові дані для розробки: 1) Визначення параметрів моніторингу середовища. 2) Розробка алгоритмів емуляції роботи сенсорів та пристроїв для збору даних. 3) Розробка алгоритмів збору даних. 4) Проектування інтерфейсу користувача для візуалізації та аналізу даних моніторингу. 5) Розробка системи зберігання та обробки даних. 6) Тестування роботи інформаційної системи.
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: 1) представляти з себе програмну реалізацію. 2) емулювати роботу сенсорів та пристроїв для збору даних. 3) надавати поточну емульовану інформацію. 4.2. Вимоги до надійності: Можна моніторити поточну інформацію про стан системи. 4.3. Вимоги до умов експлуатації офісні приміщення. 4.4. Вимоги до складу і параметрів технічних засобів – Персональний комп'ютер у повній комплектації (ноутбук)

<table>
<tr><td></td><td>4.5. Вимоги до інформаційної та програмної сумісності забезпечити сумісність з усіма обчислювальними засобами.
4.6. Вимоги до маркування та упаковки відсутні.
4.7. Вимоги до транспортування і зберігання відсутні.
Спеціальні вимоги не пред'являються.</td></tr>
<tr><td>5. Вимоги до програмної документації.</td><td>Програмною документацією до виробу « Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей» вважати:
1) Справжнє Технічне завдання на розробку програмного вироби (представити у вигляді Додатку Б до пояснювальної записки до дипломної роботи).
2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до дипломної роботи).
3) Опис програмного виробу (представити в розділі 3 пояснювальної записки до дипломної роботи).
4) Текст програми (представити в Додатку В до пояснювальної записки до дипломної роботи).</td></tr>
<tr><td>6. Техніко-економічні показники</td><td>В даному розділі можуть бути представлені:

1) Вимоги до моніторингу показників не потрібні.</td></tr>
<tr><td>7. Стадії і етапи розробки</td><td>
<table>
<tr><th>№ з/п</th><th>Назви етапів роботи</th><th>Терміни виконання етапів роботи</th></tr>
<tr><td>1</td><td>Пошук та аналіз наукової літератури</td><td>21.12.2023 - 25.01.2024</td></tr>
<tr><td>2</td><td>Аналіз існуючих моделей моніторингу середовища з використанням технологій інтернету речей</td><td>19.12.2023 - 2.01.2024</td></tr>
<tr><td>3</td><td>Аналіз принципів роботи моделей моніторингу середовища</td><td>19.12.2023 - 2.01.2024</td></tr>
<tr><td>4</td><td>Проектування архітектури системи</td><td>2.01.2024 - 2.02.2024</td></tr>
<tr><td>5</td><td>Розробка алгоритмів емуляції роботи сенсорів та пристроїв для збору даних</td><td>12.01.2024 - 2.02.2024</td></tr>
<tr><td>6</td><td>Реалізація прототипу інформаційної системи</td><td>2.01.2024 - 2.02.2024</td></tr>
<tr><td>7</td><td>Тестування роботи інформаційної системи</td><td>3.02.2024 - 30.03.2024</td></tr>
<tr><td>8</td><td>Підготовка тексту кваліфікаційної роботи</td><td>3.03.2024 - 30.04.2024</td></tr>
<tr><td>9</td><td>Оформлення пояснювальної записки.</td><td>31.03.2024 - 27.05.2024</td></tr>
</table>
</td></tr>
</table>

8. Порядок контролю і приймання	1) Перевірку ходу розробки програмного виробу керівнику робіт виконувати раз в 2 тижні. 2) Випробування програмного продукту провести відповідно до програми та методики випробувань на базі комп'ютерного класу. 3) Захист розробленої моделі провести на засіданні Атестаційної комісії. 4) Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику на CD R компакт-диску.

Виконавець

студент групи КУ-41

Окрутний Р.Я.

Замовник

д.т.н, професор кафедри ТПС

АГЕЄВ Д. В.

Додаток В

Програма і методика випробувань програмного виробу

«Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей»

1 Об'єкт випробувань

1.1 «Модель інформаційної системи моніторінгу середовища з використанням технологій інтернету речей».

1.2 Область його застосування: забезпечення сталого розвитку, ефективного використання ресурсів для поліпшення якості життя.

2. Мета випробувань

Емуляція та надання поточної інформації про стан середовища.

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї Програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4.1. Вимоги до функціональних характеристик:

1) представляти з себе програмну реалізацію.
2) надавати інформацію моніторингу середовища .

3) емулювати роботу сенсорів.

4.2. Вимоги до надійності:

Можна отримувати поточну інформацію з індикаторів

4.3. Вимоги до умов експлуатації офісні приміщення відсутні.

4.4. Вимоги до складу і параметрів технічних засобів Персональний комп'ютер у повній комплектації (ноутбук)

4.5. Вимоги до інформаційної та програмної сумісності забезпечити сумісність з усіма обчислювальними засобами.

4.6. Вимоги до маркування та упаковки відсутні.

4.7. Вимоги до транспортування і зберігання відсутні.

4.8. Спеціальні вимоги не пред'являються.

5. Вимоги до програмної документації

Склад програмної документації, що подається на випробування, включає:

Технічне завдання на розробку програмного вироби (представлено в Додатку Б до пояснювальної записки до дипломної роботи).

Ця Програма і методика випробувань розробленого програмного виробу (представлена в Додатку В до пояснювальної записки до дипломної роботи).

Опис програмного виробу (представлено в розділі 3 пояснювальної записки до дипломної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Випробування проводяться на технічних засобах, як ноутбук.

Випробування проводяться з використанням програмних засобів, таких як MySQL, IntelliJ IDEA, Node-RED, Google Chrome.

6.2 Порядок проведення випробувань

6.2.1. Перевірка програмної документації.

Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в технічному завданні документації.

6.2.2. Перевірка якості програмної документації.

Перевірку здійснювати за критерієм відповідності вимогам єдиної системи програмної документації (ЄСПД).

6.2.3. Перевірка виконання програми.

Тест 1: Перевірка створення працездатного алгоритму емуляції роботи датчиків. По виду рисунку робиться висновок про працездатність моделі (рисунок 6.1).

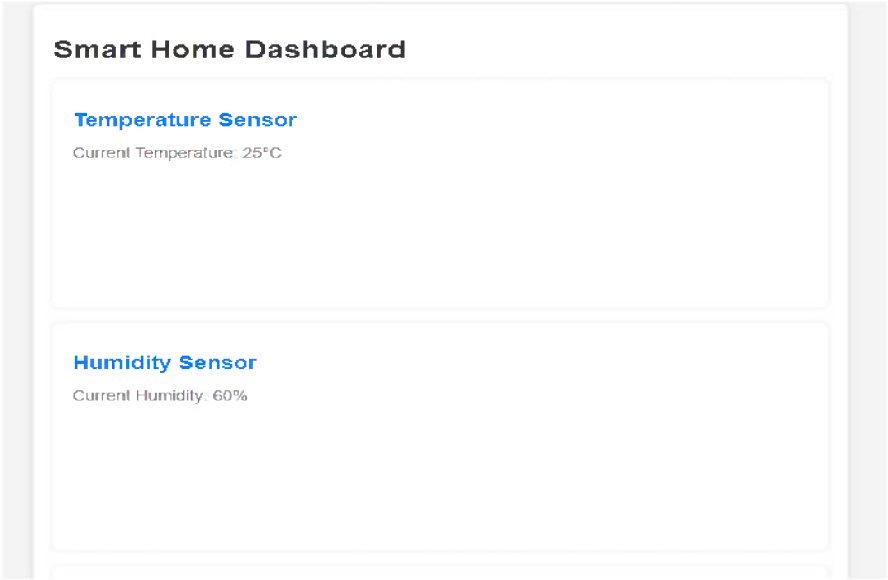

Рисунок В.1 – Перевірка створення працездатного веб-застосунку. По виду рисунку робиться висновок про працездатність моделі.

Висновок: Перевірка створення працездатного веб-застосунку пройшла успішно.

Тест 2: Перевірка створення працездатного алгоритму емуляції роботи датчиків (рисунок В.2 , В.3).

```
msg.topic = "INSERT INTO temperature (value, timestamp) VALUES (" + (Math.random() * 80).toFixed(2) + ", NOW())";
return msg;
```

Рисунок В.2 – код JavaScript-програми

	id	value	timestamp
▶	1	26.63	2024-05-29 23:29:16
	2	66.97	2024-05-30 23:30:08
	3	7	2024-05-30 23:30:40
	4	45.68	2024-05-30 23:31:40
	5	63.29	2024-05-30 23:32:40

Рисунок В.3 – Таблиця temperature з бази даних

Висновок: Алгоритм працює відповідно до поставлених задач . Перевірка створення алгоритму емуляції роботи датчиків пройшла успішно.

Тест 3: Перевірка коректності виводу поточних даних(рисунок В.4).

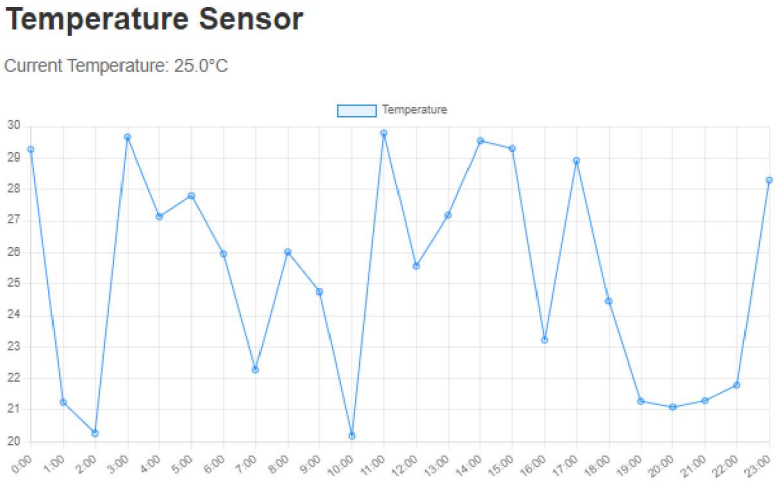

Рисунок В.4 – Перевірка коректності виводу поточних даних

Висновок: Перевірка коректності виводу поточних даних пройшла успішно.

Висновки: при вдалому виконанні всіх 3 тестів випробування розробленого додатку вважаються успішними.

Виконавець	Замовник
студент групи КУ-41	д.т.н, професор кафедри ТПС
Окрутний Р.Я.	АГЕЄВ Д. В.
_____	_____