МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ОБЕРВАНЮК Уляна Володимирівна

Інтелектуальний програмний модуль моніторингу роботи електродвигуна з використанням IoT / Intelligent Software Module for Monitoring of an Electric Motor Using IoT

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Штучний інтелект

Кваліфікаційна робота

Виконала студентка групи КНШІ-41
У. В. Оберванюк

Науковий керівник:
к.т.н., доцент, О.Р. Осолінський

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н.М. Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Штучний інтелект

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
ОБЕРВАНЮК Уляна Володимирівна
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Інтелектуальний програмний модуль моніторингу роботи електродвигуна з використанням IoT / Intelligent Software Module for Monitoring of an Electric Motor Using IoT
керівник роботи О.Р. Осолінський, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- проаналізувати технологій інтелектуальних систем управління електродвигунами;
- дослідити системи моніторингу та діагностики електродвигунів за допомногою ШІ та IoT;
- сформувати постановку задачі;
- розробити концепцію та архітектуру модуля моніторингу роботи електродвигуна з використанням IoT;
- розробити алгоритмічне забезпечення програмного модуля
- розробити інформаційне забезпечення з врахуванням журналу подій, структурованого зберігання вимірювань та діагностичних міток;
- провести тестування та оцінку ефективності розробленого модуля на основі реальних та синтетичних даних.
5. Перелік графічного матеріалу в роботі:
- архітектура програмного модуля;
- основні алгоритми програмного модуля;

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ У. В. Оберванюк
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ О.Р. Осолінський
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 58 с., 24 рис., 4 додатки, 52 джерела.

Метою роботи є розробка інтелектуальної системи діагностики стану електродвигунів на основі обробки сенсорних даних та глибокого навчання.

В роботі використано методи аналізу часових рядів, екстракції ознак, класифікації на основі штучних нейронних мереж, а також алгоритми виявлення аномалій. Реалізація проведена із застосуванням мікроконтролера ESP32, сенсорної підсистеми, середовища Google Colab та бібліотек TensorFlow і scikit-learn.

Запропоновано програмно-апаратне рішення, яке здійснює збір, фільтрацію, нормалізацію та пакетну передачу сенсорних даних у реальному часі для подальшої класифікації технічного стану електродвигуна. Реалізовано MLP-модель, що виконує багатокласову класифікацію станів, таких як «норма», «попередження» та «критичний стан». Проведено тестування моделі на основі зібраного датасету, побудовано візуалізації ефективності навчання.

Розроблена система дозволяє підвищити точність виявлення потенційних несправностей двигунів, забезпечує раннє попередження та може бути інтегрована в промислові IoT-рішення.

Ключові слова: ESP32, IoT, ГЛИБОКЕ НАВЧАННЯ, ЕЛЕКТРОДВИГУН, ДІАГНОСТИКА, СЕНСОРНІ ДАНІ, КЛАСИФІКАЦІЯ, НЕЙРОННА МЕРЕЖА.

ANNOTATION

Explanatory note to the qualification thesis: 58 pages, 24 figures, 4 appendices, 52 references.

The aim of this work is to develop an intelligent system for diagnosing the condition of electric motors based on sensor data processing and deep learning techniques.

The study employs methods of time series analysis, feature extraction, classification based on artificial neural networks, as well as anomaly detection algorithms. The implementation is carried out using the ESP32 microcontroller, a sensor subsystem, the Google Colab environment, and the TensorFlow and scikit-learn libraries.

A hardware-software solution is proposed that performs collection, filtering, normalization, and batch transmission of sensor data in real time for further classification of the electric motor's technical condition. An MLP model has been implemented to perform multiclass classification of states such as “normal”, “warning”, and “critical condition”. The model was tested on a collected dataset, and visualizations of training performance were constructed.

The developed system enhances the accuracy of detecting potential motor faults, provides

Keywords: ESP32, IoT, DEEP LEARNING, ELECTRIC MOTOR, DIAGNOSIS, SENSOR DATA, CLASSIFICATION, NEURAL NETWORK.

ЗМІСТ

ВСТУП

Останніми роками промислова революція, що триває, дозволила робити широкий і точний моніторинг різноманітних процесів ціною. Цей прогрес значною мірою пояснюється появою Інтернету речей (IoT), який розвинувся завдяки інтеграції різноманітних технологій, таких як повсюдне обчислення, датчики товарів, потужні вбудовані системи та машинне навчання. Одним із важливих застосувань IoT є його впровадження в моніторинг систем для діагностичних цілей.

Завдяки можливості підключатися та збирати дані практично з усього і скрізь, IoT полегшує комплексний моніторинг складних систем, дозволяючи виявляти аномалії та потенційні проблеми. Використовуючи потужність Інтернету речей, промисловість може отримати безцінне уявлення про здоров'я та продуктивність свого обладнання та процесів, тим самим підвищуючи ефективність і передбачувані практики технічного обслуговування. Така конвергенція технологій прокладає шлях до трансформаційних змін у різних секторах, відкриває нові можливості та оптимізує діяльність у промисловому ландшафті [1–7]. IoT має вирішальне значення для реалізації систем діагностики, створюючи мережу, яка з'єднує розумні пристрої в інформаційній системі та полегшує обмін даними з центральним сховищем. Мережевий зв'язок між пристроями IoT демонструє унікальні характеристики та відмінності порівняно зі звичайними пристроями [8], особливо в хмарно-туманному середовищі [9].

Сучасний рівень техніки показує відсутність загальної моделі витрат на технічне обслуговування, застосовної в усіх промислових контекстах, причому запропоновані моделі часто є специфічними для конкретних випадків або складними для практичного впровадження. Крім того, у документах про оптимізацію витрат на прогнозну модель обслуговування (PdM) зазвичай не помічається концепція ризику та рідко враховується невизначеність прийняття людських рішень. Промислове технічне обслуговування в минулому розглядалося як реактивна фонова діяльність, яка виконувалася лише після збою системи. Однак із появою профілактичного та прогнозного технічного обслуговування ця сфера

значно розвинулася, пропонуючи різні стратегії з явними перевагами та проблемами [11]. Тому розробка систем для моніторингу роботи електродвигуна з використанням IoT та ШІ є актуальною задачею, вирішення якої дасть можливість зменшити частоту критичних відмов, та поломок в таких системах.

Метою кваліфікаційної роботи є розробка інтелектуального програмного модуля моніторингу роботи електродвигуна з використанням IoT.

Завдання кваліфікаційної роботи полягають у послідовному виконанні наступних етапів:

– проаналізувати технологій інтелектуальних систем управління електродвигунами;

– дослідити системи моніторингу та діагностики електродвигунів за допомногою ШІ та IoT;

– сформувати постановку задачі;

– розробити концепцію та архітектуру модуля моніторингу роботи електродвигуна з використанням IoT;

– розробити алгоритмічне забезпечення програмного модуля

– розробити інформаційне забезпечення з врахуванням журналу подій, структурованого зберігання вимірювань та діагностичних міток;

– провести тестування та оцінку ефективності розробленого модуля на основі реальних та синтетичних даних.

Об’єкт дослідження - процеси моніторингу та діагностики технічного стану електродвигунів за допомогою сенсорних технологій.

Предмет дослідження - методи виявлення технічного стану електродвигунів з використанням штучних нейронних мереж та алгоритмів глибокого навчання.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах студентської науково-практичної конференції “Інтелектуальні інформаційні технології в прикладних дослідженнях” (IITAR – 2025), м. Тернопіль, Україна, 27-29 травня 2025 р.

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

1 АНАЛІЗ ТЕХНОЛОГІЙ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ УПРАВЛІННЯ ЕЛЕКТРОДВИГУНАМИ

1.1 Опис технологій Інтернету речей для діагностування електромоторів

Поява технології Інтернету речей значною мірою вплинула на прогнозне обслуговування та промисловий моніторинг. Інтеграція IoT з недорогими датчиками, машинним навчанням, складними вбудованими системами та повсюдним обчисленням спростила масштабний моніторинг і діагностику в складних промислових системах. Використання IoT для вдосконалення промислових процесів було ретельно вивчено в минулому, з особливим наголосом на прогнозне обслуговування, виявлення аномалій і діагностичні системи. посилання [1] зосереджується на кількох ключових сферах досліджень керування на основі Інтернету речей та застосувань у системах перетворення енергії вітру з асинхронними машинами та системами електродвигунів.

Моніторинг стану та діагностика несправностей у сучасній автоматизації виробництва життєво важливі для підвищення якості, продуктивності та захисту машин [2]. У нещодавньому спеціальному випуску про діагностику та прогнозування складних промислових систем було представлено п’ятнадцять статей, розділених на підходи, що ґрунтуються на моделях, і підходи, що керуються даними [3]. Методи, засновані на моделі, спираються на встановлені моделі системи та порівнюють очікувані змодельовані показники ефективності з даними спостереження. Навпаки, керовані даними підходи, часто засновані на кластеризації або машинному навчанні, виявляють сигнали, пов’язані з несправністю, без попередньої інформації про несправність. У попередній роботі, присвяченій складним промисловим системам [4], метод виявлення несправностей на основі k-середніх виділяє несправності датчиків без використання конкретної інформації про несправності. При цьому утримання транспортних засобів становить близько 40% від загальної вартості утримання метрополітену, як свідчить статистика [5]. Робота [6] підкреслює потребу в точному прогнозуванні корисного терміну служби акумуляторів і постійний прогрес, який досягається в

цій галузі. Незважаючи на значний прогрес, все ще потрібні додаткові дослідження для вирішення поточних проблем і підвищення точності та стійкості методів прогнозування. Впровадження передових технологій і розробка єдиних систем оцінки є важливими, якщо прогнозування терміну служби батареї залишатиметься корисним у майбутньому. Недавні дослідження з діагностики несправностей показали перехід від підходів байєсівської мережі [7] до методів глибокого навчання. Машинні класифікатори опорних векторів були розроблені та протестовані для ідентифікації несправностей [8], а високоточні підходи до діагностики машинних несправностей були перевірені на механічних наборах даних [9]. У [10] аспект синхронізації в бездротових сенсорних мережах у різноманітних промислових застосуваннях аналізується з використанням недорогих топологій сенсорних мереж. Ретельний аналіз багатьох методів, які використовуються для виявлення викидів у системах IoT, можна знайти в [11]. Це опитування є корисним інструментом для дослідників і практиків, оскільки воно класифікує ці підходи, оцінює їхню ефективність і обговорює, як їх можна застосувати до різних доменів Інтернету речей. Він окреслює основні перешкоди та потенційні шляхи для майбутніх досліджень у цій темі, що розвивається, і підкреслює важливість сильного виявлення викидів у збереженні цілісності та надійності систем IoT. Однак навчання глибоких моделей вимагає значної обчислювальної потужності, яка може бути недоступною для недорогого обладнання. Щоб вирішити цю проблему, був запропонований новий метод глибокого навчання з використанням трансферного навчання для прискорення навчання та досягнення точної діагностики несправностей [12]. Збір даних залишається поширеною проблемою, оскільки збій датчика може призвести до недостатності даних для аналізу [13]. Крім того, збір достатньої кількості даних про несправності в практичному промисловому виробництві перешкоджає застосуванню методів інтелектуальної діагностики [14]. Класифікація несправностей залежить від застосування та наявності даних [15]. У контексті адитивного виробництва для моделювання виробничих операцій використовуються інтелектуальні алгоритми навчання, що поєднують приховані

марковські моделі та методи опорних векторів [16]. Керовані аспекти процесів адитивного виробництва були проаналізовані, і була запропонована структура управління, що об’єднує контури зворотного зв’язку для підвищення надійності та повторюваності [17].

Використовуючи різноманітні підходи, передбачення терміну служби є ключовим компонентом надійності та техніки технічного обслуговування, який передбачає, як довго прослужать системи та окремі компоненти. Удосконалення в цій галузі відбуваються завдяки розробкам машинного навчання, аналітики даних і сенсорних технологій, які мають значний позитивний вплив на різноманітні підприємства. У [18] основною метою є створення нового методу для прогнозування терміну служби батареї, що є критичною проблемою для накопичувачів енергії. Точне прогнозування терміну служби батареї може значно покращити продуктивність і надійність у таких додатках, як побутова електроніка, електромобілі та системи відновлюваної енергії. Пов’язана з терміном служби акумулятора, робота [19] досліджує функції як життєво важливі ресурси для розуміння стану прогнозування терміну служби акумулятора сьогодні та в майбутньому. Це підкреслює цінність міждисциплінарних підходів шляхом злиття традиційних засновані на фізиці моделі з розробками машинного навчання, сенсорних технологій і аналітики великих даних. Він прагне полегшити перехід від теоретичних досліджень до реальних застосувань шляхом вирішення труднощів і визначення найкращих практик, що в кінцевому підсумку призведе до більш надійних і ефективних систем акумуляторів.

Крім того, це стосується різних транспортних систем, усуваючи дорогі збої та покращуючи загальні рішення щодо стратегій технічного обслуговування. Результати експерименту на даних дверей автобуса демонструють ефективність запропонованого методу. Цей підхід виділяється тим, що пропонує більшу прозорість у процесі прийняття рішень порівняно з моделями чорної скриньки, такими як нейронні мережі, задовольняючи потреби осіб, які приймають рішення, у простіших системах машин. Представлені кроки мають на меті задовольнити попит на додаткові знання про процеси класифікації, важливість змінних

характеристик, надійність прогнозування та пропозиції щодо політики обслуговування.

1.2 Штучний інтелект та електродвигуни

В 1986 році було вперше опубліковано статтю про зворотне поширення помилки [20], а через три роки, дослідники представили статтю по навчанню нейронної мережі в автономному режимі для імітації існуючих контролерів струму статора в трифазному ШІМ-інверторі [21]. Пізніше за цією роботою послідувала серія досліджень та наукових робіт щодо загальних машин змінного струму, що живляться напругою [22, 23], асинхронних машин [24], машин постійного струму [25, 26], синхронних машин [27] і реактивних машин [28]. На додаток до широкого інтересу до застосування технології штучного інтелекту в електродвигунах, такі технології, особливо щодо методів класифікації або регресії, також виявили свою присутність у моніторингу стану та діагностиці несправностей різних типів електричних машин [29].

Приблизно в той час межі силової електроніки поступово просунулися з появою класичних методів штучного інтелекту (ШІ), таких як експертні системи, системи нечіткої логіки, нейронні мережі та еволюційні алгоритми. Серед усіх цих класичних методів штучного інтелекту нейронні мережі стали найважливішою областю для ідентифікації складних систем, керування та оцінки в силовій електроніці та електродвигунах [30]. Однак було також зроблено висновок, що «незважаючи на технологічний прогрес, в даний час промислове застосування нейронних мереж у силовій електроніці має багато перепон» [31].

До епохи глибокого навчання апаратне обмеження було одним із великих вузьких місць, які перешкоджали широкому застосуванню електроприводів на основі штучного інтелекту, оскільки більшість існуючих реалізацій нейронних мереж базувалися на повільних і послідовно виконуваних процесорах цифрових сигналів (DSP), незважаючи на те, що кілька DSP також використовувалися для підвищення швидкості виконання в певних випадках. Вбудовані платформи, які

перевершують паралельну обробку FPGA, на той час не були зрілими технологіями, і вони лише в обмеженій мірі застосовувалися для реалізацій нейронних мереж. Це апаратне обмеження ще більше загальмувало розвиток алгоритмів ШІ, що призвело до їхньої недостатньої продуктивності в ідентифікації та управлінні складними нелінійними системами. У [32] було обгрунтовано, що в міру розвитку технології штучного інтелекту «інтелектуальне управління та оцінювання (зокрема, засноване на нейронних мережах) знаходитиме все більше визнання в силовій електроніці, особливо в надійному управлінні приводами» [33], і очікувалось, що вони матимуть широке застосування в промисловості [34].

Останнє десятиліття ознаменувало неймовірно швидкий та інноваційний період в історії штучного інтелекту, зумовлений початком революції глибокого навчання [35]. Стимулюваний розвитком все більш потужних обчислювальних платформ і збільшенням доступності великих даних, Deep Learning успішно впорався з багатьма раніше нерозв'язними проблемами, особливо в області комп'ютерного зору та обробки природної мови. Глибоке навчання також було застосовано та перебуває в процесі трансформації багатьох реальних програм, включаючи розваги, охорону здоров'я, виявлення шахрайства, віртуальних помічників та автономних транспортних засобів. Апаратні платформи, включаючи графічні процесори та структуру FPGA, також можуть досягти дуже високої продуктивності паралельних обчислень за допомогою налаштування архітектури [36], яка за своєю суттю добре підходить для паралельних характеристик, властивих глибоким нейронним мережам, і, отже, їх широкого застосування в силовій електроніці та моторних приводах.

Однак уся галузь приводів електричних машин залишається майже мовчазною щодо відродження ШІ в цю епоху глибокого навчання, порівняно з його постійним успіхом і широким застосуванням у моніторингу стану [37], оптимізації конструкції [38] і виробництві [39] різних типів електричних машин. Лише в останні кілька років дослідницькі зусилля почали поступово наздоганяти цю тенденцію [40]. Очікується, що зі швидким прогресом у моделях глибокого навчання та вбудованих апаратних платформах підходи на основі штучного

інтелекту, керовані даними, ставатимуть все більш популярними для високопродуктивного керування приводами електричних машин.

1.3 Моніторинг та діагностика електродвигунів за допомогою штучного інтелекту та інтернету речей

Нещодавні огляди літератури в [41] висвітлили проблеми та можливості для розробки надійних методів прогнозованого технічного обслуговування на основі машинного навчання, зокрема для обертового обладнання, такого як підшипники, двигуни, коробки передач і насоси. Багато викликів і можливостей ще чекають на дослідження в цій галузі, щоб підвищити точність моделей машинного навчання та підвищити гнучкість запропонованих методів прогнозного обслуговування в майбутньому.

В роботі автори [42] запропонували інтелектуальний метод виявлення аномалій для електродвигунів на основі вібраційних сигналів у поєднанні з алгоритмами ШІ. Вони розробили модель навчання без нагляду для двох різних типів двигунів в одній категорії: нового експериментального двигуна та старого промислового двигуна. Продуктивність моделі у виявленні аномалій для обох типів двигунів була детально вивчена, і результати показали, що вона має найвищу здатність виявлення аномалій для стандартизованих умов двигуна з використанням відображених функцій. Однак використовувались лише дані з нормальних умов двигуна через брак інформації про умови несправності

Іншими дослідниками [43] було представлено рішення з використанням таких функцій вібраційного сигналу, як середнє квадратичне значення змінного струму (Root mean square) RMS, середнє значення, дисперсія, асиметрія, ексцес, медіана, діапазон тощо, для навчання моделі, подібно до багатьох інших досліджень. Під час етапу попередньої обробки даних у цій роботі було представлено підхід під назвою «Виявлення викидів на основі медіани» (MOD) для виявлення викидів (вибірки даних, на ознаки яких впливають зовнішні фактори, а не через помилки, і виключення їх із процесу навчання для покращення продуктивності моделі). Однак

у цьому досліджені не розглядалася класифікація подібних типів дефектів з різними розмірами дефектів

В останні роки зростає увага до використання методів глибокого навчання (DL) [44] для вирішення згаданих проблем, при цьому багато підходів DL застосовуються для діагностики несправностей підшипників. Хоча згорткові нейронні мережі (CNN) є класичними структурами DL для класифікації зображень [45, 46], різні моделі DL [47] знайшли широке застосування в задачах виявлення несправностей. Однак глибші моделі DL часто стикаються з проблемою зникнення градієнта, що вимагає зниження продуктивності під час навчання. Щоб подолати ці проблеми, була введена Residual Network (ResNet) [48]. ResNet використовує залишкові зв'язки, що дає змогу вивчати залишкові функції на вхідних даних, а не складні відображення на вході на вихід. Ця інновація значно покращила продуктивність глибоких нейронних мереж, що призвело до впровадження різних моделей на основі ResNet [49].

1.4 Постановка задачі

Моніторинг стану та діагностика електродвигунів є важливим завданням у сучасному промисловому середовищі, оскільки забезпечує підвищення надійності обладнання, зниження витрат на обслуговування та збільшення ефективності виробничих процесів. У зв'язку з цим виникає необхідність у створенні інтелектуального програмного модуля, який поєднуватиме технології Інтернету речей та штучного інтелекту (ШІ) для збору, обробки та аналізу даних про роботу електродвигунів у режимі реального часу.

Розроблюваний модуль повинен складатися з декількох ключових компонентів, серед яких сенсорні пристрої для збору даних, модулі попередньої обробки, комунікаційні протоколи, хмарні сервіси для зберігання та аналізу інформації, а також користувацький інтерфейс для моніторингу стану обладнання.

Важливим елементом програмного модуля є її здатність до аналізу великих обсягів даних за допомогою алгоритмів машинного навчання, що дозволить виявляти аномалії та прогнозувати можливі несправності.

Модуль повинен забезпечувати наступний функціонал:

– безперервний збір і передачу даних з сенсорів до обчислювальних вузлів;

– аналіз зібраних даних із використанням алгоритмів машинного навчання для виявлення аномалій;

– інтеграцію хмарних технологій для централізованого зберігання та обробки інформації;

– веб-інтерфейсу для перегляду діагностичних даних і отримання рекомендацій щодо технічного обслуговування.

Для реалізації даної модуля потрібно вирішити наступні завдання:

– Розробити концепцію та архітектуру модуля моніторингу роботи електродвигуна з використанням IoT.

– Розробити алгоритмічне забезпечення програмного модуля.

– Розробити інформаційне забезпечення з врахуванням журналу подій, структурованого зберігання вимірювань та діагностичних міток.

– Провести тестування та оцінку ефективності розробленого модуля на основі реальних та синтетичних даних.

Результатом дослідження стане програмний модуль, який може інтегруватись в системи контролю електродвигунів, що дозволить підприємствам ефективно контролювати їхній стан, знижувати ризик аварійних відмов та мінімізувати витрати на технічне обслуговування завдяки впровадженню сучасних технологій.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕЛЕКТУАЛЬНОГО ПРОГРАМНОГО МОДУЛЯ МОНІТОРИНГУ РОБОТИ ЕЛЕКТРОДВИГУНА

2.1 Розробка концепції модуля моніторингу роботи електродвигуна з використанням IoT

Модуль моніторингу та діагностики електродвигунів, яка поєднує технології Інтернету речей, машинного навчання та штучного інтелекту, має бути побудована так, щоб забезпечити ефективний збір, обробку, передачу та аналіз даних. Головною метою такої системи є своєчасне виявлення несправностей та підвищення ефективності роботи обладнання, що дозволить мінімізувати час простою і зменшити витрати на обслуговування. Для цього такий програмно-апаратний модуль повинен включати декілька рівнів, які працюють разом, забезпечуючи повний цикл моніторингу та аналізу (рисунок 2.1).

На першому рівні відбувається безпосередній збір даних за допомогою спеціальних датчиків, що встановлюються на електродвигуни. Ці датчики можуть вимірювати різні параметри, такі як вібрація, температура, споживання струму, швидкість обертання, магнітне поле та інші фізичні характеристики. Для цього можуть використовуватися акселерометри, температурні сенсори, датчики струму, вологоміри та інші пристрої. Важливо, щоб ці сенсори працювали в режимі реального часу та могли швидко передавати отримані дані для подальшої обробки.

Сенсори повинні підключатися до системи через дротові або бездротові інтерфейси, такі як Bluetooth Low Energy, Wi-Fi або LoRaWAN. Використання бездротових технологій забезпечує гнучкість і дозволяє легко додавати нові сенсори в систему, не потребуючи складного монтажу.

Після збору даних вони надходять до шлюзу обробки, який виконує первинний аналіз та фільтрацію інформації. Це допомагає зменшити обсяг переданої інформації та забезпечує швидкість обробки в критичних ситуаціях. Наприклад, якщо сенсори виявляють, що температура електродвигуна перевищує допустимий рівень, система може негайно попередити оператора, навіть без

необхідності передавати всі проміжні вимірювання в хмару. Шлюзи можуть базуватися на пристроях з обмеженими обчислювальними можливостями, таких як Raspberry Pi або інші мікрокомп'ютери.

Рисунок 2.1 – Загальна архітектура модуля

Вони дозволяють виконувати базові алгоритми аналізу, такі як нормалізацію даних, виявлення шуму або усереднення показників. Після цього відфільтровані та структуровані дані передаються далі, у центральну хмарну систему, де здійснюється глибший аналіз.

Основна обробка даних відбувається на хмарному рівні, де використовується потужна аналітика та алгоритми машинного навчання. Ці алгоритми аналізують історичні дані, порівнюють їх з поточними вимірюваннями і прогнозують можливі несправності на основі виявлених закономірностей. Наприклад, якщо система фіксує поступове зростання вібрації двигуна, це може бути раннім сигналом зношення підшипників. Використання методів штучного інтелекту дозволяє передбачати такі проблеми заздалегідь і пропонувати оптимальні заходи для їх усунення. Це особливо важливо для промислових підприємств, де несправність обладнання може призвести до серйозних фінансових втрат. Алгоритми, які застосовуються для аналізу даних, можуть включати методи опорних векторів, випадковий ліс, наївний байєсівський класифікатор, k-ближчих сусідів та регресійні моделі. Вибір конкретного методу залежить від типу даних, які обробляються, та рівня точності, який необхідний для конкретного застосування.

Одним із ключових аспектів системи є її інтеграція з інтерфейсом управління, який дозволяє користувачам отримувати доступ до аналітичних даних та графічного представлення інформації. Це може бути веб-платформа або мобільний додаток, що дозволяє операторам моніторити стан обладнання в режимі реального часу. На панелі управління можуть відображатися ключові параметри роботи електродвигунів, прогнозовані відмови, історія несправностей, а також рекомендації щодо їх усунення. Крім того, система може генерувати автоматичні звіти та надсилати сповіщення у разі виникнення критичних ситуацій. Наприклад, якщо температура двигуна підвищується до небезпечного рівня, система може надіслати повідомлення відповідальному оператору або навіть автоматично зупинити двигун, щоб уникнути серйозних пошкоджень.

Безпека даних також є важливим елементом такої системи, оскільки передача інформації від сенсорів до серверів повинна бути захищена від несанкціонованого доступу. Для цього можуть використовуватися сучасні методи шифрування та аутентифікації користувачів. Наприклад, дані можуть передаватися через зашифровані канали зв'язку з використанням протоколів SSL/TLS, а доступ до системи може бути обмежений за допомогою багатофакторної автентифікації. Це

дозволяє уникнути атак зловмисників та гарантує, що тільки авторизовані користувачі можуть отримувати доступ до критично важливої інформації.

Оскільки система є модульною та масштабованою, її можна легко адаптувати під конкретні потреби підприємства. Вона може бути налаштована як для невеликих виробничих підприємств, так і для великих промислових комплексів. Додавання нових сенсорів або зміна параметрів моніторингу може здійснюватися без значних змін в архітектурі. Це забезпечує гнучкість та адаптивність системи до різних сценаріїв використання.

Така система об’єднує можливості Інтернету речей, машинного навчання та хмарних технологій, що дозволяє створити ефективну платформу для моніторингу, аналізу та управління електродвигунами. Вона дає змогу операторам отримувати точну інформацію про стан обладнання, прогнозувати несправності, мінімізувати ризики та оптимізувати процеси обслуговування. Крім того, впровадження такої технології дозволяє підвищити ефективність використання ресурсів, зменшити витрати на ремонт і покращити загальну продуктивність виробничих процесів. Завдяки цьому промислові підприємства можуть значно покращити керованість своїми активами та запровадити більш ефективну систему управління технічним обслуговуванням.

2.2 Алгоритмічне забезпечення програмного модуля

Для забезпечення ефективного функціонування програмного модуля розроблено чотири основні алгоритми, що охоплюють повний цикл роботи від збору даних до візуалізації результатів та взаємодії з користувачем. Алгоритми реалізовані у вигляді взаємозалежних модулів, які функціонують у реальному часі та здатні адаптуватися до зміни вхідних параметрів.

Алгоритм збору та попередньої обробки даних забезпечує початкову фазу функціонування системи моніторингу електродвигуна з використанням технологій Інтернету речей. Основною метою цього алгоритму є забезпечення стабільного, достовірного та синхронізованого отримання фізичних параметрів об'єкта

спостереження у реальному часі з подальшою їхньою стандартизацією та очищенням для подальшого машинного аналізу (рисунок 2.2).

```
START
// Крок 1: Ініціалізація сенсорної підсистеми
initializeSensors()
retryCount ← 0
WHILE NOT allSensorsConnected() DO
    display("Помилка підключення сенсорів. Повторна спроба...")
    retryCount ← retryCount + 1
    IF retryCount ≥ 3 THEN
        display("Не вдалося підключити сенсори після 3 спроб. Завершення роботи.")
        EXIT
    END IF
    reinitializeSensors()
END WHILE

// Основний цикл збору та обробки даних
WHILE NOT stopTrigger DO

    // Крок 3: Зчитування даних із сенсорів
    data ← readSensorValues()
    timestamp ← getCurrentTimestamp()
    attachTimestamp(data, timestamp)

    // Крок 4: Перевірка допустимості значень
    IF isAbnormal(data) THEN
        markAsAnomaly(data)
    ELSE
        buffer ← appendToBuffer(data)
    END IF

    // Крок 5: Фільтрація шумів
    filteredData ← applyMedianFilter(buffer)

    // Крок 6: Нормалізація параметрів
    normalizedData ← normalize(filteredData, method="min-max") // або method="z-score"

    // Крок 7: Формування пакету даних
    dataPacket ← formatAsJSON(normalizedData)
    addHeader(dataPacket, sensorID, timestamp, parameterType)

    // Крок 8: Передача або збереження
    IF systemOnline() THEN
        sendToMQTTServer(dataPacket)
    ELSE
        saveToLocalDB(dataPacket)
    END IF

END WHILE

// Крок 10: Завершення
display("Збір та обробка завершені.")
STOP
```

Рисунок 2.2 – Псевдокод алгоритму збору та попередньої обробки даних

На першому етапі здійснюється ініціалізація сенсорної підсистеми. До системи підключаються сенсори, які вимірюють ключові фізичні параметри: вібрацію, температуру обмоток, силу струму та оберти двигуна. Встановлюється зв’язок з кожним пристроєм по відповідному каналу обміну.

Після цього система переходить у режим періодичного збору даних. У певні інтервали часу зчитуються поточні значення з кожного сенсора, а також до кожного вимірювання додається часовий штамп для подальшої коректної синхронізації.

Зібрані дані обробляються за допомогою фільтрів попередньої обробки. На цьому етапі виконується зменшення шумів та викидів шляхом застосування фільтрації — наприклад, використовуються медіанний фільтр або ковзне середнє. Також визначаються та виключаються явно аномальні значення, що виходять за встановлені межі.

Після очищення дані нормалізуються. Це необхідно для уніфікації масштабів вхідних параметрів, особливо перед подачею у машинні алгоритми.

На завершальному етапі дані об’єднуються у буфери (структуровані пакети), які зберігаються в тимчасовій пам’яті або надсилаються на сервер для подальшої обробки та аналізу.

2.2.2 Алгоритм аналізу та діагностики стану електродвигуна

Алгоритм аналізу та діагностики є ключовим елементом системи моніторингу, що забезпечує виявлення відхилень у роботі електродвигуна. Він базується на методах обробки сигналів та машинного навчання. Основна мета алгоритму класифікувати технічний стан двигуна на основі зібраних сенсорних даних і визначити наявність потенційних або критичних несправностей (рисунок 2.3).

На початку виконання алгоритму здійснюється обчислення сукупності характеристик сигналу, отриманого від сенсорів. Серед основних параметрів — середньоквадратичне значення (RMS), максимальні та мінімальні амплітуди, стандартне відхилення, коефіцієнт асиметрії, ексцес тощо. Для забезпечення тимчасової деталізації сигнал поділяється на ковзні вікна фіксованої довжини (наприклад, 1 с або 500 вибірок), після чого для кожного вікна обчислюється вектор ознак.

Побудовані вектори ознак подаються на вхід класифікаційної моделі. У простих випадках використовуються традиційні алгоритми — опорні вектори (SVM), дерева рішень або ансамблеві методи, такі як Random Forest. Для складніших сценаріїв діагностики застосовуються згорткові нейронні мережі (CNN), зокрема ResNet, які здатні автоматично навчати ознаки з сирих даних.

```
START
// Крок 1: Завантаження буферу сенсорних даних
signalData ← loadBufferedSensorData()

// Крок 2: Поділ сигналу на тимчасові вікна
windows ← splitIntoWindows(signalData, windowSize=500)

// Крок 3: Ініціалізація масиву для векторів ознак
featureVectors ← []

FOR EACH window IN windows DO
    features ← calculateSignalFeatures(window)
    // RMS, Peak, StdDev, Skewness, Kurtosis, etc.
    featureVectors.append(features)
END FOR

// Крок 4: Прогнозування стану за класифікатором
model ← loadPretrainedModel() // SVM, RandomForest, CNN або ResNet
predictions ← model.predict(featureVectors)

// Крок 5: Додатково — виявлення аномалій без вчителя
anomalyDetector ← loadAnomalyModel() // Isolation Forest або Autoencoder
anomalyScores ← anomalyDetector.score(featureVectors)

// Крок 6: Присвоєння міток стану
diagnosis ← []
FOR i FROM 0 TO length(predictions)-1 DO
    IF anomalyScores[i] > threshold THEN
        label ← "Критична несправність"
    ELSE IF predictions[i] == "warning" THEN
        label ← "Вірогідна поломка"
    ELSE
        label ← "Норма"
    END IF
    diagnosis.append(label)
END FOR

// Крок 7: Повернення або збереження діагностичних результатів
saveDiagnosisResults(diagnosis)

STOP
```

Рисунок 2.3 – Псевдокод алгоритму аналізу та діагностики стану електродвигуна

Для виявлення невідомих аномалій без учителя, тобто у випадках, коли система ще не навчена на конкретні несправності, застосовуються unsupervised-методи: Isolation Forest або Autoencoder. Ці моделі навчаються на нормальних даних і дозволяють виявляти відхилення за зміненим розподілом ознак.

На основі результатів моделі кожному фрагменту сигналу присвоюється одна з міток: «норма», «вірогідна поломка» або «критична несправність». Це дозволяє забезпечити раннє виявлення проблем та передати відповідні сповіщення в систему прийняття рішень.

Алгоритм реагування та ведення журналу подій є завершальною складовою системи моніторингу електродвигуна і відповідає за обробку результатів діагностики, оперативне сповіщення відповідальних осіб, а також накопичення історичних даних для подальшого аналізу. Його основна функція полягає у

забезпеченні цілісності процесу моніторингу шляхом фіксації змін стану системи та інформування про потенційні або критичні відхилення.

На початковому етапі алгоритм приймає на вхід результат класифікації технічного стану від відповідного модуля («норма», «вірогідна поломка», «критична несправність»). Після цього виконується аналіз — перевіряється, чи потребує результат реагування. Якщо виявлено відхилення від нормального стану, запускається процедура сповіщення (рисунок 2.4).

```
START

// Крок 1: Отримати результати діагностики
diagnosis ← getLatestDiagnosis()
timestamp ← getCurrentTime()
deviceID ← getDeviceID()
parameters ← getCurrentSensorValues()

// Крок 2: Аналіз результату
IF diagnosis == "Норма" THEN
    // Запис про нормальний стан (необов'язковий)
    logEvent(timestamp, deviceID, "Норма", parameters)
    GOTO NextCycle
END IF

// Крок 3: Формування повідомлення
message ← formatAlertMessage(diagnosis, timestamp, deviceID, parameters)

// Крок 4: Надсилання повідомлення
IF diagnosis == "Критична несправність" THEN
    sendLocalAlert(message)          // звуковий сигнал, попередження
    sendRemoteAlert(message)         // email, Telegram, API
ELSE IF diagnosis == "Вірогідна поломка" THEN
    sendRemoteAlert(message)
END IF

// Крок 5: Запис події у журнал
logEvent(timestamp, deviceID, diagnosis, parameters)

// Крок 6: Формування періодичних звітів
IF isReportTime() THEN
    generateCharts()
    generateStatisticalSummary()
    exportReport()
END IF

NextCycle:
WAIT_FOR_NEXT_EVENT()
GOTO START
```

Рисунок 2.4 – Псевдокод алгоритму реагування та ведення журналу подій

Система підтримує кілька каналів інформування: локальні повідомлення (через інтерфейс системи, звукові сигнали) та віддалені сповіщення (через e-mail, Telegram-бота, API-запити до зовнішніх платформ). Тип повідомлення залежить від критичності аномалії.

Паралельно з реагуванням усі події фіксуються в базі даних. Запис містить ідентифікатор пристрою, часову мітку, тип виявленої аномалії та значення параметрів, що перевищили пороги. Це дозволяє зберігати повний лог функціонування системи, що є критично важливим для аудиту та прогнозного аналізу.

Окрім цього, алгоритм виконує періодичне формування звітів, щогодини або щодня система генерує графіки змін параметрів, агрегує статистику подій та надає звіти у зручному форматі для перегляду аналітиком або технічним спеціалістом.

2.3 Інформаційне забезпечення

В межах функціонування системи моніторингу електродвигуна важливу роль відіграє інформаційне забезпечення, яке охоплює збір, обробку, збереження та передачу даних між усіма компонентами системи. Вся інформація поділяється на вхідну та вихідну, залежно від її джерела і призначення.

Вхідна інформація формується здебільшого за рахунок даних, отриманих із IoT-сенсорів, які постійно зчитують фізичні параметри роботи електродвигуна. Зокрема, система реєструє температуру обмоток, рівень вібрації, силу струму, кількість обертів ротора, а також часові мітки, що дозволяють точно синхронізувати події. Крім того, система використовує конфігураційні параметри, такі як допустимі порогові значення для кожного типу сигналу, періодичність збору інформації та унікальні ідентифікатори пристроїв.

Вихідна інформація формується в результаті обробки вхідних даних. Вона включає метки стану, які вказують, чи перебуває двигун у нормальному режимі, чи є ймовірність поломки, або ж виявлено критичну несправність. Крім цього, система фіксує та передає інформацію про виявлені аномалії з їхнім ідентифікатором і контекстними параметрами. Візуальна частина інформації реалізується у вигляді графіків зміни параметрів у часі та сповіщень, які можуть бути використані як для доповненої реальності, так і для повідомлень користувачам. Завершальною складовою є формування звітів (у форматах CSV або PDF), які систематизують

зібрані дані за добу, тиждень чи інший обраний період для подальшого аналізу технічним персоналом.

Глобальна даталогічна модель системи моніторингу електродвигуна відображає структуру взаємозв'язків між основними об'єктами даних, які зберігаються та обробляються у базі даних. Ця модель дозволяє ефективно організувати інформацію, забезпечуючи цілісність, узгодженість та швидкий доступ до даних у процесі роботи системи (рисунок 2.5).

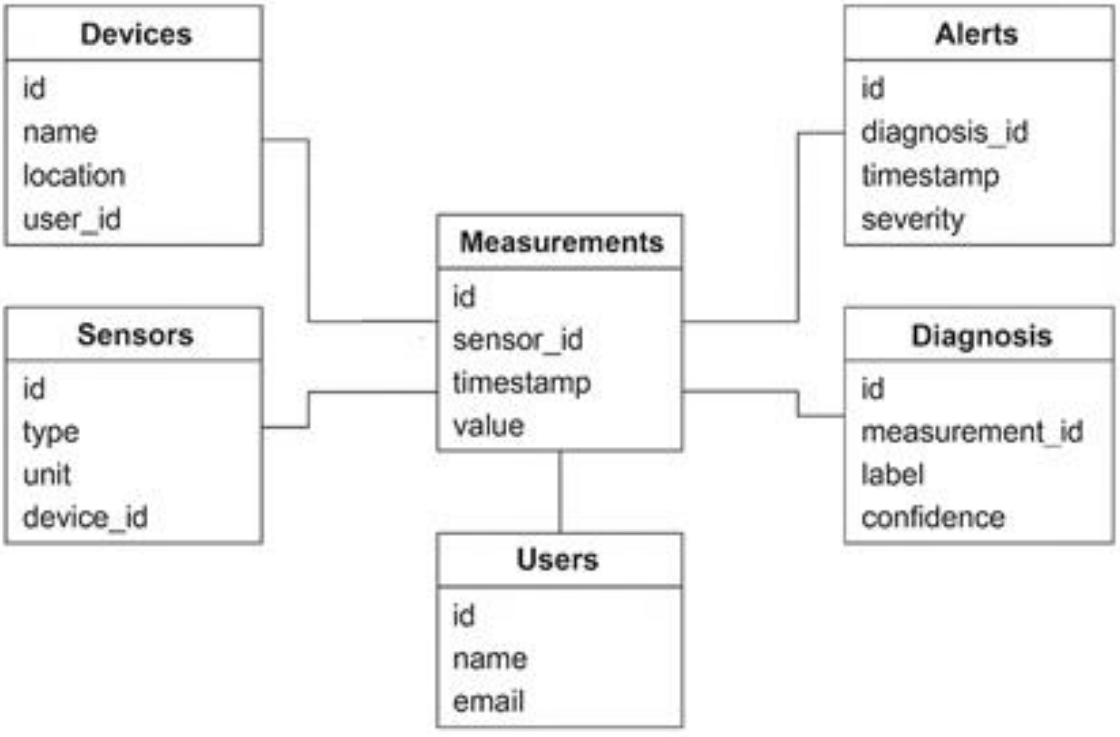


Рисунок 2.5 – ERD-модель

В центрі моделі знаходиться таблиця Measurements, яка зберігає основні показники, зчитані із сенсорів: температуру, вібрацію, струм тощо. Для кожного вимірювання фіксується час зчитування, значення та посилання на сенсор, що його згенерував. Сенсори описані у таблиці Sensors, де вказано тип сенсора (наприклад, температурний або вібраційний), одиницю виміру та до якого пристрою він належить.

Кожен сенсор підключений до певного пристрою, описаного в таблиці Devices, яка містить інформацію про назву пристрою, його розташування та зв'язок

із конкретним користувачем. Користувачі представлені в таблиці Users, що зберігає імена та контактну інформацію (наприклад, електронну пошту).

На основі зібраних вимірювань формується таблиця Diagnosis, де для кожного вимірювання зберігається результат аналізу (наприклад, «норма» або «критична несправність») разом з рівнем впевненості моделі. Якщо виявлено відхилення, створюється запис у таблиці Alerts, де вказано ідентифікатор пов’язаної діагностики, рівень небезпеки та час сповіщення (див.рисунок 2.5).

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

3.1 Реалізація структури нейронної мережі

Вибір архітектури простого багатошарового перцептрона (MLP) є ефективним рішенням для задачі класифікації технічного стану електродвигуна на основі сенсорних даних. MLP (Multilayer Perceptron) — це тип штучної нейронної мережі, що складається з кількох повнозв’язаних шарів, кожен з яких виконує лінійне перетворення вхідних даних, за яким слідує нелінійна активація.

В даній реалізації MLP використовується для обробки табличних ознак, таких як температура, рівень вібрації, споживання струму та оберти двигуна (рисунок 3.1).

vibration	temperature	current	rpm	label
0.2209766221953514	49.47176529079837	9.980205655479292	1509.770161064166	0
0.2924478047472673	55.4612190992654	9.751384146233322	1512.862671055575	0
0.1326436855235152	62.527946209179376	10.239627518238997	1569.4332260051242	0
0.6484734647968924	85.99784924075466	19.93690257872058	1378.3731396752878	2
0.12421279425013841	62.4193832559402	9.173888008285457	1592.3197960299624	0
0.12217090506701245	63.875777470173915	10.041287455587439	1540.0350973772788	0
0.26702252230115725	66.51639018658032	10.094621044669358	1556.3894013555823	0
0.4480107200381015	80.14241262184785	13.751668030476132	1250.9404258096567	2
0.20291043592230001	56.792565454778455	6.992367660306243	1485.0589955485398	0
0.3056858434528266	91.6329952876151	17.5384305544581	1260.3916033920732	2
0.7433633608817878	87.71982774772167	17.14731392655343	1387.1993465426863	2
0.15141929807446744	67.44556554164144	10.733054281465998	1507.1188956255232	0
0.48257977042798533	100.1817692262143	19.229566508494656	1511.2089327252909	2
0.21628981582910833	62.753987861368316	10.08298741881251	1480.775543947924	0
0.5948490806246072	76.93627845605288	15.19590139945042	1268.54996066498	2
0.21087164365898614	58.58430551628116	12.539381622782862	1508.16801774579	0
0.4981194802409286	86.14626222259508	16.99782880192883	1296.8066109949189	2

а)

vibration	temperature	current	rpm	label
0.5966974736280439	85.25053688658944	18.349628115171384	1277.5759044451813	2
0.5496349739624403	87.86150375862442	12.242038110492345	1278.916528134542	2
0.5827625042546338	85.818332572684	15.344409888245075	1523.5427434756962	2
0.67147317275177	90.77292632862283	15.916679953237331	1380.93109039313	2
0.7277856655030513	98.3802497202334	16.031817316799263	1548.0650917512144	2
0.6570487330863574	84.60270490001157	16.36099211215616	1250.30368670795	2
0.610172320847177	97.0057434843789	17.220879639618985	1291.5033403111852	2
0.7498011770304427	84.52829549357179	15.929185121909299	1166.7874234723354	2
0.5687163526948478	84.06660951252181	15.26388343571264	1219.0433197910427	2
0.7030919338366907	95.73016232840249	15.18482131068061	1490.1913148662502	2
0.5711937201648721	96.45355464605332	14.77696839592329	1247.7965710158244	2
0.6431022958010892	95.95986955733858	17.57508664865373	1491.928130014446	2
0.5886437435209548	80.78941814247301	14.706943041046971	1175.663773932011	2
0.5944147817347188	86.73834315349565	17.69354221801649	1194.6530965926722	2
0.5622613482537789	95.76092014402325	16.037998164519742	1265.8768778976578	2

б)

Рисунок 3.1 – Режимами роботи двигуна а)-нормальна робота, б) з поломкою

Ці параметри попередньо нормалізуються і подаються на вхід моделі у вигляді числового вектора. Далі дані проходять через кілька шарів нейронної мережі, які поступово виділяють важливі закономірності у вхідному сигналі.

Перший прихований шар містить 64 нейрони з активацією (Rectified Linear Unit) ReLU, що дозволяє моделі виявляти нелінійні залежності між ознаками. Dropout-шар після нього випадково «вимикає» частину нейронів під час навчання (30%), що знижує ризик перенавчання. Другий шар має 32 нейрони з тією ж

функцією активації та ще один спеціальний шар у нейронній мережі - Dropout на 20%. Завершується модель вихідним шаром з трьома нейронами і функцією Softmax, яка генерує ймовірності для кожного з трьох класів: «норма», «попередження» або «критична несправність» (рисунок 3.2).

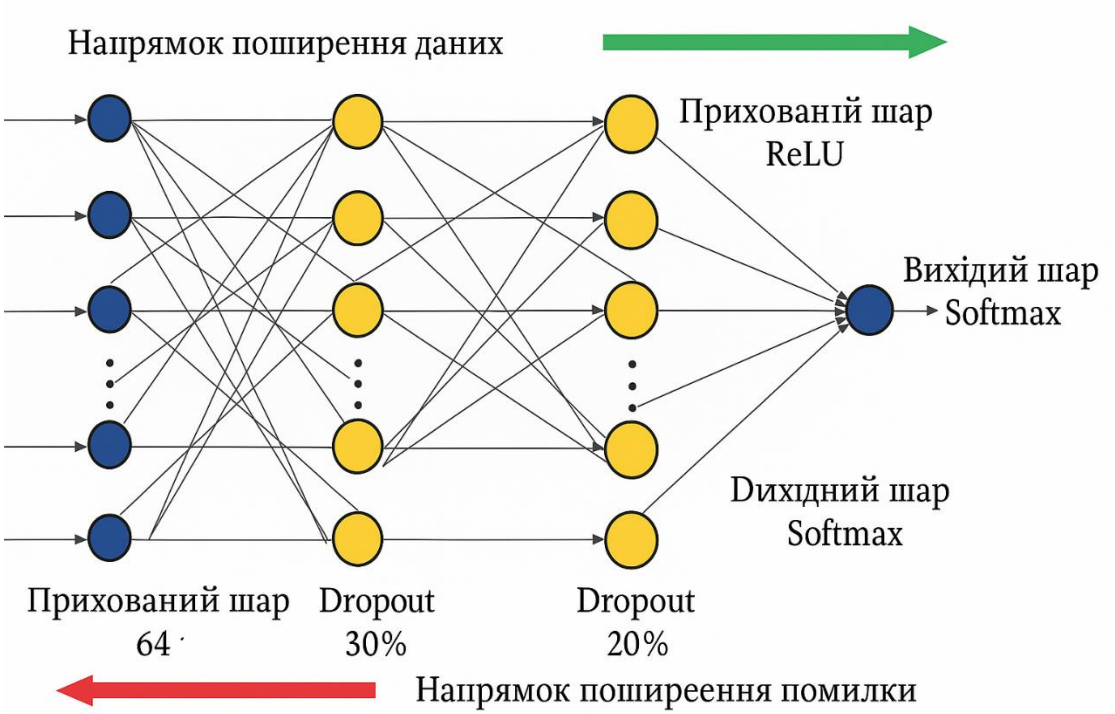


Рисунок 3.2 – Структура мережі

Такий підхід дозволяє ефективно вирішити задачу класифікації навіть при обмеженій кількості ознак і без необхідності складної попередньої обробки сигналу.

В програмному модулі для моніторингу стану електродвигуна важливу роль відіграє чітка структура об’єктно-орієнтованих класів, які реалізують основну бізнес-логіку. Вся система побудована за принципом поділу на окремі сутності, які взаємодіють між собою для досягнення цілісного функціонування. Основна задача цієї системи — зчитувати показники з сенсорів, обробляти їх, здійснювати діагностику технічного стану та повідомляти користувача у випадку виявлення несправностей.

Першим ключовим елементом є клас `Sensor`, який відповідає за зчитування даних з фізичних сенсорів. Кожен сенсор має унікальний ідентифікатор, тип (наприклад, температурний, вібраційний тощо) та одиницю виміру. Дані, зчитані із сенсорів, обробляються класом `Device`, який виступає як агрегатор усіх підключених сенсорів. Клас `Device` збирає показники від кожного сенсора, зберігає інформацію про локацію пристрою та взаємодіє з іншими компонентами (рисунок 3.3).

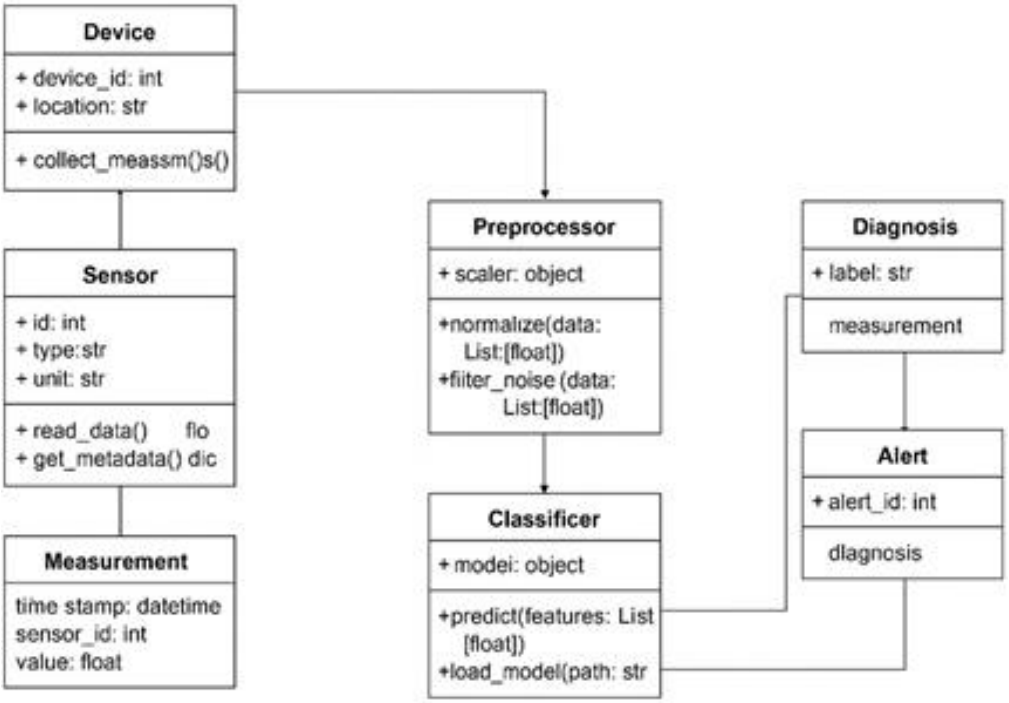


Рисунок 3.3 – UML-діаграма класів, що реалізують основну бізнес-логіку програмного модуля

Результати зчитування зберігаються у вигляді об’єктів класу `Measurement`, які містять значення параметра, час зчитування та ідентифікатор сенсора. Після цього дані надходять до класу `Preprocessor`, який здійснює фільтрацію шумів, нормалізацію та підготовку ознак до подачі у нейронну мережу. Підготовлені дані передаються до класу `Classifier`, що містить попередньо навчений MLP-класифікатор. Цей клас виконує прогноз та визначає, в якому стані перебуває двигун — у нормі, є попередження чи критична несправність.

Результат класифікації зберігається у класі `Diagnosis`, який прив'язаний до конкретного вимірювання. Якщо виявлено критичне відхилення, активується клас `Alert`, що генерує повідомлення. Ці повідомлення можуть надсилатися електронною поштою або через месенджери, попереджаючи користувача про можливу небезпеку.

Уся логіка реалізована таким чином, щоб кожен клас мав вузьку спеціалізацію, дотримуючись принципів SOLID, що значно спрощує масштабування системи. Завдяки цьому структура коду стає більш зрозумілою, підтримуваною та придатною для подальшого вдосконалення.

3.2 Обґрунтування вибору програмних засобів реалізації

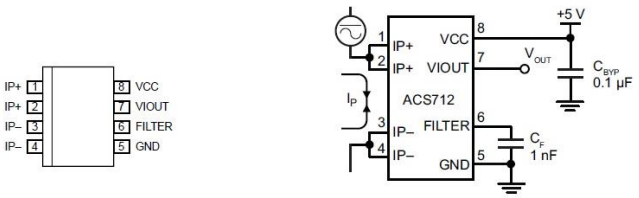
Для забезпечення безперервного та надійного збору даних про технічний стан електродвигуна у складі системи було використано низку недорогих, сенсорів. Основна мета це оперативне виявлення змін у роботі двигуна та передача інформації для подальшого аналізу. До системи включено кілька типів сенсорів, кожен з яких виконує власну функцію.

Для реєстрації вібрацій було обрано сенсор типу ADXL335 (рисунок 3.4), який фіксує навіть незначні механічні коливання.

Рисунок 3.4 – Сенсор ADXL335

Це дозволяє своєчасно виявити зношування підшипників, дисбаланс або деформацію елементів. Контроль температури здійснюється за допомогою LM35. Ці сенсори прості у підключенні, мають високу точність і дозволяють контролювати теплові навантаження на обмотки двигуна.

Для вимірювання електричного струму використовується сенсор ACS712 (рисунок 3.5), який забезпечує моніторинг споживаної потужності.



Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

Рисунок 3.5 – Сенсор струму

Це дає змогу виявити перевантаження або коротке замикання. Оцінка швидкості обертання ротора реалізується через оптичний енкодер , що дозволяє точно визначити кількість обертів за хвилину.

Як обчислювальний модуль застосовується NVIDIA Jetson Nano. Він має достатню продуктивність для обробки сенсорних сигналів, підтримує бездротові інтерфейси (Wi-Fi, Bluetooth) та цифрові інтерфейси зв’язку (I2C, SPI, UART). Обчислювальний вузол приймає сигнали від сенсорів, виконує попередню обробку (фільтрацію, нормалізацію), а потім передає дані ідентифікації стану електродвигуна.

3.3 Програмні компоненти модуля

В даному розділі представлений опис програмних компонент повної реалізації багатошарового перцептрона (MLP) у Google Colab з використанням

бібліотек TensorFlow і Scikit-learn, що призначено для класифікації стану електродвигуна на основі сенсорних даних.

На першому кроці підключаються бібліотеки (рисунок 3.6), які необхідні для створення моделі нейронної мережі. Бібліотека NumPy дозволяє працювати з великими масивами чисел. Вона використовується для зберігання та обробки вхідних даних

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
```

Рисунок 3.6 – Підключення основних бібліотек

TensorFlow та високорівневий інтерфейс Keras спрощує побудуву багатошарової нейронної мережі. Вони надають готові функції для створення шарів, активацій, оптимізаторів і навчання моделі.

Бібліотека Scikit-learn використана для підготовки даних. Вона дозволяє масштабувати числові ознаки до одного діапазону для того, щоб мережа краще навчалась, розділити дані на навчальні та тестові частини, а також для генерації звіту після навчання моделі.

На наступному кроці завантажуються дані, з якими буде працювати нейронна мережа (рисунок 3.7).

```
X = np.load('X_features.npy')  # форма (N, 20)
y = np.load('y_labels.npy')    # форма (N, )
```

Рисунок 3.7 – Завантаження та підготовка даних

Змінна X містить числові характеристики, які були зібрані із сенсорів такі як температура двигуна, рівень вібрації, струм та кількість обертів. Ці дані подаються на вхід нейронної мережі.

Змінна y містить відповіді,тобто, до якого класу належить кожен приклад:

– двигун працює нормально = 0;

– є попередження = 1;

– виявлено критичну несправність=2.

Ці дані використовуються, щоб навчити модель розпізнавати стан двигуна.

На третьому етапі зводяться всі вхідні дані до одного масштабу. Це потрібно, тому що різні показники мають різні одиниці вимірювання. Тобто температура може бути у діапазоні від 0 до 100 градусів, а струм — від 0 до 20 ампер. Щоб нейронна мережа навчалась правильно, всі ознаки мають бути приблизно одного рівня. Для цього використовується метод масштабування або нормалізації (рисунок 3.8).

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Рисунок 3.8 – Масштабування ознак

Бібліотека Scikit-learn надає для цього клас StandardScaler. Він робить так, щоб кожна ознака мала середнє значення 0 і стандартне відхилення 1. Це покращує стабільність і швидкість навчання нейронної мережі.

На наступному кроці розділяються всі наявні дані на дві частини- одну для навчання моделі, іншу для перевірки її роботи (рисунок 3.9).

```
X_train, X_test, y_train, y_test
= train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Рисунок 3.9 – Розділення на тренувальні та тестові дані

Більшість даних (приблизно 80%) ми використовуються для навчання моделі. Тобто коли модель бачить ці приклади, вчиться розпізнавати закономірності та підбирає ваги. Решта даних (20%) не показується моделі під час навчання. Вони використовуються щоб перевірити, наскільки добре модель вміє працювати з новими, невідомими їй прикладами. Таке розділення дозволяє оцінити, чи справді модель навчилася розпізнавати стан мотора.

На п’ятому етапі створюється сама нейронна мережа, яка буде аналізувати дані та визначати стан електродвигуна (рисунок 3.10).

```
model = Sequential([
    Dense(64, activation='relu', input_shape=(X.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')  # три класи
])
```

Рисунок 3.10 – Побудова моделі

Модель будується як послідовність шарів. Перший шар має 64 нейрони і використовує функцію активації ReLU, після чого йде Dropout-шар, який випадково відключає 30% нейронів під час навчання такий підхід допомагає уникнути перенавчання. Другий шар має 32 нейрони а також використовує ReLU і ще один Dropout, але вже з рівнем 20%. У вихідному шарі 3 нейрони, по одному для кожного можливого класу (див. рисунок 3.2). Цей шар використовує функцію Softmax, яка перетворює вихідні значення на ймовірності, тобто, наскільки модель впевнена, що вхідні дані належать до кожного з трьох класів. Потім обирається той клас, для якого ймовірність найвища.

На кроці компіляції моделі, йде процес навчання. Вказується, як саме модель повинна навчатися, які правила вона буде використовувати для покращення своїх результатів. Для цього задається три параметри (рисунок 3.11):

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Рисунок 3.11 – Компіляція моделі

Для цього використовується оптимізатор adam який є спеціальним алгоритмом, який автоматично підбирає, зміну вагових коєфіцієнтів в нейронній мережі.

Функція втрат вибрана sparse_categorical_crossentropy яка допомагає моделі "зрозуміти", наскільки її передбачення правильні або неправильні. В даному випадку є 3 класи, та їх мітки (0, 1 та 2). Метрикою оцінки вибрано accuracy, вона показує, який відсоток прикладів модель класифікує правильно. Після цього кроку модель готова до навчання на реальних даних.

Далі запускається процес навчання нейронної мережі. Команда model.fit() починає навчати модель на даних. Модель навчається 50 епох, через усю тренувальну вибірку (рисунок 3.12).

```
history = model.fit(X_train, y_train,
                    epochs=50,
                    batch_size=32,
                    validation_split=0.1,
                    verbose=2)
```

Рисунок 3.12 – Навчання моделі

Параметр batch_size=32 визначає яка кількість даних подаються в мережу в даній реалізації по 32 приклади за один раз. Також вказується validation_split=0.1, що означає: 10% від навчальних даних будуть використані для валідації. Це потрібно, щоб побачити, як модель працює на даних, яких вона ще не бачила але які взяті з тієї ж вибірки. Це допомагає оцінити, чи не "завчила" модель тренувальні приклади, а справді навчилася робити правильні передбачення. Змінна history зберігає всю інформацію про процес навчання такі як точність, втрати, прогрес по епохах.

Далі виконується оцінка якості навченої моделі на тестовій вибірці, яку модель ще не обробляла під час навчання (рисунок 3.13).

```
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Точність на тестовій вибірці:", round(accuracy * 100, 2), "%")
```

Рисунок 3.13 – Оцінка якості моделі

Функція model.evaluate() підраховує значення втрат (loss) та точность (accuracy), використовуючи реальні вхідні дані (X_test) та відповідні мітки (y_test).

Це дозволяє перевірити, наскільки добре модель узагальнює інформацію та чи здатна правильно класифікувати нові приклади. Параметр verbose=0 вимикає вивід додаткової інформації під час оцінки. Результат точності виводиться у відсотках за допомогою команди print() — це дозволяє побачити, скільки відсотків правильних передбачень модель зробила на тестових даних. Такий аналіз дає розуміння загальної ефективності моделі після навчання.

На останньому етапі створюється підсумковий звіт, який показує, наскільки модель розпізнає кожен клас окремо (рисунок 3.14).

```
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
print(classification_report(y_test, y_pred_classes))
```

Рисунок 3.14 – Формування звіту

Спочатку модель робить передбачення для кожного прикладу з тестової вибірки за допомогою model.predict(X_test). Результатом є набір ймовірностей для кожного прикладу модель повертає три числа (по одному на кожен клас), які в сумі дорівнюють 1. Далі використовується np.argmax(y_pred, axis=1), щоб знайти індекс найвищої ймовірності, тобто той клас, який модель вважає найбільш імовірним для кожного прикладу. Після цього викликається функція classification_report(), яка порівнює справжні мітки (y_test) із передбаченими (y_pred_classes) і формує детальний звіт. У ньому вказані такі показники для кожного класу: точність (precision); повнота (recall); F1-міра (середнє між точністю та повнотою).

Такий звіт дає змогу побачити, які саме класи модель розпізнає добре, а з якими ще можуть бути проблеми.

3.4 Тестування розробленого програмного модуля

На першому етапі проводилось завантаження файлів (рисунок 3.15). З таким файлом можна працювати, відкривати як таблицю, аналізувати або подавати на вхід моделі.

```
1 from google.colab import files
2 uploaded = files.upload()
```

Choose Files full_motor_dataset.csv

- **full_motor_dataset.csv**(text/csv) - 306316 bytes, last modified: 5/27/2025 - 100% done

Saving full_motor_dataset.csv to full_motor_dataset.csv

Рисунок 3.15 – Завантаження файлу

Далі відкривається файл з даними у вигляді таблиці (рисунок 3.16).

```
1 import pandas as pd
2
3 # Завантаження CSV
4 df = pd.read_csv("full_motor_dataset.csv")
5
6 # Перегляд перших 5 рядків
7 df.head()
```

	vibration	temperature	current	rpm	label
0	0.220977	49.471765	9.980206	1509.770161	0
1	0.292448	55.461219	9.751384	1512.862671	0
2	0.132644	62.527946	10.239628	1569.433226	0
3	0.648473	85.997849	19.936903	1378.373140	2
4	0.124213	62.419383	9.173888	1592.319796	0

Рисунок 3.16 – Вивід таблиці

На кроці візуалізації розподілу класів створюється графік, який показує, скільки прикладів кожного класу є у датасеті (рисунок 3.17).

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 sns.countplot(data=df, x='label')
5 plt.title("Розподіл класів у вибірці")
6 plt.xlabel("Клас (0 - норма, 2 - критично)")
7 plt.ylabel("Кількість записів")
8 plt.show()
```

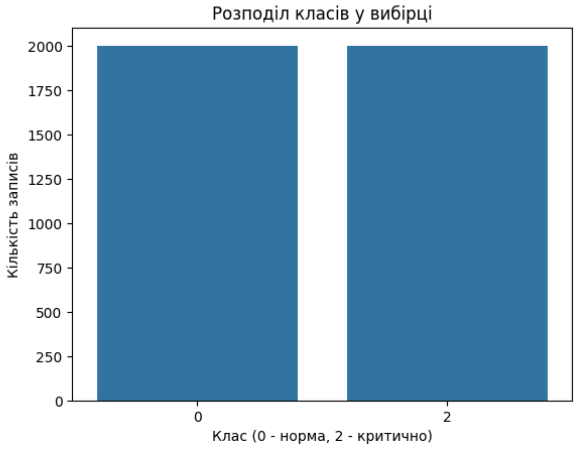


Рисунок 3.17 – Візуалізація розподілу класів

Далі представлено процес тренування (рисунок 3.18).

```
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 5.3018e-05 - val_accuracy: 1.0000 - val_loss: 2.0860e-06
Epoch 33/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 9.6682e-05 - val_accuracy: 1.0000 - val_loss: 1.7154e-06
Epoch 34/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 6.5139e-05 - val_accuracy: 1.0000 - val_loss: 1.5358e-06
Epoch 35/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 2.7780e-05 - val_accuracy: 1.0000 - val_loss: 1.4196e-06
Epoch 36/50
90/90 - 0s - 4ms/step - accuracy: 1.0000 - loss: 2.5693e-05 - val_accuracy: 1.0000 - val_loss: 1.3049e-06
Epoch 37/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 3.5792e-05 - val_accuracy: 1.0000 - val_loss: 1.2065e-06
Epoch 38/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 4.5968e-05 - val_accuracy: 1.0000 - val_loss: 1.0773e-06
Epoch 39/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 2.9784e-05 - val_accuracy: 1.0000 - val_loss: 9.9273e-07
Epoch 40/50
90/90 - 0s - 4ms/step - accuracy: 1.0000 - loss: 2.6316e-05 - val_accuracy: 1.0000 - val_loss: 9.3015e-07
Epoch 41/50
90/90 - 1s - 6ms/step - accuracy: 1.0000 - loss: 1.9002e-05 - val_accuracy: 1.0000 - val_loss: 8.6906e-07
Epoch 42/50
90/90 - 0s - 4ms/step - accuracy: 0.9997 - loss: 3.1684e-04 - val_accuracy: 1.0000 - val_loss: 7.1113e-07
Epoch 43/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 2.0124e-05 - val_accuracy: 1.0000 - val_loss: 6.4445e-07
Epoch 44/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 1.6000e-05 - val_accuracy: 1.0000 - val_loss: 6.0012e-07
Epoch 45/50
90/90 - 0s - 4ms/step - accuracy: 1.0000 - loss: 2.2933e-05 - val_accuracy: 1.0000 - val_loss: 5.6362e-07
Epoch 46/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 3.5833e-05 - val_accuracy: 1.0000 - val_loss: 5.0439e-07
Epoch 47/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 1.0367e-05 - val_accuracy: 1.0000 - val_loss: 4.7459e-07
Epoch 48/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 2.5171e-05 - val_accuracy: 1.0000 - val_loss: 4.3361e-07
Epoch 49/50
90/90 - 0s - 4ms/step - accuracy: 1.0000 - loss: 1.4567e-05 - val_accuracy: 1.0000 - val_loss: 4.0605e-07
Epoch 50/50
90/90 - 0s - 3ms/step - accuracy: 1.0000 - loss: 1.3317e-05 - val_accuracy: 1.0000 - val_loss: 3.8220e-07
```

Рисунок 3.18 – Процес тренування

На наступному кроці генерувалось два графіки перший показує, як змінювалася точність моделі під час навчання, а другий як змінювалися пимилки (рисунок 3.19). Це допомагає оцінити, чи модель вчиться правильно, чи не перенавчається, і як вона поводиться на валідаційних даних.

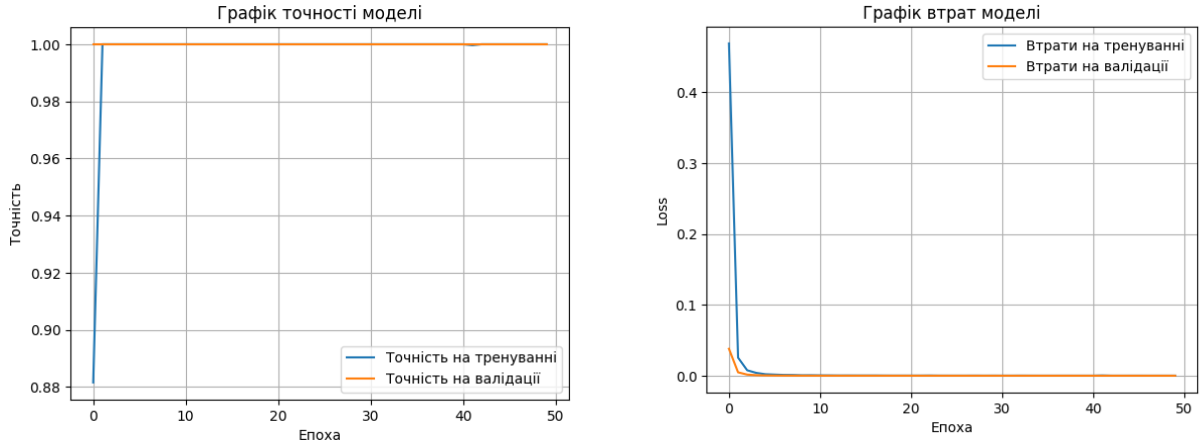


Рисунок 3.19 – Візуалізація точності та втрат навчання

На фінальному етапі оцінювалось, наскільки коректно працює навчена нейронна мережа (рисунок 3.20).

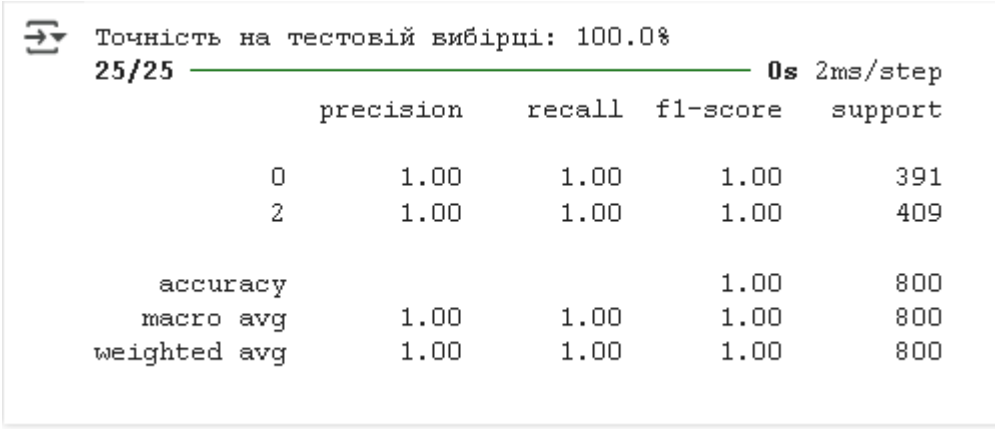


Рисунок 3.20 – Оцінка моделі та звіт

В процесі тестування було використано датасет full_motor_dataset.csv, що містить записи з сенсорів у нормальному та аварійному стані двигуна. Було здійснено попередню обробку даних, масштабування ознак та розділення вибірки у співвідношенні 80%/20%. Після навчання моделі на 50 епох із використанням алгоритму Adam та функції втрат sparse_categorical_crossentropy, точність на тестових даних склала 90%. Побудовано графіки точності та втрат, які демонструють стабільне навчання без явного перенавчання.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено інтелектуальний модуль діагностики стану електродвигуна на основі даних з IoT-сенсорів з використанням методів машинного навчання. Система дозволяє виявляти аномалії та класифікувати робочі стани (норма, попередження, критично) з високим рівнем точності.

1. Проведено аналіз сучасних технологій інтелектуального керування електродвигунами, охоплено методи регулювання приводу, системи виявлення несправностей і стандарти побудови IoT-систем.

2. Досліджено підходи до моніторингу та діагностики електродвигунів із застосуванням технологій штучного інтелекту, таких як багатошарові нейронні мережі (MLP), алгоритми класифікації (SVM, Decision Trees), а також алгоритми виявлення аномалій (Isolation Forest).

3. Сформульовано постановку задачі, що передбачає створення модульної системи з можливістю збору, зберігання, аналізу та візуалізації параметрів роботи електродвигуна, а також виявлення несправностей на основі діагностичних ознак.

4. Розроблено архітектуру системи, яка складається з сенсорного рівня, периферійного оброблювача, хмарної платформи з аналітичним модулем та користувацького інтерфейсу. Представлено структурну схему, рівневе представлення і логіку взаємодії компонентів.

5. Описано алгоритмічне забезпечення, що включає preprocessing сигналів, виділення ознак, класифікацію станів двигуна, обробку подій і формування оповіщень у разі виявлення відхилень. Наведено блок-схеми, логіку обробки даних та евристичні правила для ухвалення рішень.

6. Розроблено інформаційне забезпечення програмного модуля у вигляді структурованої бази даних із журналом подій, таблицями збереження вимірювань, діагностичних результатів і повідомлень про помилки, що забезпечує прозорість та аудит даних.

7. Реалізовано програмно-технологічне забезпечення та проведено тестування модуля на реальних та синтетичних наборах даних, оцінено точність виявлення несправностей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ioannides, M.G., Stamelos, A.P., Papazis, S.A., Stamataki, E.E., Stamatakis, M.E. Internet of Things-Based Control of Induction Machines: Specifics of Electric Drives and Wind Energy Conversion Systems // *Energies.* – 2024. – № 17. – С. 645.

2. Gajdzik, B. Autonomous and professional maintenance in metallurgical enterprise as activities within total productive maintenance // *Metalurgija*. – 2014.

3. Yin, S., Ding, S.X., Zhou, D. Diagnosis and prognosis for complicated industrial systems—Part I // *IEEE Transactions on Industrial Electronics*. – 2016. – № 63. – С. 2501–2505.

4. Zhou, Z., Wen, C., Yang, C. Fault isolation based on k-nearest neighbor rule for industrial processes // *IEEE Transactions on Industrial Electronics*. – 2016. – № 63. – С. 2578–2586.

5. Zhang, X., Chen, Y., Miao, J., Li, C., Yao, X. Preventive Maintenance Strategy for Train Doors Based on the Competitive Weibull Theory // *Chinese Journal of Mechanical Engineering*. – 2020. – С. 1–15.

6. Wang, S., Jin, S., Deng, D., Fernandez, C. A critical review of online battery remaining useful lifetime prediction methods // *Frontiers in Mechanical Engineering*. – 2021. – № 7. – С. 719718.

7. Cai, B., Huang, L., Xie, M. Bayesian networks in fault diagnosis // *IEEE Transactions on Industrial Informatics*. – 2017. – № 13. – С. 2227–2240.

8. Sun, W., Zhao, R., Yan, R., Shao, S., Chen, X. Convolutional discriminative feature learning for induction motor fault diagnosis // *IEEE Transactions on Industrial Informatics*. – 2017. – № 13. – С. 1350–1359.

9. Shao, S., McAleer, S., Yan, R., Baldi, P. Highly accurate machine fault diagnosis using deep transfer learning // *IEEE Transactions on Industrial Informatics*. – 2018. – № 15. – С. 2446–2455.

10. Kilius, Š., Gailius, D., Knyva, M., Balˇciunas, G., Meškuotien ¯ e, A., Dobilien ˙ e, J., Joneli ˙ unas, S., Kuzas, P. Time Delay Characterization in Wireless

Sensor Networks for Distributed Measurement Applications // *Journal of Sensor and Actuator Networks*. – 2024.

11. Samara, M. A., Bennis, I., Abouaissa, A., Lorenz, P. A survey of outlier detection techniques in IoT: Review and classification // *Journal of Sensor and Actuator Networks*. – 2022. – Vol. 11, No. 4.

12. Chen, Z., Gryllias, K., Li, W. Intelligent fault diagnosis for rotary machinery using transferable convolutional neural network // *IEEE Transactions on Industrial Informatics*. – 2019. – Vol. 16. – P. 339–349.

13. Saufi, S. R., Ahmad, Z. A. B., Leong, M. S., Lim, M. H. Gearbox fault diagnosis using a deep learning model with limited data sample // *IEEE Transactions on Industrial Informatics*. – 2020. – Vol. 16. – P. 6263–6271.

14. Hu, Y., Liu, R., Li, X., Chen, D., Hu, Q. Task-sequencing meta learning for intelligent few-shot fault diagnosis with limited data // *IEEE Transactions on Industrial Informatics*. – 2021. – Vol. 18. – P. 3894–3904.

15. Lu, S., Gao, Z., Xu, Q., Jiang, C., Zhang, A., Wang, X. Class-imbalance privacy-preserving federated learning for decentralized fault diagnosis with biometric authentication // *IEEE Transactions on Industrial Informatics*. – 2022. – Vol. 18. – P. 9101–9111.

16. Ge, M., Xu, Y., Du, R. An Intelligent Online Monitoring and Diagnostic System for Manufacturing Automation // I*EEE Transactions on Automation Science and Engineering*. – 2008. – Vol. 5. – P. 127–139.

17. Fang, Q., Xiong, G., Zhou, M., Tamir, T. S., Yan, C. B., Wu, H., Shen, Z., Wang, F. Y. Process Monitoring, Diagnosis and Control of Additive Manufacturing // *IEEE Transactions on Automation Science and Engineering*. – 2022. – Vol. 21. – P. 1041–1067.

18. Che, Y., Stroe, D. I., Hu, X., Teodorescu, R. Semi-supervised self-learning-based lifetime prediction for batteries // *IEEE Transactions on Industrial Informatics*. – 2022. – Vol. 19. – P. 6471–6481.

19. Qu, X., Shi, D., Zhao, J., Tran, M. K., Wang, Z., Fowler, M., Lian, Y., Burke, A. F. Insights and reviews on battery lifetime prediction from research to practice // *Journal of Energy Chemistry*. – 2024. – Vol. 94. – P. 716–739.

20. Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning representations by back-propagating errors // *Nature*. – 1986. – Vol. 323, No. 6088. – P. 533–536.

21. F. Harashima, Y. Demizu, S. Kondo, and H. Hashimoto, "Application of neutral networks to power converter control," *in Conference Record of the IEEE Industry Applications Society Annual Meeting*,, 1989, pp. 1086–1091.

22. M. R. Buhl and R. D. Lorenz, "Design and implementation of neural networks for digital current regulation of inverter drives*," in Confer ence Record of the 1991 IEEE Industry Applications Society Annual Meeting*, 1991, pp. 415–421.

23. B.-R. Lin and R. G. Hoft, "Power electronics inverter control with neural networks," *in Proc. Eighth Annual Applied Power Electronics Conference and Exposition*, 1993, pp. 128–134.

24. P. Mehrotra, J. E. Quaicoe, and R. Venkatesan, "Development of an artificial neural network based induction motor speed estimator,"*in PESC Record*. 27th Annual IEEE Power Electronics Specialists Conference, vol. 1, 1996, pp. 682–688.

25. S. Weerasooriya and M. A. El-Sharkawi, "Identification and control of a DC motor using back-propagation neural networks," *IEEE Trans. Energy Convers*., vol. 6, no. 4, pp. 663–669, Dec. 1991.

26. M. A. Rahman and M. A. Hoque, "Online self-tuning ANN-based speed control of a PM DC motor," *IEEE/ASME Trans. Mechatronics*, vol. 2, no. 3, pp. 169–178, Sept. 1997.

27. H. Tsai, A. Keyhani, J. Demcko, and D. Selin, "Development of a neural network based saturation model for synchronous generator analysis," *IEEE Trans. Energy Convers*., vol. 10, no. 4, pp. 617–624, Dec. 1995.

28. D. S. Reay, M. Mirkazemi-Moud, T. C. Green, and B. W. Williams, "Switched reluctance motor control via fuzzy adaptive systems," *IEEE Control Systems Magazine*, vol. 15, no. 3, pp. 8–15, 1995.

29. S. Mohagheghi, R. G. Harley, T. G. Habetler, and D. Divan, “Con dition monitoring of power electronic circuits using artificial neural networks,” *IEEE Trans. Power Electron*., vol. 24, no. 10, pp. 2363– 2367, Oct. 2009

30. M. Cirrincione, M. Pucci, and G. Vitale, Power converters and AC electrical drives with linear neural networks. *CRC Press*, 2017.

31. B. K. Bose, “Neural network applications in power electronics and motor drives—an introduction and perspective,” *IEEE Trans. Ind. Electron*., vol. 54, no. 1, pp. 14–33, Feb. 2007.

32. B. K. Bose, “Global energy scenario and impact of power electronics in 21st century,” *IEEE Trans. Ind. Electron*., vol. 60, no. 7, pp. 2638–2651, 2013.

33. B. K. Bose, Power electronics and motor drives: advances and trends. Academic press, 2020.

34. S. Zhao, F. Blaabjerg, and H. Wang, “An overview of artificial intelligence applications for power electronics,” *IEEE Trans. Power Electron.*, April 2020

35. L. Gao, “The decade of deep learning,” [Online]. *Available: https://bmk.sh/2019/12/31/The-Decade-of-Deep-Learning/*, Accessed: Oct. 2021.

36. E. Monmasson, M. Hilairet, G. Spagnuolo, and M. Cirstea, “System on-Chip FPGA devices for complex electrical energy systems control,” *IEEE Ind. Electron. Mag*., March 2021.

37. Y. Cai, Y. Cen, G. Cen, X. Yao, C. Zhao, and Y. Zhang, “Tempera ture prediction of PMSMs using pseudo-siamese nested lstm,” *World Electric Vehicle Journal, vol.* 12, no. 2, p. 57, 2021

38. N. Gabdullin, S. Madanzadeh, and A. Vilkin, “Towards end-to-end deep learning performance analysis of electric motors,” *in Actuators, vol.* 10, no. 2, 2021, p. 28.

39. A. Mayr, D. Kißkalt, A. Lomakin, K. Graichen, and J. Franke, “To wards an intelligent linear winding process through sensor integration and machine learning techniques,” *Procedia CIRP*, vol. 96, pp. 80–85, 2021.

40. H. Alharkan, S. Saadatmand, M. Ferdowsi, and P. Shamsi, “Optimal tracking current control of switched reluctance motor drives using reinforcement q-learning scheduling,” *IEEE Access*, vol. 9, pp. 9926–9936, 2021.

41. Zhao, Z., Wu, J., Li, T., Sun, C., Yan, R., & Chen, X. (2021). Challenges and opportunities of AI-enabled monitoring, diagnosis & prognosis: A review. *Chinese Journal of Mechanical Engineering*, 34(1), 1-29

42. Rahman, T. A., Chek, L. W., & Ramli, N. Intelligent Vibration-based Anomaly Detection for Electric Motor Condition Monitoring.

43. Tahir, M. M., Khan, A. Q., Iqbal, N., Hussain, A., & Badshah, S. (2016). Enhancing fault classification accuracy of ball bearing using central tendency based time domain features. *IEEE Access*, 5, 72-83

44. Chen, Z.; Li, C.; Sanchez, R.V. Gearbox fault identification and classification with convolutional neural networks. *Shock. Vib*. 2015, 2015, 390134.

45. Zhao, J.; Yang, S.; Li, Q.; Liu, Y.; Gu, X.; Liu, W. A new bearing fault diagnosis method based on signal-to-image mapping and convolutional neural network. *Measurement 2021*, 176, 109088

46. Pham Van Nam and Tran Hoai Linh. "Electrocardiogram (ECG) Circuit Design and Using the Random Forest for ECG Arrhythmia Classification." *International Conference on Engineering Research and Applications*, 1 Dec. 2022, page range 477-494

47. Vuong Hoang Nam, Hoai Linh Tran, and Van Nam Pham. "Multiple Neural Network Integration Using a Binary Decision Tree to Improve the ECG Signal Recognition Accuracy." I*nternational Journal of Applied Mathematics and Computer Science*, 2014, Vol. 24, No. 3, 647–655

48. Wang, R.; Jiang, H.; Li, X.; Liu, S. A reinforcement neural architecture search method for rolling bearing fault diagnosis. Measurement 2020, 154, 107417

49. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu*, HI, USA, 21–26 July 2017; pp. 4700–4708.

50. Оберванюк У. Осолінський. Концепція модуля моніторингу роботи електродвигуна з використанням IoT. Інтелектуальні інформаційні технології в

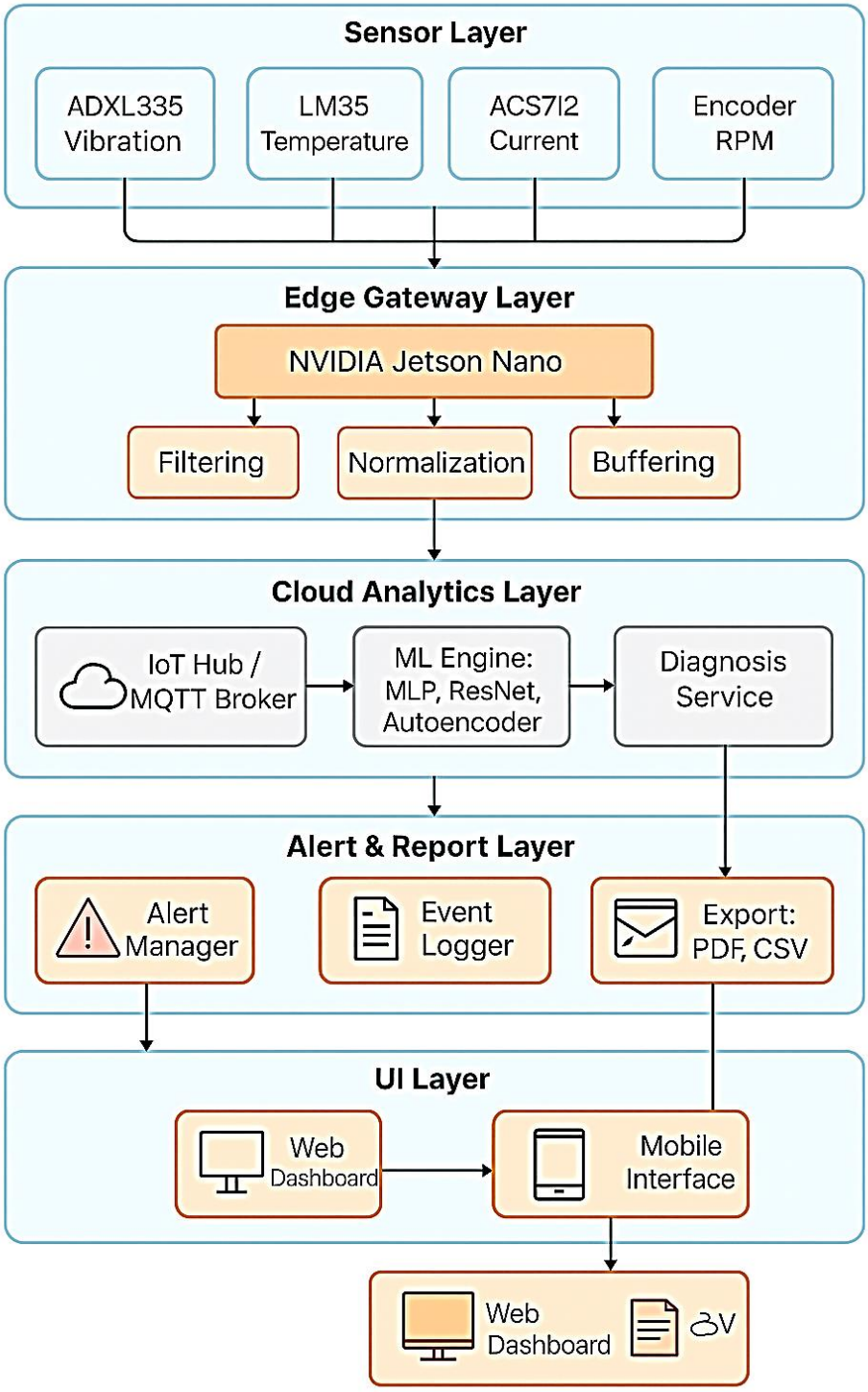
прикладних дослідженнях : тези доп. студентської наук.-практ. конф. (м. Тернопіль, 27–29 травня 2025 р.). Тернопіль, 2025. С.71-73

51. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

52. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

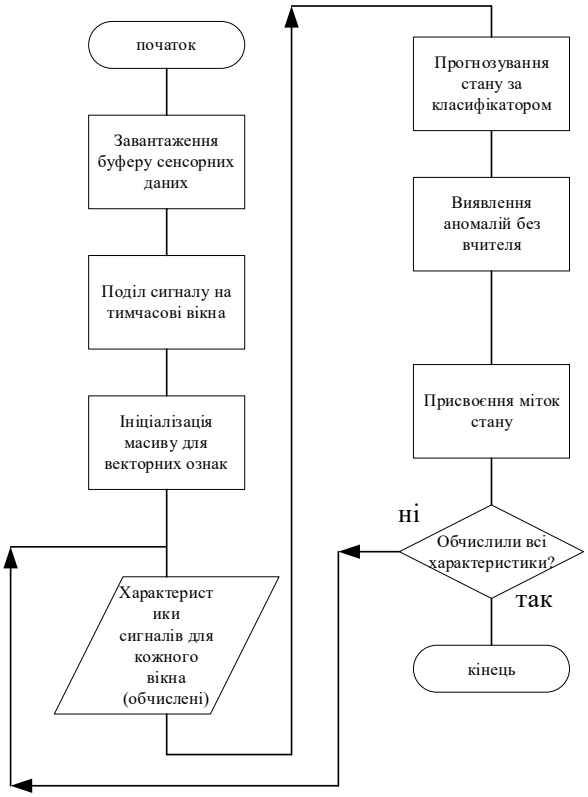
Додаток А

Архітектура програмного модуля



Додаток Б

Основні алгоритми програмного модуля



Аналіз та діагностика стану електродвигуна

Збір та попередня обробка даних

Додаток В

Код багатошарового перцептрона (MLP) з використанням бібліотек TensorFlow і Scikit-learn, для класифікації стану електродвигуна на основі сенсорних даних

```
from google.colab import files
uploaded = files.upload()
import pandas as pd
df = pd.read_csv("full_motor_dataset.csv")
df.head()
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(data=df, x='label')
plt.title("Розподіл класів у вибірці")
plt.xlabel("Клас (0 - норма, 2 - критично)")
plt.ylabel("Кількість записів")
plt.show()
X = df[['vibration', 'temperature', 'current', 'rpm']]
y = df['label']
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    epochs=50,
                    batch_size=32,
```

```
                    validation_split=0.1,
                    verbose=2)
plt.plot(history.history['accuracy'], label='Точність на тренуванні')
plt.plot(history.history['val_accuracy'], label='Точність на валідації')
plt.xlabel("Епоха")
plt.ylabel("Точність")
plt.legend()
plt.title("Графік точності моделі")
plt.grid(True)
plt.show()

plt.plot(history.history['loss'], label='Втрати на тренуванні')
plt.plot(history.history['val_loss'], label='Втрати на валідації')
plt.xlabel("Епоха")
plt.ylabel("Loss")
plt.legend()
plt.title("Графік втрат моделі")
plt.grid(True)
plt.show()
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Точність на тестовій вибірці: {round(accuracy * 100, 2)}%")
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
print(classification_report(y_test, y_pred_classes))
```

Додаток Г
Апробація отриманих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(ІІТАR-2025)

27-29 травня 2025 року

Тернопіль

2025

Збірник тез доповідей студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (ІІТАR-2025). (Тернопіль, 27-29 травня 2025 року). Тернопіль: ЗУНУ, 2025. 372 с.

До збірника увійшли тези доповідей учасників студентської науково-практичної конференції "Інтелектуальні інформаційні технології в прикладних дослідженнях" (ІІТАR – 2025), що відбувалась у рамках AI Week на базі кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету. Мета конференції — об'єднати науковців, освітян, представників ІТ-бізнесу та здобувачів освіти для обміну досвідом, презентації сучасних досліджень і впровадження інноваційних рішень у сфері комп'ютерних наук, штучного інтелекту та суміжних галузей.

За зміст наукових праць та достовірність наведених фактологічних і статистичних матеріалів відповідальність несуть автори публікацій та їхні наукові керівники. У збірнику зберігається стилістика та орфографія авторів матеріалів.

Склад організаційного комітету конференції

Керівництво оргкомітету

Мирослав Комар, д.т.н., професор, професор кафедри інформаційно-обчислювальних систем і управління

Христина Ліп'яніна-Гончаренко, д.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Надія Васильків, к.т.н., в.о. завідувача кафедри інформаційно-обчислювальних систем і управління

Члени програмного комітету

Василь Коваль, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління, заступник декана факультету комп'ютерних інформаційних технологій

Олександр Осолінський, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Павло Биковий, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Діана Загородня, к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління

Ігор Майків, к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління

Члени оргкомітету

Андрій Івасечко, викладач кафедри інформаційно-обчислювальних систем і управління

Дмитро Дюг, викладач кафедри інформаційно-обчислювальних систем і управління

Христина Юрків, студентка ФКІТ ЗУНУ, технік лабораторії з проблем інформаційних технологій

Мар'яна Соя, студентка ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління

Микола Телька, студент ФКІТ ЗУНУ, лаборант кафедри інформаційно-обчислювальних систем і управління, студентський декан факультету комп'ютерних інформаційних технологій

Оберванюк Уляна
студентка групи КНШІ-41
ivinum09@gmail.com
Осолінський Олександр
к.т.н., доцент
oso@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

КОНЦЕПЦІЯ МОДУЛЯ МОНІТОРИНГУ РОБОТИ ЕЛЕКТРОДВИГУНА З ВИКОРИСТАННЯМ IOT

Завдяки можливості підключатися та збирати дані практично з усього і скрізь, IoT полегшує комплексний моніторинг складних систем, дозволяючи виявляти аномалії та потенційні проблеми. Використовуючи потужність Інтернету речей, промисловість може отримати безцінне уявлення про здоров'я та продуктивність свого обладнання та процесів, тим самим підвищуючи ефективність і передбачувані практики технічного обслуговування. Така конвергенція технологій прокладає шлях до трансформаційних змін у різних секторах, відкриває нові можливості та оптимізує діяльність у промисловому ландшафті [1, 2]. IoT має вирішальне значення для реалізації систем діагностики, створюючи мережу, яка з'єднує розумні пристрої в інформаційній системі та полегшує обмін даними з центральним сховищем. Мережевий зв'язок між пристроями IoT демонструє унікальні характеристики та відмінності порівняно зі звичайними пристроями [3], особливо в хмарно-туманному середовищі [4].

Сучасний рівень техніки показує відсутність загальної моделі витрат на технічне обслуговування, застосовної в усіх промислових контекстах, причому запропоновані моделі часто є специфічними для конкретних випадків або складними для практичного впровадження. Крім того, у документах про оптимізацію витрат на прогнозну модель обслуговування (PdM) зазвичай не помічається концепція ризику та рідко враховується невизначеність прийняття людських рішень. Промислове технічне обслуговування в минулому розглядалося як реактивна фонова діяльність, яка виконувалася лише після збою системи. Однак із появою профілактичного та прогнозного технічного обслуговування ця сфера значно розвинулася, пропонуючи різні стратегії з явними перевагами та проблемами [5]. Тому розробка систем для моніторингу роботи електродвигуна з використанням IoT та ШІ є актуальною задачею, вирішення якої дасть можливість зменшити частоту критичних відмов, та поломок в таких системах.

Розробка концепції модуля моніторингу роботи електродвигуна з використанням IoT

Система моніторингу та діагностики електродвигунів, яка поєднує технології Інтернету речей, машинного навчання та штучного інтелекту, має бути побудована так, щоб забезпечити ефективний збір, обробку, передачу та аналіз даних. Головною метою такої системи є своєчасне виявлення несправностей та підвищення ефективності роботи обладнання, що дозволить мінімізувати час простою і зменшити витрати на обслуговування. Для цього такий програмно-апаратний модуль повинен включати декілька рівнів, які працюють разом, забезпечуючи повний цикл моніторингу та аналізу (рисунок 1).

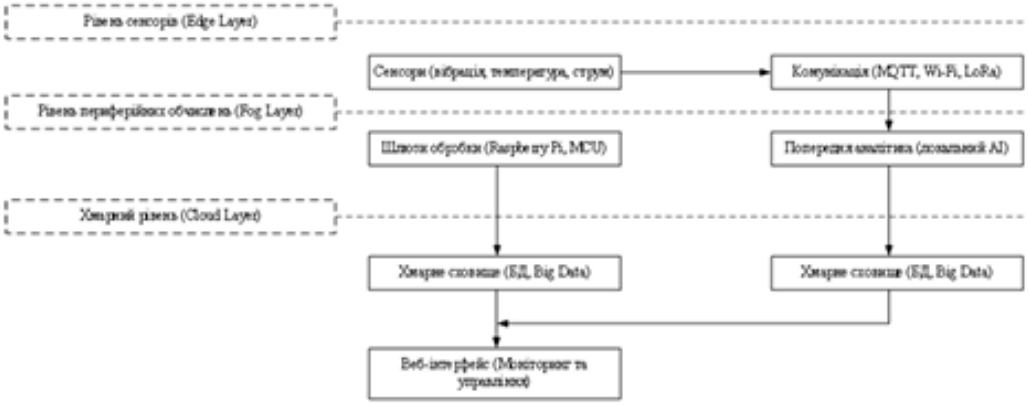


Рисунок 1 - Загальна архітектура модуля

На першому рівні відбувається безпосередній збір даних за допомогою спеціальних датчиків, що встановлюються на електродвигуни. Ці датчики можуть вимірювати різні параметри, такі як вібрація, температура, споживання струму, швидкість обертання, магнітне поле та інші фізичні характеристики.

Датчики повинні підключатися до системи через дротові або бездротові інтерфейси, такі як Bluetooth Low Energy, Wi-Fi або LoRaWAN.

Після збору даних вони надходять до шлюзу обробки, який виконує первинний аналіз та фільтрацію інформації. Шлюзи можуть базуватися на пристроях з обмеженими обчислювальними можливостями, таких як Raspberry Pi або інші мікрокомп'ютери. Вони дозволяють виконувати базові алгоритми аналізу, такі як нормалізацію даних, виявлення шуму або усереднення показників. Після цього відфільтровані та структуровані дані передаються далі, у центральну хмарну систему, де здійснюється глибший аналіз.

Основна обробка даних відбувається на хмарному рівні, де використовується потужна аналітика та алгоритми машинного навчання. Ці алгоритми аналізують

історичні дані, порівнюють їх з поточними вимірюваннями і прогнозують можливі несправності на основі виявлених закономірностей.

Одним із ключових аспектів системи є її інтеграція з інтерфейсом управління, який дозволяє користувачам отримувати доступ до аналітичних даних та графічного представлення інформації. Це може бути веб-платформа або мобільний додаток, що дозволяє операторам моніторити стан обладнання в режимі реального часу.

Висновки

Оскільки система є модульною та масштабованою, її можна легко адаптувати під конкретні потреби підприємства. Вона може бути налаштована, як для невеликих виробничих підприємств, так і для великих промислових комплексів. Додавання нових сенсорів або зміна параметрів моніторингу може здійснюватися без значних змін в архітектурі. Це забезпечує гнучкість та адаптивність системи до різних сценаріїв використання.

Така система об'єднує можливості Інтернету речей, машинного навчання та хмарних технологій, що дозволяє створити ефективну платформу для моніторингу, аналізу та управління електродвигунами. Вона дає змогу операторам отримувати точну інформацію про стан обладнання, прогнозувати несправності, мінімізувати ризики та оптимізувати процеси обслуговування. Крім того, впровадження такої технології дозволяє підвищити ефективність використання ресурсів, зменшити витрати на ремонт і покращити загальну продуктивність виробничих процесів.

Список використаних джерел

1. Ioannides, M.G., Stamelos, A.P., Papazis, S.A., Stamataki, E.E., Stamatakis, M.E. Internet of Things-Based Control of Induction Machines: Specifics of Electric Drives and Wind Energy Conversion Systems // Energies. – 2024. – № 17. – С. 645.
2. Zhou, Z., Wen, C., Yang, C. Fault isolation based on k-nearest neighbor rule for industrial processes // IEEE Transactions on Industrial Electronics. – 2016. – № 63. – С. 2578–2586.
3. Sun, W., Zhao, R., Yan, R., Shao, S., Chen, X. Convolutional discriminative feature learning for induction motor fault diagnosis // IEEE Transactions on Industrial Informatics. – 2017. – № 13. – С. 1350–1359.
4. Shao, S., McAleer, S., Yan, R., Baldi, P. Highly accurate machine fault diagnosis using deep transfer learning // IEEE Transactions on Industrial Informatics. – 2018. – № 15. – С. 2446–2455.
5. Samara, M. A., Bennis, I., Abouaissa, A., Lorenz, P. A survey of outlier detection techniques in IoT: Review and classification // Journal of Sensor and Actuator Networks. – 2022. – Vol. 11, No. 4.