ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Міністерство освіти та науки України
Харківський національний університет радіоелектроніки

Кафедра ЕОМ

Кваліфікаційна робота
на тему:

Методи обробки та аналізу даних в IoT

студент:
групи СПм-23-3
До Нгок Куен

Керівник:
проф. каф. ЕОМ
Міхаль О.П.

Харків 2025

Аналіз предметної області

Метою кваліфікаційної роботи є розробка та реалізація методу діагностування технічного стану сенсорних пристроїв у розподіленій автоматизованій системі моніторингу, що функціонує на основі архітектури Інтернету речей. Особлива увага приділяється інтеграції методів машинного навчання для виявлення відхилень у поведінці сенсорів, що дозволяє підвищити надійність, стабільність і передбачуваність роботи екологічної інфраструктури.

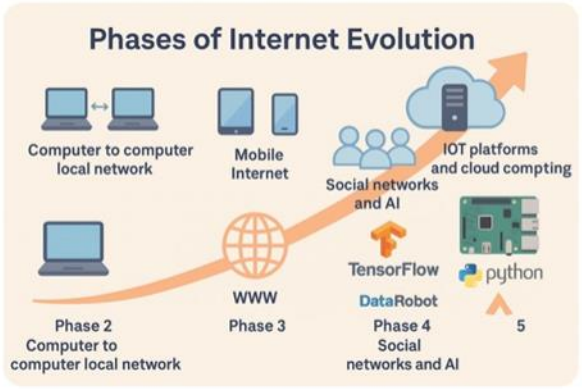
Завдання:

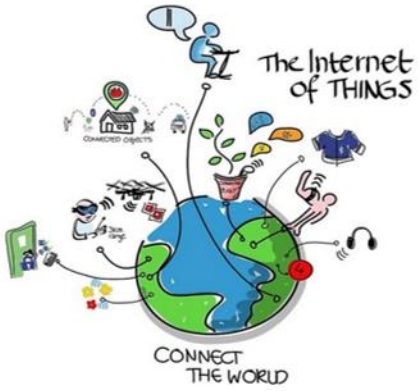
- провести системний аналіз технологій Інтернету речей у контексті екологічного моніторингу;
- дослідити існуючі методи обробки даних та розподілу завдань між вузлами IoT-мереж;
- розробити метод виявлення аномалій і потенційних несправностей у сенсорній мережі на основі штучних нейронних мереж;
- реалізувати розроблений метод та проаналізувати результати.

Об'єктом дослідження є розподілена система моніторингу навколишнього середовища, що функціонує на основі технологій Інтернету речей (IoT) та включає: вимірювальні пристрої, які автономно фіксують параметри стану довкілля, інфраструктуру передачі данихінфраструктуру передачі даних (сенсорні мережі, протоколи зв'язку типу MQTT), серверну й хмарну аналітику, включно з модулями предиктивного аналізу й діагностики, архітектуру взаємодії елементів, що забезпечує безперервне функціонування системи, її масштабованість і відмовостійкість.

3

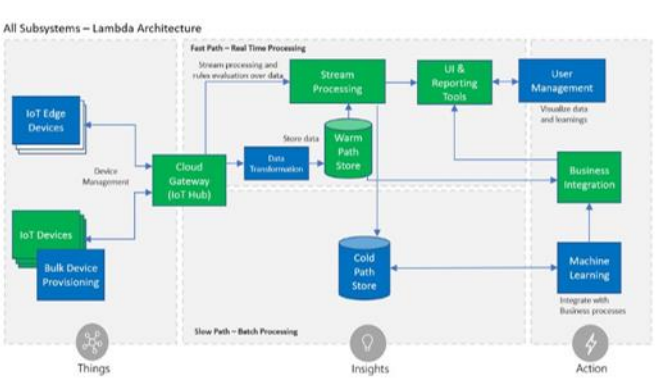
Класифікація IoT

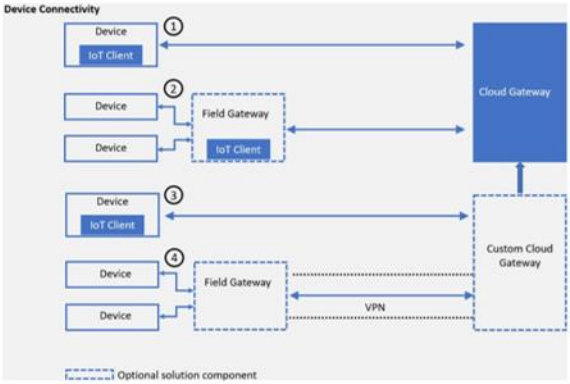




4

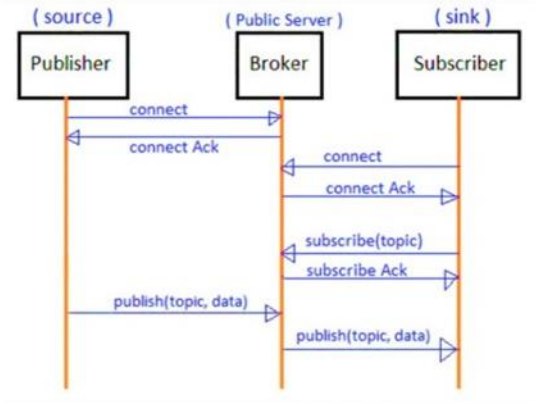
Архітектура IoT рішення

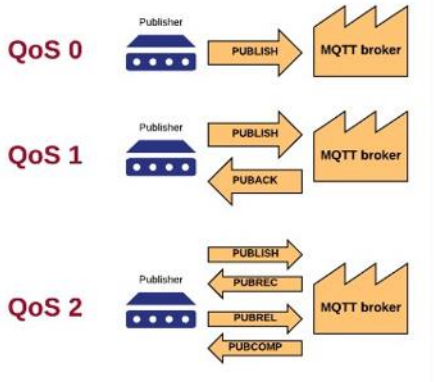




5

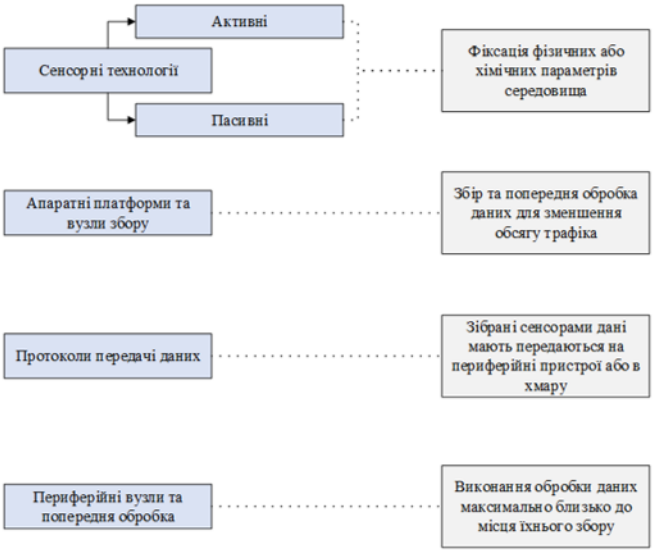
Передача даних в архітектурі IoT: MQTT





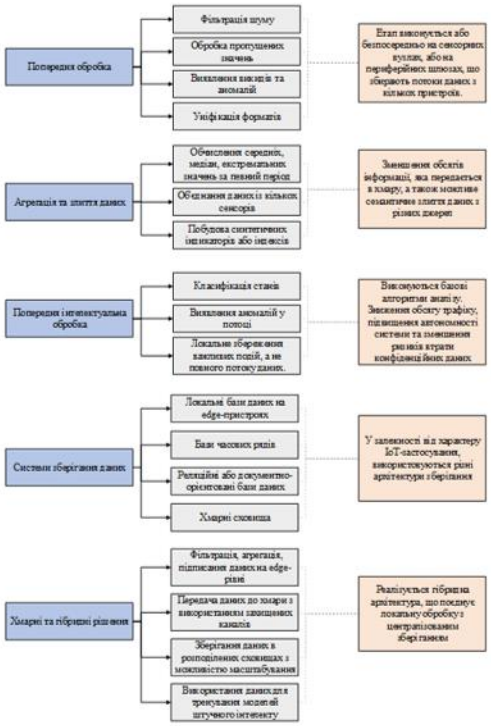
6

Існуючі методи збору даних в IoT



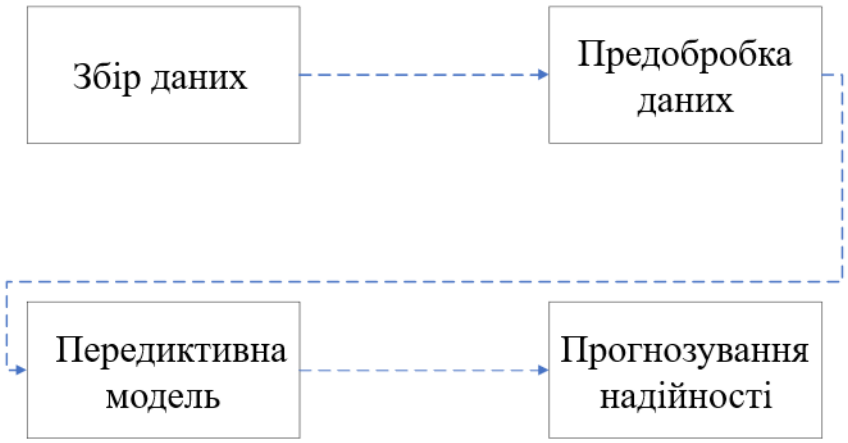
Методи попередньої обробки та зберігання даних в IoT

7



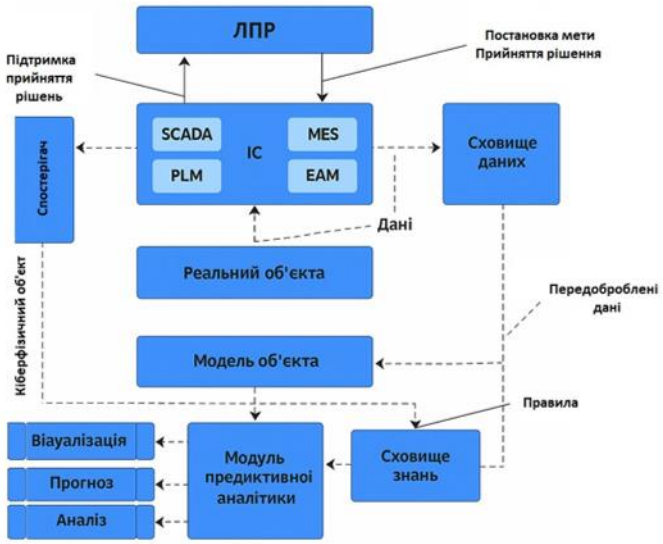
Процес обробки даних в обчислювальному вузлі системи предиктивної аналітики

8



Структура типової системи обробки діагностичної інформації

9



10

Метод для виявлення та прогнозування аномальних станів сенсорних пристроїв у розподіленій системі моніторингу довкілля

11

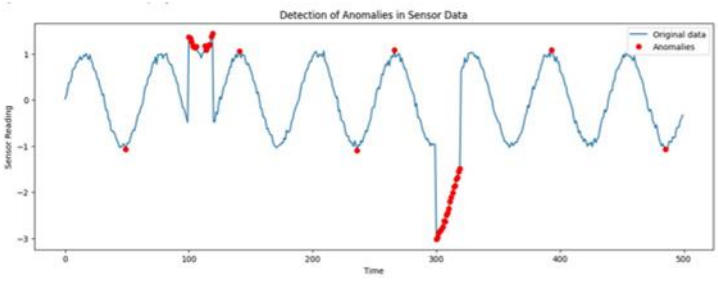
Фрагменти реалізації методу в Google Colab

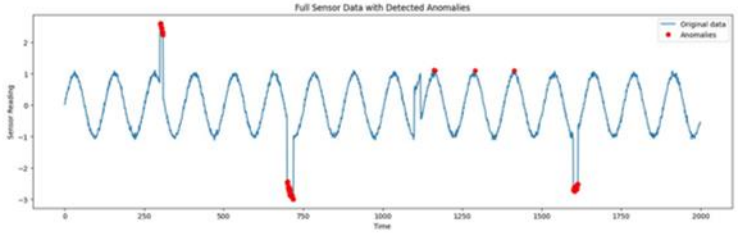
```
# Встановлення необхідних бібліотек
!pip install numpy pandas sklearn tensorflow matplotlib

# Імпорт бібліотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Створення синтетичних даних з аномаліями
np.random.seed(42)
time_steps = 500
normal_data = np.sin(np.linspace(0, 50, time_steps)) + np.random.normal(0, 0.05, time_steps)
anomalous_data = normal_data.copy()
anomalous_data[100:120] += 2
anomalous_data[300:320] -= 2
df = pd.DataFrame({'value': anomalous_data})

# Масштабування даних
scaler = MinMaxScaler()
df['scaled'] = scaler.fit_transform(df[['value']])

# Вибір нормальних даних для тренування
normal_indices = list(range(0,100)) + list(range(120,300)) + list(range(320,500))
X_normal = df.loc[normal_indices, 'scaled'].values.reshape(-1, 1)
X_train, X_test = train_test_split(X_normal, test_size=0.2, shuffle=False)

# Створення та навчання автоенкодера
model = Sequential([
    Dense(16, activation='relu', input_shape=(1,)),
    Dense(8, activation='relu'),
    Dense(16, activation='relu'),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
history = model.fit(X_train, X_train, epochs=50, batch_size=16, validation_data=(X_test, X_test))

# Прогнозування та виявлення аномалій
df['reconstruction'] = model.predict(df['scaled'].values.reshape(-1,1))
df['loss'] = np.abs(df['scaled'] - df['reconstruction'])
threshold = df.loc[normal_indices, 'loss'].mean() + 3 * df.loc[normal_indices, 'loss'].std()
df['anomaly'] = df['loss'] > threshold

# Візуалізація результатів
plt.figure(figsize=(15,5))
plt.plot(df['value'], label='Original data')
plt.plot(df[df['anomaly']].index, df[df['anomaly']]['value'], 'ro', label='Anomalies')
plt.legend()
plt.title('Detection of Anomalies in Sensor Data')
plt.xlabel('Time')
plt.ylabel('Sensor Reading')
```

```
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 1.8335e-09 - val_loss: 4.9095e-12
Epoch 26/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 6.4649e-10 - val_loss: 1.4452e-11
Epoch 27/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 8.0893e-10 - val_loss: 1.0828e-11
Epoch 28/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.0387e-09 - val_loss: 4.8675e-12
Epoch 29/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.9406e-10 - val_loss: 8.6983e-12
Epoch 30/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.4419e-10 - val_loss: 4.4423e-12
Epoch 31/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.1799e-10 - val_loss: 2.8466e-11
Epoch 32/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.7418e-10 - val_loss: 8.7456e-13
Epoch 33/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 8.3672e-11 - val_loss: 5.7377e-12
Epoch 34/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.2277e-10 - val_loss: 1.8725e-12
Epoch 35/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2.0597e-10 - val_loss: 8.1240e-13
Epoch 36/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.1979e-10 - val_loss: 2.0806e-12
Epoch 37/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 7.3487e-11 - val_loss: 3.3239e-12
Epoch 38/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 1.3940e-10 - val_loss: 5.5253e-14
Epoch 39/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 4.3093e-11 - val_loss: 1.5464e-12
Epoch 40/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.9987e-11 - val_loss: 7.7907e-13
Epoch 41/50
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 7.2228e-11 - val_loss: 1.9014e-13
```

12

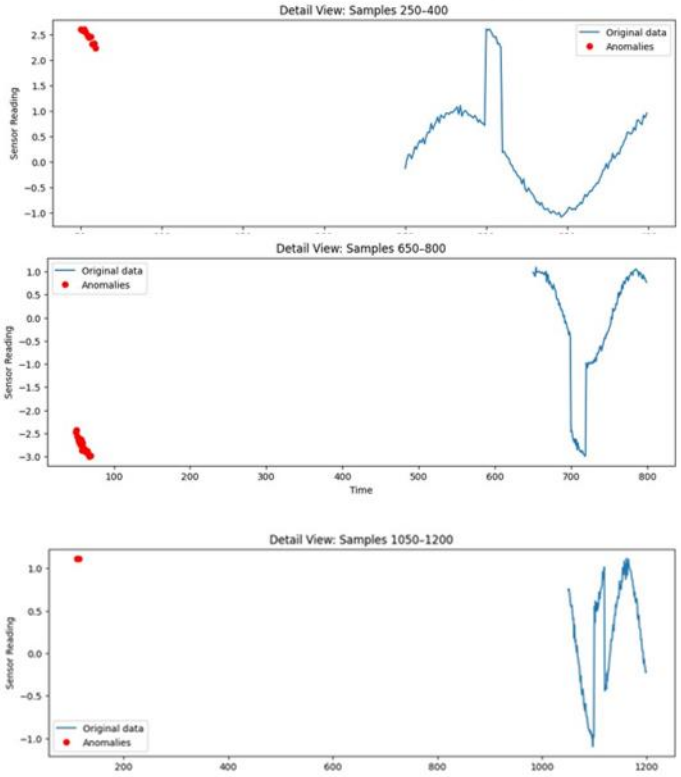
Аналіз отриманих результатів





13

Аналіз отриманих результатів



14

Апробація роботи

DATA PROCESSING AND ANALYSIS METHODS IN IOT USING MACHINE LEARNING

Introduction

Analysis of publications.

Main part

Do K., Klymova I., Naumova E., Herevych M., Yankovskyi O. Data processing and analysis methods in IOT using machine learning. Системи управління, навігації та зв'язку, вип.2. Полтава, 2025. С. 119-124.

15

ВИСНОВКИ

У результаті проведеного дослідження було розроблено, обґрунтовано та програмно реалізовано метод діагностування несправностей у розподіленій системі моніторингу довкілля на основі технологій Інтернету речей з використанням інструментів машинного навчання. Запропонована архітектура передбачає організацію потокової обробки даних з великої кількості сенсорних пристроїв, що працюють у режимі реального часу, з подальшою реконструкцією нормальної поведінки системи за допомогою нейронної мережі типу автоенкодер. Це дозволяє виявляти відхилення, які потенційно сигналізують про вихід з ладу обладнання або порушення в роботі компонентів системи.

У дослідженні було реалізовано повноцінний цикл аналізу: від генерації імітованих даних із вбудованими аномаліями до навчання моделі на нормальних вибірках та ідентифікації відхилень на основі реконструктивної похибки. Побудовані графіки підтвердили високу ефективність запропонованого підходу – всі ключові аномалії були точно виявлені навіть на складних ділянках сигналу. Метод продемонстрував здатність адаптуватися до різних типів порушень, таких як імпульсні збурення, плавні зміщення або комбінації відхилень.

Особливу увагу приділено структурній моделі обробки діагностичної інформації, у межах якої були розглянуті способи виявлення аномалій на основі класифікаційних, кластеризаційних, статистичних та гібридних методів. У якості найбільш придатного підходу для цільового завдання було обґрунтовано застосування автоенкодера з можливістю реконструкції часових рядів і виявлення нетипових сегментів.

Важливим елементом роботи стало також формування прогнозних моделей для предиктивного обслуговування, що базуються на накопичених даних та історії змін у поведінці сенсорів. Це дозволяє не лише фіксувати факт наявності несправностей, а й формувати інтервали ймовірного відмовлення з метою завчасного втручання. Такий підхід значно підвищує надійність, безперервність і адаптивність функціонування розподіленої IoT-системи моніторингу.

Окремо було проаналізовано актуальні наукові публікації, що підтвердили міждисциплінарний характер тематики та потребу в об'єднанні знань з інформатики, штучного інтелекту, телекомунікацій та інформаційної безпеки. Зроблено висновок, що перспективи розвитку таких систем мають прямий зв'язок із вдосконаленням механізмів захисту даних, підвищенням прозорості прийняття рішень у нейромережах та етичним впровадженням аналітики в чутливих середовищах.

Загалом, результати роботи засвідчили доцільність використання інтелектуальних методів для моніторингу стану сенсорних пристроїв у великих розподілених IoT-мережах. Запропоноване рішення може стати основою для побудови систем раннього попередження про несправності, систем екологічного контролю, а також платформ для цифрового управління розумними середовищами.

ДОДАТОК Б

Програмний код

Б.1 Підготовка середовища та даних

```
# Встановлюємо бібліотеки
!pip install numpy pandas sklearn tensorflow matplotlib

# Імпорт бібліотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Б.2 Створення тестових даних

```
# Генеруємо нормальні (без аномалій) дані
np.random.seed(42)
time_steps = 500
normal_data = np.sin(np.linspace(0, 50, time_steps)) +
np.random.normal(0, 0.05, time_steps)

# Вводимо штучні аномалії
anomalous_data = normal_data.copy()
anomalous_data[100:120] += 2
anomalous_data[300:320] -= 2

# Перетворення у DataFrame
df = pd.DataFrame({'value': anomalous_data})
```

Б.3 Підготовка даних

```
scaler = MinMaxScaler()
df['scaled'] = scaler.fit_transform(df[['value']])
# Формуємо набори даних для навчання (тільки нормальні дані)
normal_indices = list(range(0,100)) + list(range(120,300)) +
list(range(320,500))
X_normal = df.loc[normal_indices, 'scaled'].values.reshape(-1,
1)

X_train, X_test = train_test_split(X_normal, test_size=0.2,
```

```
shuffle=False)
```

Б.4 Навчання автоенкодера

```
model = Sequential([
    Dense(16, activation='relu', input_shape=(1,)),
    Dense(8, activation='relu'),
    Dense(16, activation='relu'),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')

history = model.fit(X_train, X_train,
                    epochs=50,
                    batch_size=16,
                    validation_data=(X_test, X_test)
```

Б.5 Детектування аномалій

```
df['reconstruction'] =
model.predict(df['scaled'].values.reshape(-1,1))
df['loss'] = np.abs(df['scaled'] - df['reconstruction'])

threshold = df.loc[normal_indices, 'loss'].mean() + 3 *
df.loc[normal_indices, 'loss'].std()

df['anomaly'] = df['loss'] > threshold
```

Б.6 Візуалізація результатів

```
# Візуалізація: весь сигнал
plt.figure(figsize=(18,5))
plt.plot(df['value'], label='Original data')
plt.plot(df[df['anomaly']].index, df[df['anomaly']]['value'],
'ro', label='Anomalies')
plt.legend()
plt.title('Full Sensor Data with Detected Anomalies')
plt.xlabel('Time')
plt.ylabel('Sensor Reading')
plt.show()

# Візуалізація окремих ділянок
for start in [250, 650, 1050, 1550]:
    plt.figure(figsize=(12,4))
    plt.plot(df['value'].iloc[start:start+150], label='Original
data')
    segment = df.iloc[start:start+150]
```

```
    plt.plot(segment[segment['anomaly']].index - start,
segment[segment['anomaly']]['value'], 'ro', label='Anomalies')
    plt.legend()
    plt.title(f'Detail View: Samples {start}-{start+150}')
    plt.xlabel('Time')
    plt.ylabel('Sensor Reading')
    plt.show()
```