ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУІМЕНІ
ТАРАСА ШЕВЧЕНКА

Кафедра інформаційних систем та технологій

О.В. КРАВЧЕНКО
О.М. СІПКО

АРХІТЕКТУРА IoT ЕКОСИСТЕМ

Методичні рекомендації до виконання лабораторних робіт для здобувачів освітнього ступеня «магістр» спеціальності 126 – Інформаційні системи та технології освітньої програми «Програмні технології Інтернет речей»

Київ 2025

Рецензенти:

д.т.н., професор кафедри технологій управління факультету інформаційних технологій Київського національного університету імені Тараса Шевченка Хлевна Юлія Леонідівна

к.т.н., доцент, зав. каф. статистики та прикладної математики факультету інформаційних технологій і систем Черкаського державного технологічного університету Карапетян Анаіт Радиківна

КРАВЧЕНКО Ольга Віталіївна, к.т.н., доцент
СІПКО Олена Миколаївна, к.т.н.

Архітектура IoT екосистем [Електронний ресурс]: Методичні рекомендації до виконання лабораторних робіт для здобувачів освітнього ступеня «магістр» спеціальності 126 «Інформаційні системи та технології» освітня програма «Програмні технології інтернет речей» / Укл. Кравченко О.В., Сіпко О.М. – К.: Київський національний університет імені Тараса Шевченка, 2025. – 25 с.

Лабораторні роботи з дисципліни ***«Архітектура IoT екосистем»****.*

Методичні рекомендації містять 5 лабораторних робіт, призначених для виконання в комп'ютерному класі або із застосуванням програмних симуляторів. Метою лабораторних робіт є засвоєння та поглиблення теоретичних знань, набуття практичних навичок і вмінь щодо розробки та аналізу архітектури IoT-екосистем.

Публікується в авторській редакції.

Електронна версія цього видання опублікована на сайті кафедри інформаційних систем і технологій за адресою:

_____

(дата публікації «___» _____ 20___ р.)

Зміст

ВСТУП

Розвиток технологій Інтернету речей (IoT) суттєво трансформує сучасні цифрові екосистеми, відкриваючи нові можливості для автоматизації, моніторингу та оптимізації процесів у промисловості, міській інфраструктурі, агросекторі, медицині та багатьох інших сферах. Архітектура IoT-екосистем охоплює широке коло концепцій, методів і технологій, які забезпечують побудову надійних, масштабованих і безпечних рішень, що відповідають вимогам сучасної інженерії.

Ці методичні рекомендації призначені для студентів, які вивчають дисципліну «Архітектура IoT-екосистем». Вони містять комплекс лабораторних робіт, спрямованих на формування ключових компетентностей та практичних вмінь, необхідних для розробки та впровадження IoT-рішень.

Мета методичних рекомендацій – сприяти набуттю знань і практичних навичок, необхідних для створення ефективних IoT-екосистем з урахуванням вимог до безпеки, надійності та масштабованості.

Основні завдання:

- Ознайомити студентів із ключовими поняттями та компонентами архітектури IoT-систем.
- Навчити застосовувати локальні обчислювальні мережі, агрегатори та маршрутизатори для ефективної передачі даних від сенсорів та розумних пристроїв у межах IoT-екосистем.
- Розвинути вміння забезпечувати безпечну передачу даних у промислових IoT-мережах, зокрема за допомогою VPN-технологій, шифрування та інших засобів кібербезпеки.
- Формувати навички налагодження взаємодії програмного забезпечення IoT-систем з базами даних та створення веб-інтерфейсів для віддаленого моніторингу та керування.
- Навчити студентів проєктувати архітектуру IoT-систем, складати технічні завдання, обирати відповідне обладнання й програмні засоби.

- Засвоїти математичні методи аналізу даних, алгоритми маршрутизації, балансування навантаження, а також застосовувати теорію графів для побудови ефективної мережевої топології.
- Сприяти розвитку вмінь презентувати, документувати та обґрунтовувати результати досліджень і технічних рішень.

Методичні рекомендації містять:

- Теоретичні відомості до кожної теми;
- Послідовний опис етапів виконання лабораторних робіт;
- Практичні приклади реалізації завдань;
- Контрольні питання для самоперевірки;

Запропоновані лабораторні роботи охоплюють основні теми дисципліни «Архітектура IoT-екосистем», що формують базу для практичного застосування набутих знань і навичок. Зміст лабораторних завдань узгоджується з тематичним планом дисципліни та забезпечує цілісність навчального процесу:

- Лабораторна робота 1. Вибір предметної області та формулювання проблематики досліджень в архітектурі IoT-екосистем відповідає Темі 1 (Вступ до Інтернету речей) та частково Темі 3 (Екосистема IoT).
- Лабораторна робота 2. Розробка технічного завдання для IoT-дослідження охоплює Тему 2 (Промисловий Інтернет речей), Тему 3 (сенсори/виконавчі механізми) та Тему 4 (мережеве забезпечення в IoT-екосистемах).
- Лабораторна робота 3. Аналіз математичних методів і алгоритмів в IoT-дослідженні відповідає Темі 4 (опрацювання даних у мережах), Темі 5 (глобальна мережа, аналітика, безпека), Темі 6 (машинне навчання, туманні обчислення, загрози).
- Лабораторна робота 4. Аналіз та оптимізація маршрутизації даних в IoT-мережах безпосередньо стосується Теми 4 (локальні обчислювальні мережі, маршрутизатори, агрегатори), а також Теми 6 (маршрутизація, оптимізація, безпека мереж).

- Лабораторна робота 5. Підготовка підсумкової проєктної роботи та презентації дослідження IoT-екосистем інтегрує всі теми курсу, з особливим акцентом на застосуванні знань про архітектуру, мережеву взаємодію, безпеку, обробку даних і презентацію результатів.

Запропонований зміст методичних рекомендацій дозволяє студентам отримати цілісне уявлення про функціонування IoT-екосистем, їхнє проєктування та впровадження в реальні середовища, що сприяє розвитку професійних компетентностей відповідно до сучасних вимог галузі.

ЛАБОРАТОРНА РОБОТА 1
ВИБІР ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУЛЮВАННЯ ПРОБЛЕМАТИКИ ДОСЛІДЖЕНЬ В АРХІТЕКТУРІ IOT-ЕКОСИСТЕМ

Мета:

- Ознайомитися з основами IoT-екосистем та їх архітектурою.
- Оцінити та обґрунтувати вибір теми дослідження.
- Визначити основні проблеми в сфері IoT-екосистем та сформулювати задачі для їх вирішення.

Теоретичні відомості:

IoT-екосистема – це мережа взаємопов'язаних пристроїв, які передають дані через Інтернет без участі людини. Вона складається із сенсорів, що збирають інформацію, шлюзів і мережевих протоколів для передачі даних, хмарних сервісів для їх зберігання та аналізу, аналітичних систем на основі штучного інтелекту та користувацьких інтерфейсів для взаємодії.

IoT-системи поділяються на індустріальні, що автоматизують виробництво, розумні міста для оптимізації міської інфраструктури, медичні рішення для моніторингу здоров'я, транспортні системи для управління логістикою та сільське господарство для покращення врожайності.

Головні виклики IoT включають кібербезпеку, ефективне управління великою кількістю пристроїв, оптимізацію енергоспоживання та обробку великих обсягів даних у реальному часі.

Хід роботи:

Завдання 1. Обрання напрямку дослідження IoT:

- Ознайомитися з існуючими IoT-рішеннями та їх областями застосування.
- Вибрати одну з областей IoT, яка є актуальною для дослідження (перелік варіантів тем див. Додаток 1).

Завдання 2. Аналіз наукових публікацій та джерел:

- Виконати пошук літератури за обраною темою.
- Розглянути основні підходи та технології, що використовуються у дослідженнях IoT.

Завдання 3. Формулювання проблеми дослідження:

- Виявити основні труднощі та виклики у вибраній сфері IoT.
- Обґрунтувати необхідність вирішення цих проблем.

Завдання 4. Постановка задач дослідження:

- Сформулювати основні питання, на які потрібно знайти відповіді в рамках проєкту.

Завдання 5. Визначення мети дослідження:

- Сформулювати основну мету, яку необхідно досягти в межах проекту.

Завдання 6. Оцінка можливих підходів та план дослідження:

- Визначити потенційні методи вирішення обраної проблеми.
- Скласти план дослідження, що включає вибір технологій та методології аналізу.

Контрольні питання:

1. Що таке IoT-екосистема і які її основні компоненти?
2. Які архітектурні підходи використовуються для побудови IoT-систем?
3. Які основні виклики існують при розробці IoT-рішень?
4. Як оцінити актуальність вибраної теми дослідження?
5. Які основні етапи формулювання проблеми дослідження?
6. Як сформулювати мету та задачі дослідження в сфері IoT?
7. Які методи можуть бути використані для аналізу IoT-екосистем?

ЛАБОРАТОРНА РОБОТА 2

РОЗРОБКА ТЕХНІЧНОГО ЗАВДАННЯ ДЛЯ IOT-ДОСЛІДЖЕННЯ

Мета роботи:

- Ознайомитися з принципами розробки технічного завдання (ТЗ) для IoT-систем.
- Навчитися визначати вимоги до IoT-рішень.
- Сформувати структуру ТЗ на основі обраної предметної області.
- Визначити апаратні та програмні компоненти IoT-екосистеми.

Завдання до лабораторної роботи:

1. Ознайомитися з основними принципами складання технічного завдання.
2. Визначити основні функціональні та нефункціональні вимоги до IoT-системи.
3. Скласти структуру технічного завдання.
4. Визначити необхідне обладнання та програмне забезпечення.
5. Оформити результати у вигляді документа технічного завдання.

Теоретичні відомості:

Технічне завдання – це документ, що визначає вимоги до розробки, тестування та впровадження IoT-рішення.

Стандартна структура ТЗ включає:

1. Вступ – загальна характеристика проєкту.
2. Мета та завдання проєкту – очікуваний результат.
3. Функціональні вимоги – опис можливостей системи. Функціональні вимоги:
   - Система повинна збирати дані про рівень CO_2, температуру та вологість.
   - Дані мають передаватися на сервер у реальному часі.
   - Користувач повинен мати доступ до аналітики через веб-інтерфейс.
4. Нефункціональні вимоги – продуктивність, безпека, масштабованість. Нефункціональні вимоги:
   - Час відгуку системи не повинен перевищувати 5 секунд.

- Дані повинні зберігатися не менше 6 місяців.
- Підключення пристроїв має бути захищеним (шифрування даних).

5. Опис архітектури IoT-системи – компоненти системи, спосіб взаємодії.
6. Перелік необхідного обладнання та технологій – сенсори, мікроконтролери, мережеве обладнання.
7. Етапи реалізації – план розробки.
8. Очікувані результати та критерії оцінювання – як визначати успішність розробки.

Хід роботи:

Приклад складання структури технічного завдання.

1. Загальна характеристика

Проєкт передбачає створення IoT-системи для моніторингу якості повітря в місті.

2. Мета та завдання проєкту

Мета: створення доступної IoT-системи для аналізу якості повітря та представлення даних користувачам.

Завдання:

- Вибір апаратного забезпечення (сенсори, контролери).
- Налаштування збору та передачі даних.
- Розробка програмного забезпечення для обробки даних.
- Візуалізація інформації у вигляді веб-інтерфейсу.

3. Функціональні вимоги

- Збір даних кожні 10 секунд.
- Відображення статистики в реальному часі.
- Відправка повідомлень при перевищенні допустимих рівнів CO_2.

4. Нефункціональні вимоги

- Час реакції ≤ 5 секунд.
- Надійне зберігання даних у хмарі.
- Можливість підключення додаткових сенсорів.

5. Опис архітектури системи

Система складається з:

- Сенсорів (CO_2, температура, вологість).
- Контролера для збору даних (наприклад, ESP32, Raspberry Pi – контролери типу ESP32 або Raspberry Pi (одноплатні комп'ютери з архітектурою ARM) виконують функцію проміжної ланки між сенсорами та хмарною інфраструктурою).
- Мережевого модуля для передавання даних (Wi-Fi, LoRaWAN).
- Хмарного сервера для обробки та зберігання інформації.
- Веб-інтерфейсу для візуалізації отриманих результатів.

6. Перелік необхідного обладнання

- Датчик якості повітря (MQ-135).
- Датчик температури/вологості (DHT22).
- Контролер ESP32/Raspberry Pi.
- Бездротовий модуль (Wi-Fi або LoRaWAN).

7. Етапи реалізації

1. Аналіз вимог до системи.
2. Вибір обладнання та створення прототипу.
3. Розробка програмного забезпечення.
4. Тестування системи.
5. Оцінка продуктивності та оптимізація.

8. Очікувані результати

- Працездатна IoT-система для моніторингу якості повітря.
- Візуалізація отриманих даних у веб-інтерфейсі.
- Автоматизоване повідомлення про перевищення норм забруднення.

Контрольні питання:

1. Що таке технічне завдання і навіщо воно потрібне?
2. Які основні розділи містить ТЗ?
3. Які функціональні вимоги важливі для IoT-систем?
4. Як вибрати обладнання для IoT-рішення?

ЛАБОРАТОРНА РОБОТА 3
АНАЛІЗ МАТЕМАТИЧНИХ МЕТОДІВ І АЛГОРИТМІВ В ІОТ-ДОСЛІДЖЕННІ

Мета роботи:

- Ознайомитися з математичними методами та алгоритмами, що використовуються в дослідженнях IoT-екосистем.
- Проаналізувати ефективність алгоритмів для обробки даних в IoT.
- Обрати відповідний математичний підхід для вирішення поставленої задачі в рамках IoT-проєкту.
- Вивчити методи безпечної передачі даних у промислових IoT-мережах.
- Ознайомитися з налаштуванням VPN та шифруванням у IoT-системах.

Теоретичні відомості:

В IoT-системах збір, аналіз та передача даних є ключовими процесами. Математичні методи допомагають забезпечити:

- Фільтрацію шумів та аномалій у вхідних даних (фільтр Калмана, середнє ковзне).
- Обробку та класифікацію даних (кластеризація, машинне навчання).
- Оптимізацію використання ресурсів (евристичні алгоритми, алгоритми лінійного програмування).
- Прогнозування та виявлення тенденцій (регіональні моделі, часові ряди).
- Безпечну передачу даних (шифрування, VPN, протоколи безпеки).

Методи безпечної передачі даних у IoT:

1. Шифрування даних

- AES (Advanced Encryption Standard) – симетричне шифрування, що використовується для захисту переданих даних.
- RSA (Rivest-Shamir-Adleman) – асиметричний метод шифрування для захисту обміну ключами.
- ECC (Elliptic Curve Cryptography) – ефективне шифрування для пристроїв з обмеженими ресурсами.

2. Протоколи захисту передавання даних

- TLS (Transport Layer Security) – захист HTTP-трафіку (HTTPS) у веб-інтерфейсах IoT.
- DTLS (Datagram TLS) – розширення TLS для UDP, застосовується у CoAP-протоколах.
- MQTT-S (Secure MQTT) – використовується у брокерах повідомлень з вбудованою аутентифікацією.
- CoAP + DTLS – захист CoAP-запитів у малопотужних IoT-пристроях.

3. Налаштування VPN у IoT-мережах

- OpenVPN – тунельне з'єднання для безпечного віддаленого доступу до IoT-мереж.
- WireGuard – сучасний VPN-протокол з високою продуктивністю та простотою налаштування.

Завдання до виконання

1. Ознайомитися з математичними методами аналізу даних в IoT.
2. Вибрати метод обробки даних для конкретного типу інформації.
3. Реалізувати вибраний алгоритм у середовищі програмування або симуляторі.
4. Вивчити та налаштувати один із методів шифрування даних.
5. Реалізувати VPN-з'єднання для безпечного передавання IoT-даних.
6. Перевірити ефективність шифрування та аналізу даних.

Хід роботи

1. Вибір математичного методу

Залежно від типу задачі IoT (моніторинг, управління, прогнозування) обирається відповідний метод:

- Якщо потрібно зменшити шум у даних – фільтр Калмана, середнє ковзне.
- Якщо потрібно кластеризувати дані – алгоритм k-середніх.
- Якщо потрібно робити прогноз – регресійні методи, нейронні мережі.

2. Реалізація алгоритму

На основі вибраного математичного методу створюється програмна реалізація. Це може бути:

- Python (NumPy, SciPy, Scikit-learn для аналізу даних).
- MATLAB (для моделювання та обробки сигналів).
- Excel (для простого статистичного аналізу).

3. Приклад

Завдання:

Фільтрація шуму в даних про температуру за допомогою ковзного середнього.

Вхідні дані:

Температурні вимірювання:

[22.1, 22.3, 21.8, 22.0, 25.1, 21.9, 22.4, 22.5, 21.7, 22.2]

Реалізація в Python:

```
import numpy as np
data = [22.1, 22.3, 21.8, 22.0, 25.1, 21.9, 22.4, 22.5, 21.7, 22.2]
window_size = 3
# Розрахунок ковзного середнього
smoothed_data = np.convolve(data, np.ones(window_size)/window_size,
mode='valid')
print("Відфільтровані дані:", smoothed_data)
```

Очікуваний результат:

Згладжені значення температури без різких коливань.

4. Налаштування шифрування даних

Приклад реалізації AES-шифрування в Python:

```
from Crypto.Cipher import AES
import os

def encrypt_data(data, key):
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(data.encode())
    return cipher.nonce, ciphertext, tag

key = os.urandom(16)
nonce, ciphertext, tag = encrypt_data("IoT Secure Data", key)
print("Зашифровані дані:", ciphertext)
```

5. Налаштування VPN для IoT

Приклад створення WireGuard-конфігурації для IoT-з'єднання:

```
[Interface]
PrivateKey = <Ваш приватний ключ>
Address = 10.0.0.2/24

[Peer]
PublicKey = <Публічний ключ сервера>
Endpoint = vpn.server.com:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
```

Контрольні питання:

1. Які математичні методи використовуються для обробки даних в IoT?
2. У чому суть методу ковзного середнього?
3. Які алгоритми машинного навчання можуть бути використані для аналізу даних в IoT?
4. Як визначити, який математичний метод найкраще підходить для конкретного IoT-дослідження?
5. Що таке фільтр Калмана, і як він використовується в аналізі IoT-даних?
6. Які метрики оцінки ефективності алгоритмів аналізу даних використовуються?
7. Які існують методи усунення шуму у вхідних даних IoT?
8. Які методи шифрування найкраще підходять для IoT-пристроїв?
9. У чому переваги OpenVPN перед WireGuard у IoT-мережах?
10. Як працює TLS у безпечному передаванні IoT-даних?

ЛАБОРАТОРНА РОБОТА 4
АНАЛІЗ ТА ОПТИМІЗАЦІЯ МАРШРУТИЗАЦІЇ ДАНИХ В IoT-МЕРЕЖАХ

Мета роботи:

- Ознайомитися з основними методами маршрутизації даних в IoT-мережах.
- Дослідити алгоритми балансування навантаження в розподілених системах.
- Використати теорію графів для аналізу та оптимізації топології IoT-мереж.
- Практично застосувати алгоритми маршрутизації та балансування навантаження.

Теоретичні відомості:

Методи маршрутизації в IoT:

1. Статична маршрутизація – використання наперед визначених маршрутів для передавання даних.
2. Динамічна маршрутизація – адаптація маршрутів на основі змін в мережі (завантаження каналів, збої).
3. Протоколи маршрутизації в IoT:
    - RPL (Routing Protocol for Low-Power and Lossy Networks) – оптимізований для сенсорних мереж.
    - AODV (Ad-hoc On-Demand Distance Vector) – використовується в мобільних мережах.
    - OLSR (Optimized Link State Routing) – базується на обміні інформацією між вузлами.

Алгоритми балансування навантаження:

1. Метод найменшого навантаження – дані спрямовуються через найменш завантажені вузли.
2. Метод випадкового розподілу – рівномірний розподіл запитів між доступними серверами.
3. Round Robin – чергова передача даних між вузлами.

4. Least Connections – передача даних до вузла з найменшою кількістю активних з'єднань.
5. Adaptive Load Balancing – адаптивний розподіл з урахуванням поточного стану мережі.

Теорія графів у топології IoT-мереж:

1. Граф як модель IoT-мережі:
    - Вузли графа – IoT-пристрої.
    - Ребра – канали зв'язку між пристроями.
2. Алгоритми пошуку оптимального маршруту:
    - Алгоритм Дейкстри – знаходження найкоротшого шляху в мережі.
    - Алгоритм Беллмана-Форда – знаходження маршруту у зважених графах.
    - Алгоритм Флойда-Варшалла – визначення найкоротших шляхів між усіма вузлами.

Завдання до виконання:

1. Ознайомитися з методами маршрутизації та балансування навантаження.
2. Використати графові алгоритми для оптимізації маршрутів передачі даних.
3. Реалізувати алгоритм балансування навантаження в симуляційному середовищі.
4. Проаналізувати ефективність методів маршрутизації в різних сценаріях IoT.

Хід роботи:

1. **Створення IoT-мережі:**
    - Визначити топологію мережі (зіркова, деревоподібна, mesh).
    - Створити граф мережі (вузли – IoT-пристрої, ребра – зв’язки).
2. **Розрахунок оптимального маршруту:**
    - Використати алгоритм Дейкстри для визначення найкоротшого шляху.

- Реалізувати балансування навантаження за допомогою алгоритму Round Robin.

3. **Приклад реалізації алгоритму в Python:**

```
import networkx as nx
G = nx.Graph()
G.add_edges_from([(1, 2, {'weight': 5}), (2, 3, {'weight': 2}), (1, 3, {'weight': 9})])
shortest_path = nx.dijkstra_path(G, source=1, target=3, weight='weight')
print("Найкоротший маршрут:", shortest_path)
```

4. **Тестування та аналіз:**
   - Запустити моделювання з різними алгоритмами.
   - Проаналізувати продуктивність та затримку передачі даних.

Контрольні питання:

1. Які основні методи маршрутизації застосовуються в IoT?
2. Як працює алгоритм Дейкстри?
3. Які алгоритми використовуються для балансування навантаження?
4. Як теорія графів допомагає в оптимізації IoT-мереж?
5. Які фактори впливають на вибір маршрутизації в сенсорних мережах?

ЛАБОРАТОРНА РОБОТА 5
РЕАЛІЗАЦІЯ ТА ДЕМОНСТРАЦІЯ ІОТ-РІШЕННЯ НА ОСНОВІ РОЗРОБЛЕНОЇ АРХІТЕКТУРИ

Мета роботи:

- Завершити розробку функціонального прототипу IoT-системи відповідно до технічного завдання.
- Реалізувати взаємодію апаратної та програмної складових проєкту.
- Забезпечити безпечну передачу даних, збереження інформації та доступ до неї через веб-інтерфейс.
- Оцінити працездатність, зручність використання та стабільність розробленої системи.
- Підготувати коротку презентацію для захисту проєкту.

Теоретичні відомості:

Побудова повноцінного IoT-рішення передбачає інтеграцію кількох архітектурних рівнів: апаратного, мережевого, програмного та візуалізаційного. У цій лабораторній роботі реалізується завершений прототип IoT-системи, що має демонструвати роботу в умовах, наближених до реальних.

1. Апаратний рівень (сенсори, контролери, шлюзи):

Сенсори фіксують фізичні параметри (температура, вологість, освітлення, тиск тощо). Дані обробляються на мікроконтролері (наприклад, ESP32, Arduino), який виконує роль локального вузла. Дані передаються далі через агрегатор або шлюз — пристрій, який об’єднує локальну мережу з глобальною (через Wi-Fi, Ethernet, GSM, LoRa, тощо).

2. Мережевий рівень:

Організація зв’язку між пристроями — ключова складова IoT-систем. Для передачі даних використовуються:

- локальні мережі (LAN, Zigbee, BLE, Wi-Fi),
- протоколи прикладного рівня (MQTT, HTTP/HTTPS, CoAP),
- глобальні канали (LoRaWAN, GSM, 5G, NB-IoT).

3. Захист і безпека даних:

IoT-системи вразливі до атак через:

- незахищене передавання даних,
- відкриті порти,
- слабку автентифікацію.

Захист реалізується через:

- шифрування даних (TLS, AES),
- VPN-канали (WireGuard, OpenVPN),
- контроль доступу та авторизацію користувачів (OAuth2, JWT).

4. Програмне забезпечення та база даних:

Отримані дані можуть оброблятись локально (на edge-пристроях), або відправлятись у хмару. Зазвичай використовуються:

- SQL/NoSQL-бази даних (SQLite, PostgreSQL, InfluxDB, MongoDB),
- серверна логіка (Flask, Node.js, .NET),
- сервіси хмарної обробки (AWS IoT, ThingsBoard, Azure IoT Hub).

5. Веб-інтерфейс та візуалізація:

Для користувача важливий зручний спосіб доступу до даних. Веб-інтерфейс дозволяє:

- переглядати поточні та історичні значення,
- налаштовувати порогові значення або події,
- віддалено керувати пристроями.

Для візуалізації застосовуються:

- HTML + CSS + JavaScript + Chart.js / Plotly / D3.js,
- бібліотеки Python (Dash, Streamlit),
- готові платформи (Grafana, Node-RED Dashboard).

6. Тестування IoT-рішення, яке охоплює:

- перевірку на втрату даних, затримки, повтори;
- поведінку при розриві з’єднання;
- правильність запису в базу даних;
- відповідність веб-інтерфейсу функціоналу.

Хід роботи

Крок 1. Побудова архітектури IoT-рішення

- Визначити ключові компоненти: сенсори, контролер (наприклад, ESP32), шлюз/агрегатор, локальна мережа, серверна частина, база даних, веб-інтерфейс.
- Обрати спосіб передачі даних: Wi-Fi, MQTT, HTTP-запити, LoRaWAN тощо.
- Визначити формат даних (JSON, CSV) і частоту передачі.

Крок 2. Реалізація передачі даних

- Якщо немає фізичних сенсорів — використати емуляцію даних (наприклад, генерацію випадкових температур або вологості).
- Встановити симулятор або написати скрипт, який надсилає дані на сервер за HTTP/MQTT.

Приклад на Pyton:

```
# client_simulator.py — імітація пристрою, що надсилає дані
import requests
import random
import time

URL = 'http://localhost:5000/data'

while True:
    payload = {
        'temperature': round(random.uniform(20.0, 25.0), 2),
        'humidity': round(random.uniform(30.0, 50.0), 2)
    }
    requests.post(URL, json=payload)
    print('Sent:', payload)
    time.sleep(5)
```

Крок 3. Реалізація прийому та збереження даних

- Розгорнути сервер на Flask або іншій платформі, який прийматиме дані й зберігатиме їх у базу (наприклад, SQLite або PostgreSQL).

Приклад на Pyton:

```
# server.py — сервер, що приймає дані
from flask import Flask, request, jsonify
import sqlite3

app = Flask(__name__)

# Ініціалізація БД
```

```
conn = sqlite3.connect('iot_data.db', check_same_thread=False)
conn.execute('''CREATE TABLE IF NOT EXISTS readings (id INTEGER PRIMARY KEY,
temperature REAL, humidity REAL)''')

@app.route('/data', methods=['POST'])
def receive_data():
    data = request.get_json()
    conn.execute('INSERT INTO readings (temperature, humidity) VALUES (?, ?)',
                 (data['temperature'], data['humidity']))
    conn.commit()
    return jsonify({'status': 'OK'})

@app.route('/')
def index():
    cur = conn.cursor()
    cur.execute('SELECT * FROM readings ORDER BY id DESC LIMIT 10')
    data = cur.fetchall()
    return '<br>'.join([f'Temperature: {t}°C | Humidity: {h}%' for _, t, h in data])

if __name__ == '__main__':
    app.run(debug=True)
```

Крок 4. Забезпечення безпеки передавання даних

- Якщо використовуєте HTTP — реалізувати HTTPS через Flask-TLS або reverse proxy (наприклад, nginx із сертифікатом).
- Для MQTT — налаштувати аутентифікацію (логін/пароль) і TLS-з’єднання.
- Для VPN — використовувати WireGuard/OpenVPN для тунелювання мережі між вузлом і сервером.

Крок 5. Розробка веб-інтерфейсу

- У найпростішому випадку – HTML-сторінка, яка відображає дані з бази або графіки (наприклад, Chart.js).
- Альтернатива — Dash, Streamlit або Plotly для інтерактивної візуалізації.

Крок 6. Тестування системи

- Провести перевірку коректності надсилання, прийому та збереження даних.
- Імітувати помилкові/аномальні значення.
- Перевірити стійкість веб-інтерфейсу, відповідність даних.

Крок 7. Підготовка презентації

- Описати призначення та функціонал системи.
- Навести діаграму архітектури.
- Показати фрагменти коду, результат роботи сервера, приклад інтерфейсу.
- Описати використані протоколи безпеки та базу даних.

Контрольні питання:

1. Яка мережна архітектура використана у вашому рішенні?
2. Як організовано безпечну передачу даних?
3. Яким чином взаємодіє програмне забезпечення з базою даних?
4. Які можливості надає веб-інтерфейс вашої IoT-системи?
5. Які інструменти ви використали для забезпечення надійності системи?
6. Що включає ваша демонстраційна презентація?

Рекомендована література

Основна література:

1. Пулеко І.В., Єфіменко А.А. Архітектура та технології Інтернету речей: навч. посіб. – Житомир: Державний університет «Житомирська політехніка», 2022. – 234 с.
2. Arora S. Design of Secure IoT Systems: A Practical Approach Across Industries. – McGraw-Hill, 2021. – 224 p.
3. Kurniawan A. Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32. – Packt Publishing, 2019. – 239 p.
4. Internet of Things in Modern Computing. Theory and Applications / Ed. by V. Chowdary, A. Sharma, N. Kumar, V. Kaundal. – Taylor & Francis, 2023. – 239 p.

Допоміжна література:

5. Шингера Н.Я., Олійник Т.М. Архітектура комп’ютерних систем з IoT компонентами // Матеріали VII Міжнародної науково-технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій». – Тернопіль, 2018. – С. 135-138.
6. Коваль М.І. Архітектура та протоколи Інтернету речей в розумному місті // Логістика, підприємництво та електронна комерція. – 2024. – С. 76-82.

Додаток А

Варіанти тем для IoT-проекту:

1. Архітектура IoT для "розумного будинку".
2. Оптимізація енергоспоживання в IoT-системах.
3. Використання IoT у сільському господарстві.
4. Кібербезпека в IoT-екосистемах.
5. IoT для розумного міста: виклики та перспективи.
6. Аналіз трафіку IoT-мереж.
7. Використання штучного інтелекту в IoT-пристроях.
8. IoT в медицині: телеметрія та дистанційний моніторинг.
9. Прогнозування технічного обслуговування IoT-пристроїв.
10. IoT в промисловості: автоматизація та контроль.
11. Ефективні протоколи зв’язку для IoT.
12. Виявлення аномалій у великих IoT-мережах.
13. Кластеризація IoT-пристроїв для оптимізації роботи мережі.
14. IoT для моніторингу навколишнього середовища.
15. Використання блокчейну в IoT-екосистемах.
16. Оптимізація передачі даних у сенсорних мережах.
17. IoT для транспорту: моніторинг та безпека.
18. Архітектура серверів для обробки IoT-даних.
19. Аналіз ефективності IoT-мереж.
20. IoT у логістиці: автоматизація та відстеження товарів.