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

ЛЕУС Віталій Миколайович

Інтелектуальна система керуванням обладнанням за допомогою розпізнавання жестів / Intelligent Hardware Control System Based on Gesture Recognition

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма – Штучний інтелект

Кваліфікаційна робота

Виконав студент групи КНШІ-41
В.М. Леус
_____

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

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ – 2026

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

«ЗАТВЕРДЖУЮ»
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ЛЕУСУ Віталію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Інтелектуальна система керуванням обладнанням за допомогою розпізнавання жестів / Intelligent Hardware Control System Based on Gesture Recognition

керівник роботи к.т.н., доцент О.Р. Осолінський,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
– провести предметної області та сценарії керування обладнанням;
– провести огляд і аналіз існуючих рішень та засобів розпізнавання жестів;
– сформувати постановку задачі;
– розробити загальну архітектуру проєктованої системи та інформаційні зв'язки;
– розробити алгоритми розпізнавання жестів і формування команд керування;
– розробити інформаційне забезпечення системи;
– провести вибір програмних і апаратних засобів;
– реалізувати ключові модулі системи;
– реалізувати інтерфейс користувача;
– протестувати працездатність системи та проаналізувати результати.
5. Перелік графічного матеріалу в роботі:
– блок-схеми алгоритмів захоплення, трансформації та класифікації;
– діаграма логічної структури даних;
– UML-діаграма станів інтерфейсу системи;

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 01 грудня 2025 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. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ В.М. Леус
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 79 с., 17 рис., 8 табл., 34 джерела, 6 додатків.

Об’єктом дослідження є процеси людино-машинної взаємодії, асинхронного збору та інтелектуальної обробки просторово-часових даних у безконтактних інтерфейсах керування обладнанням.

Метою роботи є розробка програмно-апаратного комплексу інтелектуального безконтактного керування обладнанням на основі розпізнавання просторово-часових патернів жестів оператора.

Методи дослідження: методи комп'ютерного зору для зчитування 3D-координат; методи машинного (багатошаровий перцептрон) та глибокого навчання (рекурентні нейронні мережі) для класифікації жестів; методи математичної стандартизації просторових ознак, а також методи програмування мікроконтролерних систем.

Результатом роботи є система, яка включає оптичний сенсор Leap Motion, інтелектуальне програмне ядро мовою Python (з моделями MLP та LSTM) та апаратний виконавчий рівень на базі мікроконтролера ATmega32U4. Розроблено алгоритми фільтрації помилкових спрацювань (дебаунсинг і кулдаун) та створено асинхронний графічний інтерфейс для моніторингу станів системи у реальному часі.

Розроблений комплекс може бути використаний у робототехнічних системах, стерильних медичних приміщеннях, на промислових підприємствах зі шкідливими умовами праці, а також у системах автоматизації «розумного дому» для безпечного безконтактного керування пристроями.

Ключові слова: БЕЗКОНТАКТНЕ КЕРУВАННЯ, КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ЖЕСТІВ, МАШИННЕ НАВЧАННЯ, МІКРОКОНТРОЛЕР, НЕЙРОННІ МЕРЕЖІ, ЛЮДИНО-МАШИННИЙ ІНТЕРФЕЙС.

ANNOTATION

Explanatory note to the qualification thesis: 79 pages, 19 figures, 8 tables, 34 references, 6 appendices.

The object of the research is the processes of human-machine interaction, asynchronous collection, and intelligent processing of spatio-temporal data in contactless equipment control interfaces.

The purpose of the work is to develop a system for intelligent contactless load control based on the recognition of spatio-temporal patterns of operator gestures.

Research methods: computer vision methods for reading 3D coordinates; machine learning (multilayer perceptron) and deep learning (recurrent neural networks) methods for gesture classification; methods of mathematical standardization of spatial features, as well as programming methods for microcontroller systems.

The result of the work is a hardware-software complex that includes a Leap Motion optical sensor, a Python-based intelligent software core (with MLP and LSTM models), and a hardware executive level based on the ATmega32U4 microcontroller. Algorithms for filtering false triggers (debouncing and cooldown) were developed, and an asynchronous graphical interface was created to monitor system states in real time.

The developed complex can be used in robotic systems, sterile medical environments, at industrial enterprises with hazardous working conditions, and in "smart home" automation systems for safe contactless device control.

Keywords: CONTACTLESS CONTROL, COMPUTER VISION, GESTURE RECOGNITION, MACHINE LEARNING, MICROCONTROLLER, NEURAL NETWORKS, HUMAN-MACHINE INTERFACE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕОМ – електронно-обчислювальна машина

COM – communication port (послідовний комунікаційний порт)

FIFO – first-in, first-out (алгоритм обслуговування черги)

IDE – integrated development environment (інтегроване середовище розробки)

LED – light-emitting diode (світлодіод)

LSTM – long short-term memory (рекурентна нейронна мережа довгої короткочасної пам'яті)

MLP – multi-layer perceptron (багатошаровий перцептрон)

RGB – red, green, blue (адитивна колірна модель)

UI – user interface (інтерфейс користувача)

UML – unified modeling language (уніфікована мова моделювання)

USB – universal serial bus (універсальна послідовна шина)

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімкою еволюцією людино-машинних інтерфейсів. Традиційні засоби контактної взаємодії, такі як клавіатури, маніпулятори типу «миша» та сенсорні екрани, в багатьох специфічних галузях досягли своїх фізичних та ергономічних обмежень [1]. Зокрема, в умовах стерильних медичних операційних, на виробництвах із підвищеною хімічною чи радіаційною небезпекою, а також у сучасних автоматизованих промислових зонах прямий фізичний контакт оператора з органами керування є небажаним або критично небезпечним через ризики контамінації, травмування чи пошкодження чутливого обладнання [2, 3].

У зв'язку з цим особливої актуальності набуває розробка та впровадження інтелектуальних систем безконтактного людино-машинного керування, де основним інформаційним каналом виступають природні координаційні рухи та просторові жести рук оператора [4]. Застосування методів комп'ютерного зору та алгоритмів штучного інтелекту дозволяє трансформувати тривимірні координати анатомічних точок кисті у детерміновані виконавчі команди. Проте більшість існуючих рішень на базі стандартних RGB-камер вимагають значних обчислювальних ресурсів для безперервної локалізації об'єктів і є чутливими до умов освітленості робочої зони [5]. Перспективним вирішенням цієї задачі є інтеграція спеціалізованих інфрачервоних оптичних сенсорів із оптимізованими гібридними моделями машинного й глибокого навчання, що дозволяє забезпечити стабільну роботу контуру керування в режимі реального часу з мінімальними обчислювальними затримками.

Метою кваліфікаційної роботи є розробка програмно-апаратного комплексу інтелектуального безконтактного керування обладнанням на основі розпізнавання просторово-часових патернів жестів оператора.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів безконтактної людино-машинної взаємодії, сучасних засобів комп'ютерного зору та архітектур побудови інтелектуальних інтерфейсів;

– сформулювати постановку задачі проектування програмно-апаратного комплексу керування навантаженнями;

– розробити математичні алгоритми збору та попередньої обробки просторових даних, включаючи геометричну центризацію відносно опорного орієнтира долоні та Z-стандартизацію 63-вимірного вектора ознак;

– спроектувати та натренувати модуль класифікації статичних жестів на основі моделі багатошарового перцептрона (MLP);

– спроектувати та натренувати модуль розпізнавання часових послідовностей динамічних жестів (свайпів) на основі рекурентної глибокої нейромережі (LSTM);

– розробити інформаційне забезпечення системи, включаючи логічні структури конфігураційних файлів та оперативних журналів типу FIFO у пам'яті ЕОМ;

– обґрунтувати вибір програмно-технологічного стеку та апаратних мікроконтролерних компонентів системи;

– реалізувати програмне ядро на Python, алгоритми фільтрації часового брязкання (дебаунсинг і кулдаун) та C++ прошивку для мікроконтролера ATmega32U4;

– спроектувати та впровадити графічний інтерфейс користувача для візуального моніторингу контуру класифікації;

– провести комплексне інтеграційне тестування розробленого рішення, виконати аналіз точності моделей, потенційних помилок та системних обмежень.

Об’єкт дослідження – процеси людино-машинної взаємодії, асинхронного збору та інтелектуальної обробки просторово-часових даних у безконтактних інтерфейсах керування обладнанням.

Предмет дослідження – алгоритми просторової трансформації, моделі машинного та глибокого навчання, методи часової фільтрації дескрипторів,

структури оперативних даних та програмно-апаратні засоби інтеграції оптичних сенсорів комп'ютерного зору з мікроконтролерами.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах студентських науково-практичних конференцій: “І міжнародна науково-практична конференція студентів і молодих вчених” (ICSPM 2026), м. Тернопіль, Україна, 21-22 травня 2026 р. та “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 СТАН ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сучасний етап розвитку інформаційних технологій характеризується стрімким переходом від традиційних інтерфейсів до природних способів взаємодії між людиною та технічними системами. У центрі цієї трансформації знаходиться концепція людино-машинної взаємодії, яка спрямована на створення інтуїтивно зрозумілих та ефективних методів комунікації з комп’ютером, або іншими словами керування комп’ютерним обладнанням [14]. Одним із найбільш перспективних напрямків у цій сфері є розпізнавання жестів рук, що дозволяє керувати складним обладнанням без фізичного контакту з пристроями введення [12].

Жести за своєю суттю є формою невербальної комунікації, що охоплює виразні рухи рук, кистей або пальців, які несуть певний зміст або намір користувача [12]. Розпізнавання жестів визначається як здатність обчислювальної системи інтерпретувати ці рухи як значущі команди [9]. Цей процес є складним завданням, що поєднує в собі алгоритми комп’ютерного зору, обробку сигналів та методи штучного інтелекту для перетворення вхідних візуальних даних у цифрові інструкції.

Для систематизації розуміння розмаїття технологічних рішень, які використовуються для ідентифікації рухів людини, застосовується класифікація за типом вхідних даних та апаратного забезпечення. Існує багато різних методів розпізнавання жестів, тому для їх систематизації, нижче наведено рисунок 1.1 та коротко описано кожен з них.

Підхід на основі сенсорних рукавичок використовує контактні датчики, закріплені на рукавичці, для отримання точних координат, орієнтації та конфігурації долоні й пальців у реальному часі. Не зважаючи на високу точність, він створює фізичні перешкоди для користувача через необхідність провідного підключення та має значну вартість обладнання.

Підхід на основі комп'ютерного зору є найбільш поширеним завдяки забезпеченню безконтактної взаємодії за допомогою різних типів камер. Основні

проблеми цього підходу включають мінливість освітлення, складність фону та проблему перекриття об'єктів.

Рисунок 1.1 – Методи розпізнавання жестів рук

Підхід на основі комп'ютерного зору є найбільш поширеним завдяки забезпеченню безконтактної взаємодії за допомогою різних типів камер. Основні проблеми цього підходу включають мінливість освітлення, складність фону та проблему перекриття об'єктів.

Розпізнавання на основі кольору шкіри базується на піксельному або регіональному аналізі зображення для сегментації руки за характерним тоном шкіри. Він є популярним завдяки простоті реалізації, проте чутливий до змін інтенсивності світла та наявності об'єктів схожого кольору на фоні.

Розпізнавання на основі зовнішнього вигляду полягає у вилученні 2D-ознак (інтенсивності пікселів) безпосередньо з кадрів відеопотоку без попередньої сегментації. Метод демонструє високу швидкість роботи в реальному часі та здатність розпізнавати різні відтінки шкіри.

Розпізнавання на основі руху фокусується на вилученні та відстеженні об'єкта через серію кадрів для розпізнавання динамічних жестів. Головними труднощами тут є наявність інших рухомих об'єктів на фоні та можлива втрата цілі через швидкі рухи або оклюзію.

Розпізнавання на основі скелета базується на формалізації руки як набору суглобів та кісток, що описуються геометричними атрибутами (координати, кути між суглобами). Це дозволяє зосередитися на статистичних та геометричних параметрах, значно покращуючи детекцію складних жестів.

Розпізнавання на основі глибини реалізується завдяки використанню спеціалізованих камер глибини, що дозволяє отримувати 3D-геометричну інформацію, яка не залежить від освітлення, тіней або кольору об'єктів. Це спрощує сегментацію руки, хоча використання методу часто обмежене габаритами та вартістю сенсорів.

Розпізнавання на основі 3D-моделей використовує об'ємні або скелетні 3D-кінематичні моделі руки з високим ступенем свободи. Параметри моделі постійно оновлюються в процесі зіставлення вхідного зображення з проєкцією моделі, що забезпечує високу точність оцінки пози.

Розпізнавання на основі глибокого навчання – сучасний підхід використовує багаторівневі нейронні мережі для автоматичного навчання на великих наборах даних. Метод забезпечує найбільш надійні результати та високу здатність до прогнозування, хоча потребує значних ресурсів для навчання алгоритмів.

Повертаючись назад до систем людино-машинної взаємодії, як вже було сказано, вони виступають сполучною ланкою між намірами людини та виконавчими механізмами обладнання [13, 14]. Розвиток цих систем сьогодні спрямований на створення максимально природних інтерфейсів, що здатні імітувати звичні способи комунікації між людьми, де ключове місце посідають

жести, мовлення та вирази обличчя [14]. Архітектуру таких систем у науковій літературі прийнято класифікувати на дві великі групи залежно від кількості та різноманітності каналів введення і виведення інформації, а саме на унімодальні та мультимодальні системи [13]. Детальна структура цієї класифікації представлена на рисунку 1.2, що відображає ієрархію способів сприйняття даних обчислювальною системою та допомагає обрати оптимальну модель для конкретних умов експлуатації обладнання.

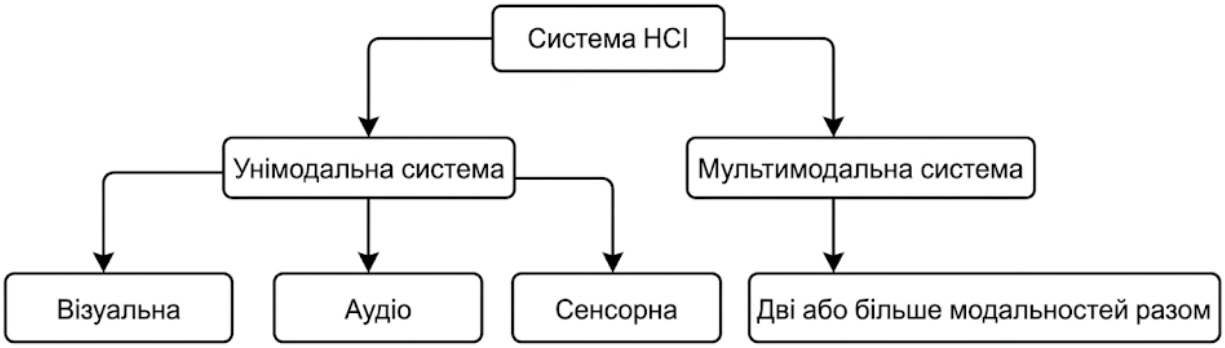


Рисунок 1.2 – Перелік систем для класифікації людино-машинної взаємодії за модальністю введення

Унімодальні системи базуються на використанні лише одного каналу передачі інформації від людини до машини, що дозволяє зосередити обчислювальні ресурси на високій точності обробки конкретного типу сигналу [13, 14]. Такі системи можуть бути візуальними, аудіальними або базуватися на даних із різних типів фізичних сенсорів. Візуальні унімодальні системи використовують методи комп'ютерного зору для відстеження рухів тіла, розпізнавання жестів рук, аналізу міміки обличчя або детекції напрямку погляду користувача [14]. Зокрема, розпізнавання жестів у таких системах дозволяє реалізувати безконтактне керування графічними інтерфейсами або складними технічними вузлами, що стає критично важливим в умовах, коли традиційні пристрої введення є недоступними або їх використання є небажаним із міркувань гігієни. Аудіальні системи, у свою чергу, фокусуються на розпізнаванні мовлення, ідентифікації мовця за голосом або розпізнаванні емоційного стану через акустичні характеристики голосу [13].

Окрему категорію складають системи на основі автономних сенсорів, таких як акселерометри або гіроскопи, що зчитують фізичні параметри руху без використання візуального каналу спостереження.

Незважаючи на високу ефективність унімодальних підходів в специфічних задачах, вони часто не здатні забезпечити повноцінне та розуміння намірів користувача в складних умовах. Оскільки люди під час звичного спілкування природно використовують одночасно кілька способів передачі інформації, логічним етапом еволюції стали мультимодальні системи людино-машинної взаємодії [13]. Такі фреймворки інтегрують дані з декількох джерел для більш комплексного та надійного оцінювання намірів людини, що значно підвищує стійкість системи до зовнішніх завад [14]. Мультимодальні інтерфейси зазвичай базуються на комбінаціях різних вхідних сигналів, наприклад, одночасному використанні жестів рук та голосових команд або відстеженні положення голови разом із аналізом мовлення [13]. Використання декількох модальностей дозволяє обчислювальній системі компенсувати недоліки або помилки окремих каналів, наприклад, у ситуаціях, коли візуальний розпізнавач жестів не може чітко ідентифікувати команду через недостатнє освітлення, але аудіальний канал надає підтверджуючу голосову інформацію.

1.2 Огляд і аналіз існуючих рішень

Світовий досвід впровадження систем людино-машинної взаємодії демонструє широкий спектр підходів, кожен із яких має специфічні переваги та обмеження залежно від сфери застосування, будь то сільське господарство, авіація чи медицина [10, 11].

Одним з найбільш революційних рішень останніх років стала розробка компанією Google фреймворку MediaPipe Hands, який на сьогодні є фактичним стандартом для відстеження скелета руки в реальному часі [18]. Основна перевага цієї системи полягає у використанні двоступеневої архітектури нейронних мереж, де перша модель здійснює швидку детекцію долоні, а друга передбачає положення

21-ї ключової точки суглобів [18]. Такий підхід дозволяє досягти високої частоти кадрів навіть на мобільних пристроях з обмеженими обчислювальними ресурсами, що робить MediaPipe ефективним інструментом для інтерактивних систем, що працюють без спеціалізованих датчиків глибини. Проте, незважаючи на стабільність трекінгу, MediaPipe вимагає додаткових програмних надбудов для розпізнавання динамічних паттернів або складних жестів, що виходять за межі статичної геометрії кисті.

У вітчизняних дослідженнях велика увага приділяється інтеграції жестових інтерфейсів у системи «розумного будинку» та кіберфізичні середовища. Прикладом успішної реалізації такого підходу є моделі, що поєднують розпізнавання жестів із голосовими командами для створення мультимодальних систем керування [2]. Це дозволяє значно підвищити надійність системи, оскільки користувач отримує альтернативні канали взаємодії у випадку перешкод в одному з них. Також існують розробки, спрямовані на контекстну адаптацію жестового керування, де система здатна змінювати інтерпретацію жестів залежно від відкритого програмного вікна або стану обладнання, що суттєво розширює функціональність інтерфейсу людино-машинної взаємодії без збільшення кількості необхідних рухів [1].

Для більш детального порівняння існуючих апаратних засобів, що використовуються для ідентифікації рухів людини в різних умовах, нижче наведено таблицю 1.1. Вона систематизує дані про джерела збору інформації, ключові технологічні виклики та сфери практичного впровадження.

Таблиця 1.1 – Порівняння засобів розпізнавання жестів

Джерело даних	Виклики (W_1)	Інструменти (W_2)	Характеристики (W_3)	Застосування (W_4)	Оцінка (Score)
1	2	3	4	5	6
Системи на основі камер:					
Вебкамери, Kinect	Поле зору (7)	Доступні (10)	Висока точність (8)	HCI, побут (9)	8.2

Продовження таблиці 1.1

1	2	3	4	5	6
Intel RealSense	Оклюзії (6)	Висока вартість (7)	RGB + Глибина (9)	Ігри, робототехніка (8)	7.6
OpenPose	Конфіденційність (5)	Програмні (9)	Поза тіла (7)	Спорт (6)	6.7
Носимі пристрої					
Смарт-годинники	Батарея (5)	Поширені (9)	Безперервність (6)	Фітнес (9)	6.6
Розумні рукавички	Комфорт (4)	Дорогі (6)	Точність кисті (9)	Промисловість (8)	7.2
Браслет Myo	Радіус дії (5)	Обмежені (6)	Активність м’язів (7)	VR (7)	6.2
Браслет Nymi	Обробка даних (4)	Низька (5)	Автентифікація (6)	Охорона здоров’я (5)	5.1
Інерціальні модулі:					
Акселерометри	Калібрування (5)	Поширені (10)	Енергоефективність (6)	Робототехніка (7)	6.6
Магнітометри	Шуми, дрейф (4)	Компактні (9)	Орієнтація (5)	Аналіз руху (6)	5.6
Джерело даних	Виклики (W_1)	Інструменти (W_2)	Характеристики (W_3)	Застосування (W_4)	Оцінка (Score)
Датчики глибини:					
Kinect / RealSense	Дальність (6)	Середня (7)	3D-інформація (9)	Ігри (8)	7.6
Structure Sensor	Освітлення (6)	Мобільні (8)	Портативність (8)	Робототехніка (7)	7.3
Zed Camera	Оклюзії (6)	Складні (6)	Стереобачення (9)	AR (8)	7.4

Продовження таблиці 1.1

1	2	3	4	5	6
Електроміо-графія:					
Поверхневі EMG	Розміщення (4)	Спеціальні (5)	Активність м’язів (7)	Протезування (6)	5.6
Внутріш-ньом’язові	Інвазивність (2)	Низька (3)	Максимальна точність (9)	Реабілітація (4)	5.2

Для об'єктивного порівняння методів розпізнавання жестів та вибору оптимального підходу було впроваджено формалізовану оцінку ефективності за методом зваженої суми із оцінкою параметрів за 10-бальною шкалою:

Інтегральний показник Score розраховується за наступною формулою:

$$Score = C_1 \times W_1 + C_2 \times W_2 + C_3 \times W_3 + C_4 \times W_4, \quad (1.1)$$

де W_1 — рівень подолання технологічних викликів (складність усунення перешкод);

W_2 — доступність та поширеність інструментарію;

W_3 — якість технічних характеристик (точність, швидкість);

W_4 — універсальність застосування;

C_n — коефіцієнт.

Коефіцієнти вагомості визначені відповідно до пріоритетів розробки системи: $C_1 = 0{,}3$ (стійкість до перешкод), $C_2 = 0{,}2$ (доступність), $C_3 = 0{,}4$ (продуктивність) та $C_4 = 0.1$ (універсальність).

Проведений детальний аналіз, результати якого занесено до таблиці 1.1, дозволяє стверджувати, що для завдань промислової автоматизації та систем «Розумний будинок» найвищу інтегральну оцінку мають системи на основі стандартних камер та спеціалізованих контролерів типу Leap Motion (8.2 бала).

Важливим напрямком у розвитку сучасних методів є застосування нейронних мереж та механізмів злиття ознак для підвищення точності розпізнавання в динамічних сценаріях [6]. Їх використання дозволяє системі враховувати часову

залежність між кадрами, що є критичним для диференціації жестів, які мають схожі початкові фази, але різні траєкторії завершення. Такі інтелектуальні системи часто розгортаються на базі вбудованих платформ, таких як Raspberry Pi 5, що забезпечують необхідну продуктивність для обробки відеопотоку в реальному часі за допомогою прискорювачів штучного інтелекту [8]. Це дозволяє створювати автономні пристрої керування, які не залежать від стабільного інтернет-з'єднання та хмарних сервісів, забезпечуючи при цьому високу швидкість реакції.

Окремим класом йдуть системи, орієнтовані на спеціалізовані галузі. Наприклад, у сільському господарстві розробляються фреймворки для взаємодії людини з роботами, де жести використовуються для дистанційного керування технікою в умовах відкритого простору та змінного освітлення [11]. В авіації та робототехніці розпізнавання жестів стає основою для планування траєкторій руху безпілотних літальних апаратів, що вимагає надзвичайно низької затримки та стійкості до швидких переміщень об'єкта в кадрі [10]. Такі специфічні умови експлуатації змушують дослідників вдосконалювати методи сегментації та класифікації, використовуючи глибокі нейронні мережі для фільтрації шумів та компенсації дрейфу сенсорів.

Соціальний аспект розпізнавання жестів розкривається в роботах, присвячених перекладу жестової мови та допомозі людям із порушеннями слуху. Такі дослідження описують підсистеми перекладу, інтегровані в кіберфізичні системи, які дозволяють автоматично трансформувати жести в текстові повідомлення або синтезований голос [4, 5]. Це вимагає використання не лише геометричних даних про положення пальців, а і аналізу пози всього тіла та міміки, що значно ускладнює архітектуру системи та висуває високі вимоги до якості вхідних даних.

До основних переваг систем на основі комп’ютерного зору належать їхня неінвазивність, відносно низька вартість обладнання та можливість використання існуючої інфраструктури – вебкамер. Проте недоліками таких систем залишаються висока обчислювальна складність алгоритмів глибокого навчання, чутливість до умов освітлення та проблема оклюзії, коли рука перекривається іншими об’єктами.

Сенсорні та носимі системи забезпечують вищу точність за будь-якого освітлення, але створюють дискомфорт для користувача та потребують регулярної підзарядки або калібрування.

1.3 Постановка задачі

Аналіз існуючих систем людино-машинної взаємодії показав, що традиційні контактні засоби керування, до яких входять сенсорні панелі, механічні перемикачі, клавіатури мають суттєві обмеження при використанні в специфічних умовах. Зокрема, у стерильних медичних середовищах, промислових зонах з агресивними факторами або комплексах керування роботизованими платформами прямий тактильний контакт є небезпечним або небажаним. З огляду на стрімкий розвиток алгоритмів комп'ютерного зору, виникає об’єктивна необхідність в розробці інтегрованого програмно-апаратного рішення для безконтактного керування, яке буде базуватися на спеціалізованих інфрачервоних сенсорах та методах штучного інтелекту, гарантуючи високу швидкість та безпеку операцій.

Основна мета дослідження – створення ефективної, завадостійкої та масштабованої архітектури програмно-апаратного комплексу, яка дозволить в режимі реального часу здійснювати збір, інтелектуальну обробку просторово-часових даних жестів оператора та їх миттєве перетворення на детерміновані виконавчі команди.

Проектована система має забезпечити безперервний моніторинг і високоточну локалізацію анатомічних точок кисті за допомогою оптичного сенсора Leap Motion Controller. Обробка даних повинна здійснюватися алгоритмами машинного та глибокого навчання (моделями MLP та LSTM) для ізольованого розпізнавання як статичних поз, так і динамічних часових траєкторій (свайпів). Керування фізичними навантаженнями має виконуватися мікроконтролерним модулем на базі чипа ATmega32U4 із підтримкою апаратного відпрацювання команд (комутація електромагнітного реле, керування сервоприводом SG90 та LED-індикація). Усі компоненти мають бути інтегровані в єдину модульну архітектуру з обов'язковою підтримкою фільтрації помилкових спрацювань

(дебаунсинг і кулдаун) та взаємодії з користувачем через асинхронний графічний інтерфейс.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Розробити архітектуру програмно-апаратного комплексу безконтактного керування з чітким розподілом на сенсорний, обчислювальний та виконавчий рівні.
2. Розробити алгоритми функціонування ключових компонентів системи, включаючи попередню геометричну стандартизацію просторових ознак та логіку фільтрації часового брязкання класів.
3. Розробити інформаційне забезпечення системи, включаючи формати збереження конфігураційних файлів моделей та структури оперативних часових буферів.
4. Провести вибір програмних і апаратних засобів для реалізації інтелектуального та мікроконтролерного рівнів.
5. Реалізувати ключові обчислювальні модулі, натренувати класифікатори на базі багатошарового перцептрона та рекурентної нейромережі.
6. Реалізувати графічний інтерфейс взаємодії з користувачем для зручного візуального моніторингу станів системи та системних логів.
7. Протестувати працездатність програмно-апаратного комплексу, оцінивши точність розпізнавання жестів на тестових вибірках та коректність апаратного відпрацювання мікроконтролерних команд у реальному часі.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Загальна архітектура проєктованої системи

На основі проведеного аналізу існуючих підходів, було запропоновано багаторівневу модульну архітектуру, яка дозволяє розділити задачі низькорівневого захоплення даних, високорівневого вилучення ознак та логічного виведення команд. Такий підхід забезпечує гнучкість системи, дозволяючи вдосконалювати окремі алгоритми без необхідності повної перебудови всього програмного комплексу [7, 15]. Концепція системи базується на принципі конвеєрної обробки відеопотоку, де кожен етап є фільтром або перетворювачем, що поступово звужує простір вхідних даних від мільйонів пікселів до конкретного коду команди керування.

Нижче наведено рисунок 2.1, що відображає концепцію пропонованого рішення.

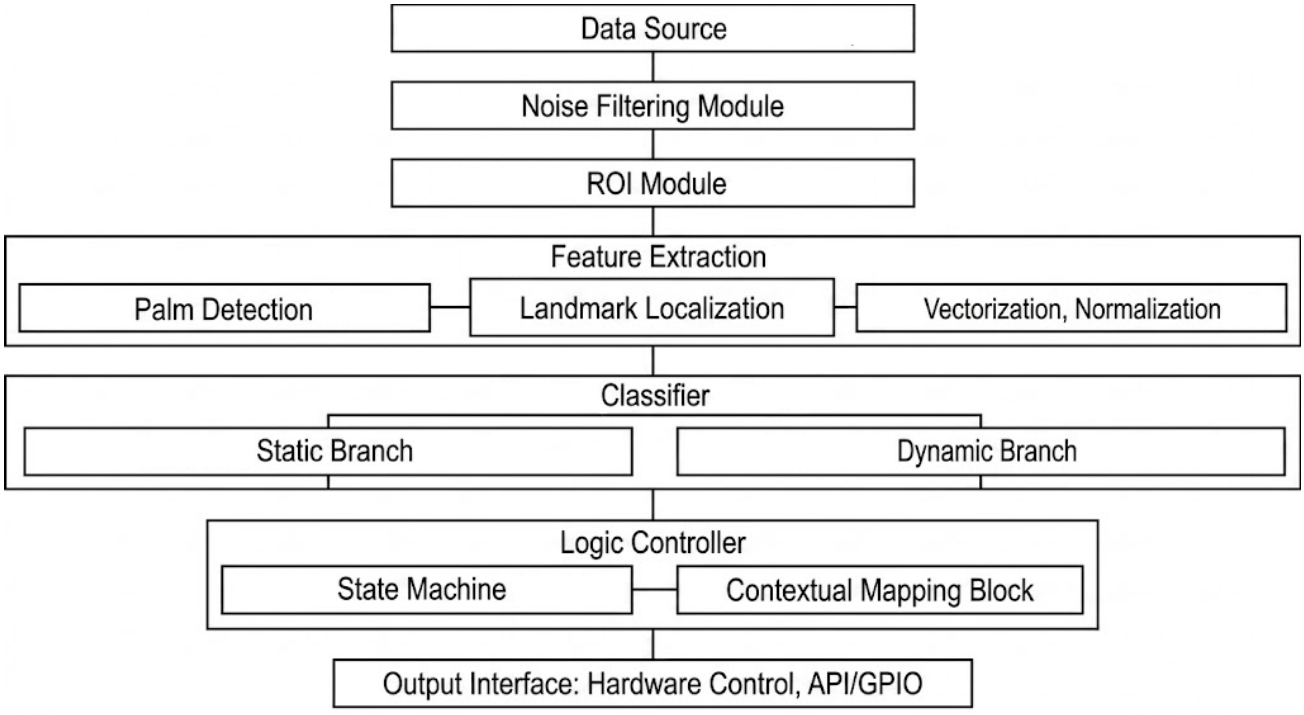


Рисунок 2.1 – Загальна архітектура системи

Першим рівнем архітектури є рівень захоплення та попередньої обробки даних. Його основне завдання полягає в отриманні стабільного відеопотоку та його

підготовці до аналізу. Концепція передбачає використання адаптивної фільтрації шумів та нормалізації інтенсивності пікселів, що дозволяє нівелювати негативний вплив змінного освітлення в робочій зоні [12, 14]. Крім того, на цьому рівні реалізується логіка просторового обмеження області інтересу. Замість обробки всього кадру, система автоматично фокусується на динамічних змінах в певній зоні, що значно знижує вимоги до обчислювальної потужності та дозволяє ігнорувати завади на фоні.

Наступним рівнем пропонованої архітектури є блок структурної абстракції та вилучення ознак. На відміну від класичних підходів, що базуються на аналізі форми долоні за кольором чи контуром, дана концепція орієнтована на побудову скелетної моделі руки. Логіка цього блоку полягає в ідентифікації ключових анатомічних точок, таких як центри суглобів та кінчики пальців, що дозволяє представити жест як набір геометричних векторів у просторі [18]. Це забезпечує інваріантність системи до масштабу (відстані руки від камери) та орієнтації кисті. Вдосконалення цього етапу полягає у впровадженні механізму прогнозування положення точок у випадку їх короткочасного перекриття іншими об'єктами, що підвищує стабільність трекінгу в реальних умовах експлуатації [14].

Центральним елементом архітектури є інтелектуальний модуль класифікації та прийняття рішень. Його логіка розділена на два паралельні процеси: розпізнавання статичних конфігурацій та аналіз динамічних траєкторій. Статичний класифікатор аналізує взаємне розташування ключових точок у поточному кадрі для виявлення миттєвих жестів-команд. Водночас динамічний аналізатор використовує буфер часових послідовностей для ідентифікації рухів рук протягом певного інтервалу часу [6, 16]. Така гібридна архітектура дозволяє розрізняти складні команди, де значення жесту може залежати від початкового та кінцевого положення руки. Для забезпечення високої точності розпізнавання концепція передбачає використання ймовірнісних моделей, які оцінюють впевненість системи в ідентифікації кожного конкретного класу жестів.

Одним з основних компонентів пропонованої концепції є модуль контекстної адаптації та мапування дій. Він виступає посередником між блоком розпізнавання

та виконавчим обладнанням. Завдання цього рівня полягає у трансформації розпізнаного класу жесту в конкретний керуючий сигнал залежно від поточного стану системи [1]. Тобто, один і той самий жест може виконувати функцію регулювання гучності в медіаплеєрі або зміну швидкості обертання двигуна в промисловій установці, залежно від активного режиму. Така логіка дозволяє мінімізувати кількість жестів, які користувачеві необхідно вивчити, роблячи інтерфейс більш ергономічним та інтуїтивно зрозумілим.

Фінальним рівнем архітектури є рівень інтерфейсу та зворотного зв'язку. Він забезпечує передачу сформованих команд на периферійне обладнання та надає користувачеві візуальну або звукову інформацію про успішне спрацювання команди. Концепція передбачає використання асинхронних черг повідомлень для передачі сигналів, що гарантує відсутність блокувань в роботі системи розпізнавання при взаємодії з повільними виконавчими механізмами [15]. Крім того, на цьому рівні реалізується логіка запобігання помилковим діям – дублюванням команд або випадковим активаціям, що досягається за допомогою часових затримок та перевірки стабільності жесту протягом кількох кадрів підряд.

Дана архітектура вдосконалює існуючі рішення за рахунок глибокої інтеграції скелетного моделювання з контекстною логікою прийняття рішень, що дозволяє досягти високої надійності без необхідності використання складних багатокамерних систем або спеціальних носимих сенсорів [8, 11].

2.2 Алгоритмічне забезпечення розпізнавання жестів і формування команд керування

Ефективність та надійність функціонування інтелектуального інтерфейсу людино-машинної взаємодії в реальному часі безпосередньо визначається його алгоритмічним забезпеченням. Запропоноване алгоритмічне рішення орієнтоване на створення детермінованого, швидкого та завадостійкого математичного конвеєра, який трансформує просторові координати анатомічних орієнтирів руки оператора у дискретні керуючі сигнали для периферійного обладнання.

На відміну від традиційних алгоритмів комп'ютерного зору, які здійснюють попіксельний аналіз плоских двовимірних зображень, розроблене математичне забезпечення базується на прямій обробці просторових метричних дескрипторів скелетної моделі [12]. Це дозволяє уникнути обчислювально складних етапів попередньої сегментації фону, контурного аналізу та детекції шкіри, які демонструють низьку стабільність за умов змінного або недостатнього освітлення робочої зони маніпуляцій [22].

Логічну структуру алгоритмічного конвеєра системи розбито на чотири послідовні етапи, кожен з яких виконує роль математичного фільтра або класифікатора:

– асинхронний перехоплювач та просторовий фільтр сигналів забезпечує стабільне підключення до низькорівневого програмного інтерфейсу сенсора, оптимізує частоту дискретизації та здійснює первинне придушення високочастотних шумів і тремору [34];

– трансформація та формування ознакового простору відповідає за збір фіксованого набору просторових дескрипторів суглобів кисті, їх векторизацію та математичне масштабування для досягнення інваріантності системи до просторового зсуву долоні [24];

– гібридна класифікація реалізує паралельну обробку сформованих ознак за допомогою математичних моделей машинного та глибокого навчання, розділяючи розпізнавання миттєвих статичних поз і тривалих динамічних траєкторій [23, 25];

– контекстна інтерпретація та фільтрація брязкоту здійснює часову стабілізацію розпізнаних класів жестів, запобігає повторним помилковим активаціям виконавчих пристроїв та мапує результати у вихідні командні байти [27, 33].

2.2.1 Алгоритм отримання та просторової фільтрації даних

Первинним етапом роботи алгоритмічного конвеєра проєктованої системи є отримання потоку просторових даних у реальному часі та їх перевірка на предмет математичної повноти й валідності. Так як система безконтактного керування орієнтована на використання координатного оптичного сенсора, класичний захват плоских двовимірних кадрів замінюється подійно-орієнтованим перехопленням структурованих масивів тривимірних координат [12].

Логіка цього етапу побудована на базі асинхронного принципу обробки сигналів [34]. Замість використання блокуючих циклів опитування пристрою, які призводять до надлишкового завантаження обчислювального ядра та збільшують загальну часову затримку людино-машинного інтерфейсу, система реагує на події генерації нових просторових зрізів даних [30]. Графічну інтерпретацію цієї логіки у вигляді блок-схеми алгоритму захоплення та первинної валідації просторових даних наведено у Додатку А.

Процес первинної обробки та просторової фільтрації вхідного потоку даних містить таку послідовність кроків:

1) Ініціалізація інтерфейсу зв'язку. Встановлення асинхронного з'єднання з координатною системою сенсора та реєстрація програмного обробника подій, який перебуває у стані очікування оновлення даних [34].

2) Перехоплення поточного зрізу даних. При зміні положення об'єкта маніпуляцій у робочому просторі сенсор формує масив, який містить поточні координати виявлених точок та анатомічних орієнтирів кисті оператора [32].

3) Фільтрація за критерієм наявності об'єкта. Алгоритм аналізує структуру зрізу на наявність цільового маніпулятора (долоні). Якщо у робочій зоні не зафіксовано жодного об'єкта, поточний ітераційний цикл примусово переривається, а система повертається у стан очікування наступного просторового зрізу. Це дозволяє миттєво відсікати випадкові завади та динамічні шуми навколишнього середовища [12].

4) Перевірка цілісності структурної моделі. Для виявленого маніпулятора виконується математична перевірка повноти просторового графа. Алгоритм

здійснює послідовний ітераційний обхід усіх структурних елементів (пальців та суглобових з'єднань) для підтвердження наявності просторових координат X, Y, Z для кожної з ключових точок [24].

5) Агрегація та математична стабілізація просторового вектора. Якщо геометрична структура є повною, тривимірні координати вилучаються та об'єднуються в один фіксований вектор ознак, розмірність якого є строго детермінованою і дорівнює 63 дескрипторам (21 просторова точка, кожна з яких описується трьома осями координат).

На цьому етапі здійснюється приглушення високочастотного апаратного шуму та мікротремору руки оператора. Для стабілізації значень координат кожної i-ї точки ($i = 1, \dots, 21$) у поточному просторовому зрізі часу t застосовується дискретний фільтр низьких частот на основі експоненційного згладжування [32]. Математична модель фільтрації для кожної з просторових осей має вигляд:

$$\begin{cases} \widehat{X}_i(t) = \alpha \cdot X_i(t) + (1-\alpha) \cdot \widehat{X}_i(t-1) \\ \widehat{Y}_i(t) = \alpha \cdot Y_i(t) + (1-\alpha) \cdot \widehat{Y}_i(t-1) \\ \widehat{Z}_i(t) = \alpha \cdot Z_i(t) + (1-\alpha) \cdot \widehat{Z}_i(t-1) \end{cases}, \tag{2.1}$$

де $X_i(t), Y_i(t), Z_i(t)$ — сирі просторові координати i-го суглоба, отримані від сенсора в момент часу t; $\widehat{X}_i(t), \widehat{Y}_i(t), \widehat{Z}_i(t)$ — згладжені (відфільтровані) координати, що записуються у фінальний вектор ознак; $\widehat{X}_i(t-1), \widehat{Y}_i(t-1), \widehat{Z}_i(t-1)$ — результати згладжування на попередньому кроці обробки; α — безрозмірний коефіцієнт згладжування ($\alpha \in (0,1]$), який визначає баланс між чутливістю системи та ступенем придушення високочастотних коливань.

Вибір оптимального значення коефіцієнта α дозволяє забезпечити гладкість траєкторії руху моделі без внесення відчутної фазової затримки в контур людино-машинної взаємодії.

2.2.2 Алгоритм формування ознакового простору на основі скелетной моделі долоні

Після успішного завершення етапу збору та первинної фільтрації просторових координат, сформований 63-вимірний вектор надходить до модуля структурної абстракції. Задачею цього алгоритмічного рівня є перетворення абсолютних метричних координат сенсора у нормалізований, інваріантний до просторового зсуву та масштабу ознаковий простір, який є оптимальним для подальшої класифікації математичними моделями [24].

Пряме використання сирих просторових координат X, Y, Z є неефективним, тому що їхні значення лінійно залежать від фізичного розташування руки оператора відносно оптичного центру пристрою, а також від індивідуальних анатомічних розмірів кисті (довжини пальців, ширини долоні) [22]. Для досягнення властивості інваріантності в алгоритмі реалізується двоетапна математична трансформація: просторова центризація та конвеєрна стандартизація ознак [24].

Крок 1 – просторова центризація. Математична логіка першого етапу полягає в переносі початку глобальної системи координат сенсора у локальний опорний центр долоні. Як базову точку відліку (орієнтир) обрано просторові координати центру долоні ($X_{palm}, Y_{palm}, Z_{palm}$), які були отримані як 21-ша анатомічна точка [32].

Математичне перетворення для кожної i-ї точки вектора ознак ($i = 1, \dots, 21$) здійснюється шляхом розрахунку відносного просторового зсуву за формулами:

$$\begin{cases} X_i' = X_i - X_{palm} \\ Y_i' = Y_i - Y_{palm} \\ Z_i' = Z_i - Z_{palm} \end{cases}, \tag{2.2}$$

де X_i, Y_i, Z_i — вихідні згладжені просторові координати i-ї анатомічної точки руки, отримані з модуля попередньої обробки; $X_{palm}, Y_{palm}, Z_{palm}$ — абсолютні координати геометричного центру долоні (опорного орієнтира), що виступають новим локальним початком відліку;

X_i', Y_i', Z_i' — трансформовані координати i-ї точки у локальній системі координат, прив'язаній до долоні оператора.

Ця операція гарантує, що якщо оператор зміщує руку в просторі праворуч, ліворуч або наближає її до сенсора, відносні вектори всередині сформованого ознакового простору залишаються незмінними, що підвищує стійкість людино-машинного інтерфейсу [12].

Крок 2 – стандартизація ознак**.** Для забезпечення інваріантності до анатомічних розмірів руки (масштабування) та оптимізації роботи градієнтних методів навчання класифікаторів, відцентрований вектор ознак піддається процедурі статистичної стандартизації [24, 26].

Кожен дескриптор k у фінальному векторі ($k = 1, \dots, 63$) трансформується на основі заздалегідь розрахованих математичних параметрів розподілу ознак за формулою:

$$z_k = \frac{x_k' - \mu_k}{\sigma_k}, \tag{2.3}$$

де x_k' — поточне значення відцентрованої координати; μ_k — математичне сподівання (середнє арифметичне значення) k-ї ознаки, отримане на етапі навчання моделей; σ_k — середнє квадратичне відхилення k-ї ознаки; z_k — фінальна стандартизована ознака, що входить до результуючого ознакового простору $Z = [z_1, z_2, \dots, z_{63}]$ [26].

Графічну інтерпретацію цієї логіки у вигляді блок-схеми алгоритму трансформації та стандартизації ознакового простору наведено у Додатку А. Сформований стандартизований вектор Z є повністю підготовленим для передачі в паралельні гілки статичного та динамічного розпізнавання [23].

2.2.3 Розпізнавання статичних і динамічних жестів

Сформований на попередньому етапі стандартизований вектор ознак $Z = [z_1, z_2, \dots, z_{63}]$ надходить до центрального елемента обчислювального конвеєра — інтелектуального модуля класифікації та прийняття рішень. Математична логіка цього підрозділу базується на концепції паралельного гібридного розпізнавання, що дозволяє одночасно ідентифікувати як миттєві просторові конфігурації долоні (статичні пози), так і часові траєкторії руху руки в просторі (динамічні свайпи) [23].

Розгалуження алгоритму на дві ізольовані обчислювальні гілки реалізується за таким принципом:

Гілка 1 – статична класифікація. Статичний класифікатор працює без урахування фактора часу та обробляє кожен кадр незалежно від попередніх. Поточний 63-вимірний вектор $Z(t)$ у момент часу t передається на вхід навченої моделі машинного навчання [22].

Математична функція класифікатора f_{static} відображає вхідний вектор у дискретну множину ймовірностей належності до одного з M заздалегідь визначених статичних класів:

$$P\big(\text{Class}_m \mid Z(t)\big) = f_{static}\big(Z(t)\big), \quad m = 1, \dots, M, \tag{2.4}$$

де $P\big(\text{Class}_m | Z(t)\big)$ — апостеріорна ймовірність того, що поточний вектор ознак належить до m-го статичного класу; $Z(t)$ — стандартизований 63-вимірний вектор ознак у поточний зріз часу t; f_{static} — математична функція базового класифікатора машинного навчання; M — загальна кількість заздалегідь визначених статичних жестів-поз у системі.

На основі розрахованого вектора ймовірностей приймається рішення щодо фінального розпізнаного класу за критерієм максимуму функції правдоподібності :

$$C_{static}(t) = \arg\max_m P\left(\text{Class}_m | Z(t)\right), \tag{2.5}$$

де $C_{static}(t)$ — результуючий індекс розпізнаного статичного класу в момент часу t, який визначається за критерієм максимуму функції ймовірності, за умови, що виконується нерівність $P\left(\text{Class}_m \middle| Z(t)\right) \geq P_{threshold}$ (де $P_{threshold}$ — встановлений внутрішній поріг упевненості системи) [26, 27].

Якщо максимальна ймовірність перевищує встановлений внутрішній поріг упевненості системи $P_{threshold}$ (наприклад, $P \geq 0.85$), то зріз кадру вважається успішно класифікованим, а ідентифікований індекс статичної пози передається далі за конвеєром [23].

Гілка 2 – динамічна класифікація. Для розпізнавання жестів, значення яких визначається траєкторією руху (наприклад, швидкі свайпи вліво чи вправо), миттєвого зрізу даних недостатньо. Алгоритм динамічної гілки використовує концепцію ковзного часового вікна фіксованої довжини N кадрами [25].

Процес динамічного аналізу складається з таких кроків:

1) Поточний вектор $Z(t)$ асинхронно додається до двобічної черги (буфера кадрів) обмеженого розміру [28, 30]. При надходженні нового кадру найстаріший кадр автоматично видаляється з пам'яті за принципом First-In, First-Out.

2) Алгоритм перевіряє умову заповненості часового вікна. Доки кількість накопичених кадрів менша за фіксоване число N ($N = 20$), динамічний аналіз не запускається.

3) При повному заповненні буфера формується двовимірна матриця ознак T розмірністю $N \times 63$, яка відображає динаміку зміни положення руки за останній часовий інтервал:

$$T = [Z(t - N + 1), Z(t - N + 2), \dots, Z(t)]^T, \tag{2.6}$$

де T — матрична послідовність ознак, що подається на вхід нейромережі; $Z(\tau)$ — стандартизований 63-вимірний вектор ознак для часового зрізу τ; t — поточний момент часу (індекс останнього отриманого кадру); N — довжина часового вікна обробки даних ($N = 20$).

4) Сформована матриця часового ряду T вирівнюється (трансформується у плоский вектор) та передається на вхід глибокої нейромережевої моделі часового аналізу $f_{dynamic}$ (наприклад, рекурентної архітектури LSTM або GRU) [21, 25]. На виході нейромережа повертає індекс розпізнаного динамічного жесту:

$$C_{dynamic}(t) = f_{dynamic}(T), \tag{2.7}$$

де $C_{dynamic}(t)$ — обчислений індекс розпізнаного динамічного жесту (напрямку руху) в момент часу t; $f_{dynamic}$ — оператор класифікації глибокої нейромережевої моделі часового аналізу; T — вхідна матриця часової послідовності ознак.

Графічну інтерпретацію представленої логіки розгалуження та паралельного обчислення у вигляді блок-схеми алгоритму гібридної статико-динамічної класифікації жестів наведено у Додатку В. Сформовані індекси класів C_{static} та $C_{dynamic}$ передаються до фінального модуля інтерпретації для генерації команд керування [27].

2.2.4 Інтерпретація жестів у команди керування та фільтрація помилкових спрацювань

Завершальним етапом роботи алгоритмічного конвеєра є трансформація зафіксованих індексів статичних (C_{static}) та динамічних ($C_{dynamic}$) жестів у детерміновані цифрові команди керування. Основним дестабілізуючим фактором на цьому рівні є «брязкіт» (chattering/flickering) розпізнаних класів на межі кадрів, що виникає через мікротремор руки або оптичні спотворення, і може призвести до хибних повторних активацій виконавчого обладнання [27].

Для забезпечення завадостійкості та плавності людино-машинної взаємодії в алгоритмі реалізовано часову фільтрацію двох типів:

– часовий дебаунсинг для статичних поз**.** Статичний жест вважається істинно активованим лише тоді, коли його індекс залишається незмінним протягом заданого часового вікна (порогу стабільності) $K_{debounce}$ кадрами підряд.

Математично стан лічильника стабільності $S(t)$ для поточного кадру t описується так:

$$S(t) = \begin{cases} S(t-1)+1, & \text{якщо } C_{static}(t) = C_{static}(t-1) \text{ та } C_{static}(t) \neq 0 \\ S(t) = 1, & \text{якщо } C_{static}(t) \neq C_{static}(t-1) \text{ та } C_{static}(t) \neq 0 \\ S(t) = 0, & \text{якщо } C_{static}(t) = 0 \end{cases}, \quad (2.8)$$

де $C_{static}(t)$ — поточний розпізнаний індекс пози;

$S(t-1)$ — значення лічильника на попередньому кадрі.

Фінальна команда генерується лише в момент, коли $S(t) = K_{debounce}$. Після цього лічильник блокується, що запобігає дублюванню сигналів при тривалому утриманні пози користувачем [31].

– часове заморожування для динамічних траєкторій**.** Оскільки динамічний жест розпізнається нейромережею миттєво на основі аналізу буфера, він не потребує накопичення так званого дебаунсу. Проте швидкий рух руки назад після виконання свайпу може спровокувати повторне помилкове спрацювання моделі [23]. Для запобігання цьому впроваджено алгоритмічний таймер блокування $T_{cooldown}$ [27]:

$$\text{Статус системи} = \begin{cases} \text{Заблоковано}, & \text{якщо } t - t_{act} \leq K_{cooldown} \\ \text{Активно}, & \text{якщо } t - t_{act} > K_{cooldown} \end{cases}, \quad (2.9)$$

де t_{act} — зріз часу (номер кадру) останньої успішної активації динамічної команди; $K_{cooldown}$ — тривалість «заморожування» контуру класифікації, що дорівнює 50 кадрам.

Верифіковані фільтрами індекси жестів проходять через рівень контекстної адаптації [1]. Залежно від обраного режиму роботи системи, один і той самий жест інтерпретується в унікальну команду. Кінцевим продуктом інтерпретації є компактний командний байт-код, що надсилається через послідовний порт [29, 33].

Сформований байт-код через асинхронні процедури виведення передається у системний буфер послідовного інтерфейсу для відправки на мікроконтролерний виконавчий блок [31].

2.3 Інформаційне забезпечення

Ефективне та завадостійке функціонування інтелектуальної системи безконтактного керування безпосередньо залежить від раціональної організації її інформаційного забезпечення. Інформаційне забезпечення являє собою сукупність уніфікованих систем класифікації, структурованих масивів просторових даних, а також логічних контурів обміну інформаційними потоками між оператором, обчислювальним комплексом штучного інтелекту та виконавчими мікроконтролерами [1, 24].

Організація інформаційних зв’язків системи базується на послідовній трансформації та фільтрації даних на кожному етапі обчислювального конвеєра [34]. Логічна структура інформаційних потоків забезпечує взаємодію між трьома ключовими рівнями: фізичного сприйняття – зчитування тривимірних координат суглобів кисті, інтелектуального аналізу – математична центризація, стандартизація та класифікація ознак моделями машинного навчання) та виконавчої інтерпретації – мапування розпізнаних станів у вихідні керуючі байт-коди [23, 32].

2.3.1 Опис вхідної та вихідної інформації

Відповідно до розробленої архітектури людино-машинного інтерфейсу, інформаційні потоки системи розподіляються на вхідну інформацію, внутрішні інформаційні стани та вихідну інформацію, що генерується як результат обчислень [12]. Структуру вхідної інформації наведено в таблиці 2.1

Таблиця 2.1 –Характеристики дескрипторів вхідної інформації

Індекс точок	Анатомічний орієнтир (вузол графа)	Параметр	Тип даних	Одиниці виміру	Діапазон значень
$1 \ldots 4$	Суглоби та кінчик великого пальця	X_k, Y_k, Z_k	Float	Міліметри (мм)	$[-500.0; +500.0]$
$5 \ldots 8$	Суглоби та кінчик вказівного пальця	X_k, Y_k, Z_k	Float	Міліметри (мм)	$[-500.0; +500.0]$
$9 \ldots 12$	Суглоби та кінчик середнього пальця	X_k, Y_k, Z_k	Float	Міліметри (мм)	$[-500.0; +500.0]$
$13 \ldots 16$	Суглоби та кінчик підмізинного пальця	X_k, Y_k, Z_k	Float	Міліметри (мм)	$[-500.0; +500.0]$
$17 \ldots 20$	Суглоби та кінчик мізинця	X_k, Y_k, Z_k	Float	Міліметри (мм)	$[-500.0; +500.0]$
21	Геометричний центр долоні	$X_{palm}, Y_{palm}, Z_{palm}$	Float	Міліметри (мм)	$[-500.0; +500.0]$

Первинними вхідними даними є часова послідовність просторових зрізів, що описують геометричну конфігурацію кисті руки оператора [32]. Ці дані надходять від координатного оптичного сенсора у вигляді структурованого масиву тривимірних координат X, Y, Z для кожної з 21 ключової точки з частотою дискретизації від 60 до 120 Гц [30, 34]. Загальний обсяг одного інформаційного зрізу становить строго 63 числових дескриптори типу floating-point [24]. Деталізовані характеристики дескрипторів вхідного потоку наведено в таблиці 2.1.

В процесі динамічної обробки даних система формує внутрішні інформаційні прапорці, що координують роботу контуру обчислень [23]:

– стан очікування (IDLE): об'єкт маніпуляцій відсутній, конвеєр заблоковано;

– накопичення (ACCUMULATION): відбувається циклічне наповнення часового журналу FIFO для динамічного аналізу [28];

– стабілізація (DEBOUNCING): очікування підтвердження незмінності статичної пози протягом заданої кількості кадрів [31];

– заморожування (COOLDOWN): тимчасове часове блокування контуру класифікації після активації команди для уникнення повторних хибних спрацювань [27].

Вихідна інформація системи є двокомпонентною і розділяється за фізичним та логічним призначенням [11, 33]. Перший компонент — це фізичний керуючий потік детермінованих однобайтових символьних команд для передачі через послідовний порт мікроконтролеру [31]. Другий компонент — інформаційно-моніторинговий потік рядкових повідомлень для візуального логування дій оператора в режимі реального часу [27]. Структуру вихідної інформації наведено в таблиці 2.2.

Таблиця 2.2 – Структура та призначення вихідних інформаційних потоків

Код стану	Назва розпізнаного жесту	Формат	Адрес призначення	Дія виконавчого пристрою
b'A'	One Finger (Ch A)	Byte	COM-порт	Світлодіод LED: ON/OFF
b'B'	Two Fingers (Ch B)	Byte	COM-порт	Модуль реле: ON/OFF
b'C'	Three Fingers (Ch C)	Byte	COM-порт	Перемикання (90° / 0°)
b'R'	Swipe Right (ALL ON)	Byte	COM-порт	Ввімкнення всіх каналів°
b'L'	Swipe Left (ALL OFF)	Byte	COM-порт	Вимкнення всіх каналів °
String	Рядок логу подій системи	ASCII	Консоль	Оновлення текстового статусу,

2.3.2 Проєктування даталогічної моделі

Розроблений програмно-апаратний комплекс функціонує в архітектурі реального часу без використання класичних реляційних чи нереляційних систем керування базами даних, проєктування глобальної даталогічної моделі фокусується на описі логічної структури файлів конфігурацій збереженого стану знань моделей штучного інтелекту та динамічних інформаційних об'єктів всередині оперативної пам'яті ЕОМ [24].

Логічна модель даних системи розділяється на два взаємопов'язані сегменти: статичні структури довготривалого зберігання на диску та динамічні структури обробки даних у часовому просторі [28].

1) логічна структура збережених конфігураційних файлів. Системне середовище оперує трьома бінарними об'єктами зберігання знань, які завантажуються в пам'ять при ініціалізації:

– параметри стандартизації ознак: логічний файл, що містить два одновимірні вектори розмірністю 63 елементи — вектор математичних сподівань $M = [\mu_1, \dots, \mu_{63}]$ та вектор стандартних відхилень $\Sigma = [\sigma_1, \dots, \sigma_{63}]$. Ці параметри використовуються для миттєвого лінійного перетворення вхідних дескрипторів [24].

– вагові коефіцієнти статичного класифікатора: логічна структура, яка зберігає математичні гіперплощини та матриці опорних векторів розбиття простору поз на 3 детерміновані класи [22].

– тензорна структура динамічної нейромережі: ієрархічний файл, що описує архітектуру шарів (рекурентних або повнозв'язних) та матриці вагових коефіцієнтів синаптичних зв'язків для часового аналізу послідовностей [21].

2) логічна структура оперативних даних та журналів. Для забезпечення обчислень алгоритмів згладжування, дебаунсингу та нейромережевого прогнозування, в оперативній пам'яті проєктуються такі логічні змінні та структури:

– часовий журнал послідовностей: лінійна матрична структура фіксованого обсягу 20×63, яка динамічно оновлюється при кожному кроці системи. Об'єкт забезпечує збереження часового вікна останніх 20 валідних зрізів координат для розпізнавання свайпів [28, 30].

– мапінгові логічні структури: детерміновані асоціативні масиви (словники соответствія), які пов'язують вихідні індекси математичних класів (0, 1, 2) з текстовими назвами жестів для інтерфейсу користувача та відповідними ASCII-символами команд послідовного порту (b'A', b'B', b'C', b'R', b'L') [33].

Графічну інтерпретацію взаємозв'язків між цими інформаційними сутностями в межах оперативної пам'яті ЕОМ – діаграму структури даних наведено у Додатку Г.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Вибір апаратних і програмних засобів

На основі архітектури системи, описаної у попередньому розділі, було проведено аналіз та вибір відповідних апаратних і програмних компонентів для фізичної реалізації людино-машинного інтерфейсу [12, 24]. Головними критеріями під час вибору технологічного стеку були:

– висока швидкість обробки просторових даних у режимі реального часу [32];

– апаратна сумісність між модулями збору даних та виконавчими пристроями [27];

– надійність компонентів під час тривалої роботи [31];

– наявність широкої підтримки з боку спільноти розробників та відкритих бібліотек машинного навчання [34].

Для логічного структурування опису, розроблену систему було розділено на три базові рівні: мікроконтролерний (виконавчий), рівень комп'ютерного зору (сенсорний) та рівень програмного забезпечення (інтелектуальний) [23].

3.1.1 Вибір апаратного забезпечення мікроконтролерного рівня

Враховуючи характер завдань, а саме безперервне зчитування керуючих байтів через послідовний порт та миттєве керування трьома каналами навантаження (світлодіод, реле, сервопривод), базою виконавчої частини було обрано плату розширення DFRobot Arduino Expansion Shield for Raspberry Pi, зображену на рисунках 3.1-3.2.

Цей модуль побудований на базі мікроконтролера ATmega32U4, який використовується в оригінальній платі Arduino Leonardo [15]. Ключовою перевагою вибору саме ATmega32U4 є вбудована апаратна підтримка інтерфейсу USB 2.0. Це дозволяє мікроконтролеру взаємодіяти з комп'ютером без використання додаткових мікросхем-перетворювачів (наприклад, CH340 або FTDI), що суттєво зменшує затримки (latency) при передачі байт-кодів команд від Python-скрипта до виконавчого пристрою [28, 31].

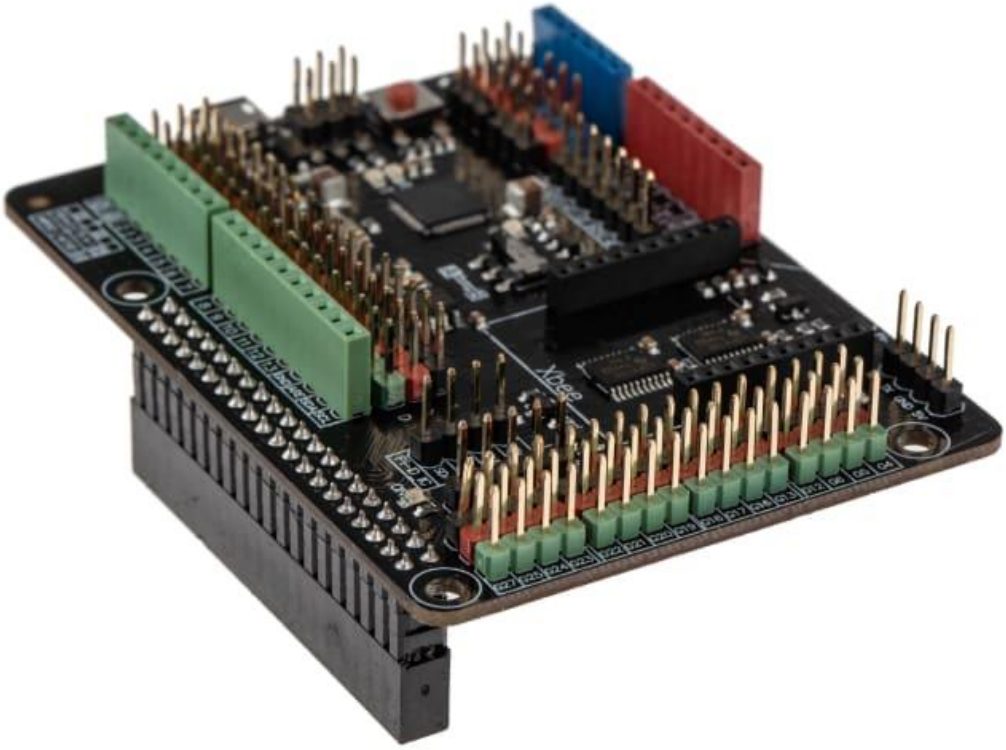


Рисунок 3.1 – Загальний вигляд плати DFRobot Arduino Expansion Shield

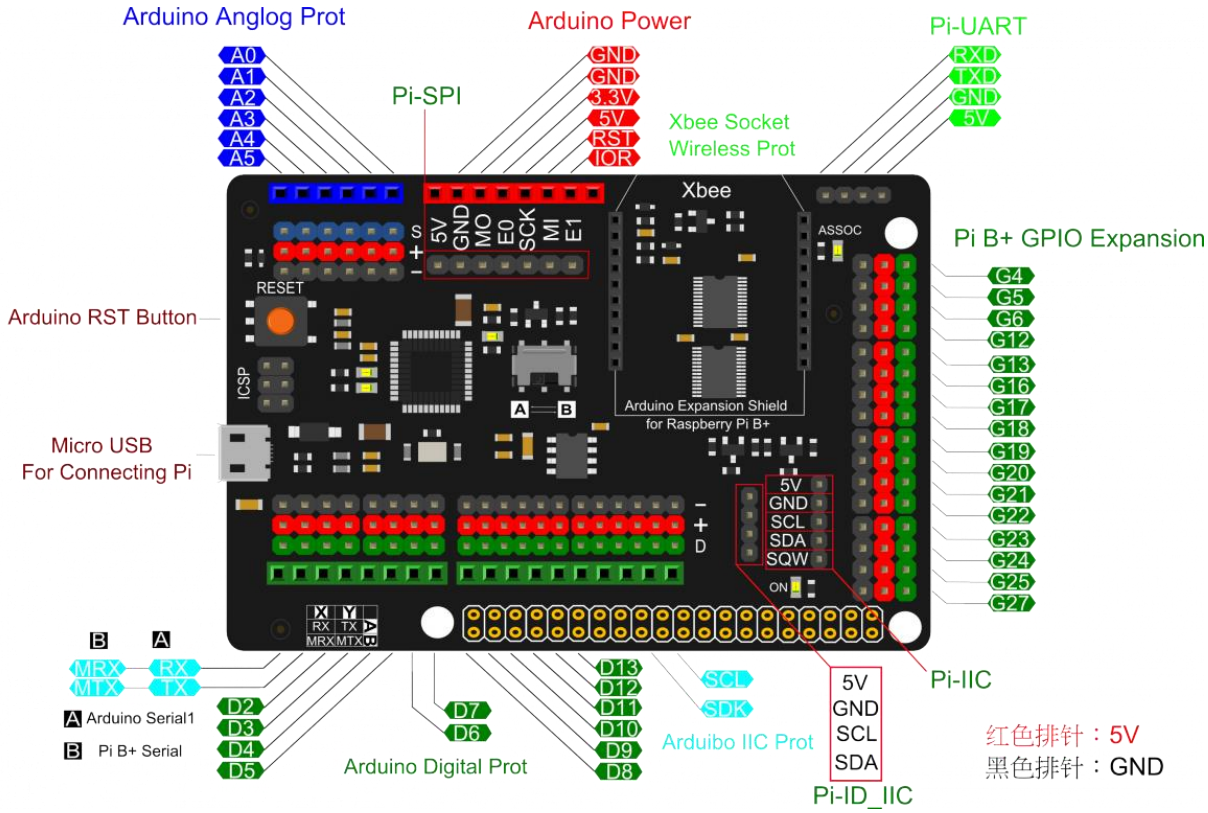


Рисунок 3.2 – Розпіновка плати DFRobot Arduino Expansion Shield

Основні технічні характеристики обраного мікроконтролерного модуля наведено у таблиці 3.1.

Таблиця 3.1 – Технічні характеристики мікроконтролерного модуля (ATmega32U4)

Характеристика	Значення
Мікроконтролер	ATmega32U4 (архітектура AVR)
Робоча напруга	5 В
Цифрові входи/виходи	20 (з них 7 підтримують ШІМ)
Аналогові входи	12
Тактова частота	16 МГц
Обсяг Flash-пам'яті	32 КБ (з яких 4 КБ використовує bootloader)
Обсяг SRAM	2.5 КБ

Відповідно до розробленого програмного коду, до мікроконтролера підключається наступна периферія [33]:

1) Світлодіод (LED) (Pin 3): Використовується як візуальний індикатор першого каналу навантаження (Channel A).

2) Електромагнітне реле (Pin 4): Виступає комутатором для керування потужним зовнішнім навантаженням (Channel B). Модуль дозволяє керувати ланцюгами змінного або постійного струму, фізично ізолюючи слабкострумову логіку контролера від високої напруги.

3) Сервопривод SG90 (Pin 9): Використовується для виконання просторових маніпуляцій (Channel C). Це компактний механізм, здатний підтримувати заданий кут повороту (0° або 90° відповідно до команд системи).

Для стабільного живлення SG90 та модуля реле використовується лінія 5V мікроконтролерної плати.

3.1.2 Вибір підсистеми комп'ютерного зору

Для забезпечення безконтактного захоплення просторових координат рук було обрано спеціалізований оптичний сенсор Leap Motion Controller, зображений на рисунку 3.3

Рисунок 3.3 – Зовнішній вигляд Leap Motion Controller

На відміну від звичайних RGB-камер, які вимагають значних обчислювальних ресурсів центрального процесора для запуску важких моделей локалізації (наприклад, MediaPipe), Leap Motion використовує дві монохромні інфрачервоні камери та три інфрачервоні світлодіоди [32]. Побудова скелетної моделі відбувається за допомогою фірмового низькорівневого API (LeapC), що повністю розвантажує процесор комп'ютерної системи і передає у програму вже готові 3D-координати анатомічних точок [34].

Ключовою перевагою сенсора є його висока точність та частота дискретизації, що є важливим для формування часових послідовностей динамічних жестів. Основні параметри сенсора наведено у таблиці 3.2.

Таблиця 3.2 – Технічні характеристики Leap Motion Controller

Характеристика	Значення
Технологія відстеження	Інфрачервона оптична стереоскопія
Частота кадрів (Framerate)	від 60 до 120 Гц (до 200 Гц у режимі високої продуктивності)
Поле зору (Field of View)	$150° \times 120°$
Інтерфейс підключення	USB 2.0 / USB 3.0
Кількість відстежуваних точок	21 анатомічна точка на кожну руку (суглоби та центр долоні)

Завдяки цим характеристикам сенсор підходить для формування вхідних векторів ознак розмірністю 63 елементи (21 точка по 3 координати X, Y, Z) [24].

3.1.3 Вибір програмного середовища та інструментарію розробки

Програмний комплекс складається з двох ізольованих, але взаємопов'язаних рівнів: інтелектуального (серверного) та мікроконтролерного (клієнтського). Для їх ефективної реалізації було обрано різні середовища розробки [23].

Середовище розробки логіки керування. Основна обчислювальна логіка системи, збір датасетів, навчання моделей штучного інтелекту та графічний інтерфейс користувача реалізовані мовою програмування Python 3. Як інтегроване середовище розробки (IDE) було обрано Visual Studio Code (VS Code), зображений на рисунку 3.4, через його високу продуктивність, підтримку розширень для роботи з даними та вбудований термінал.

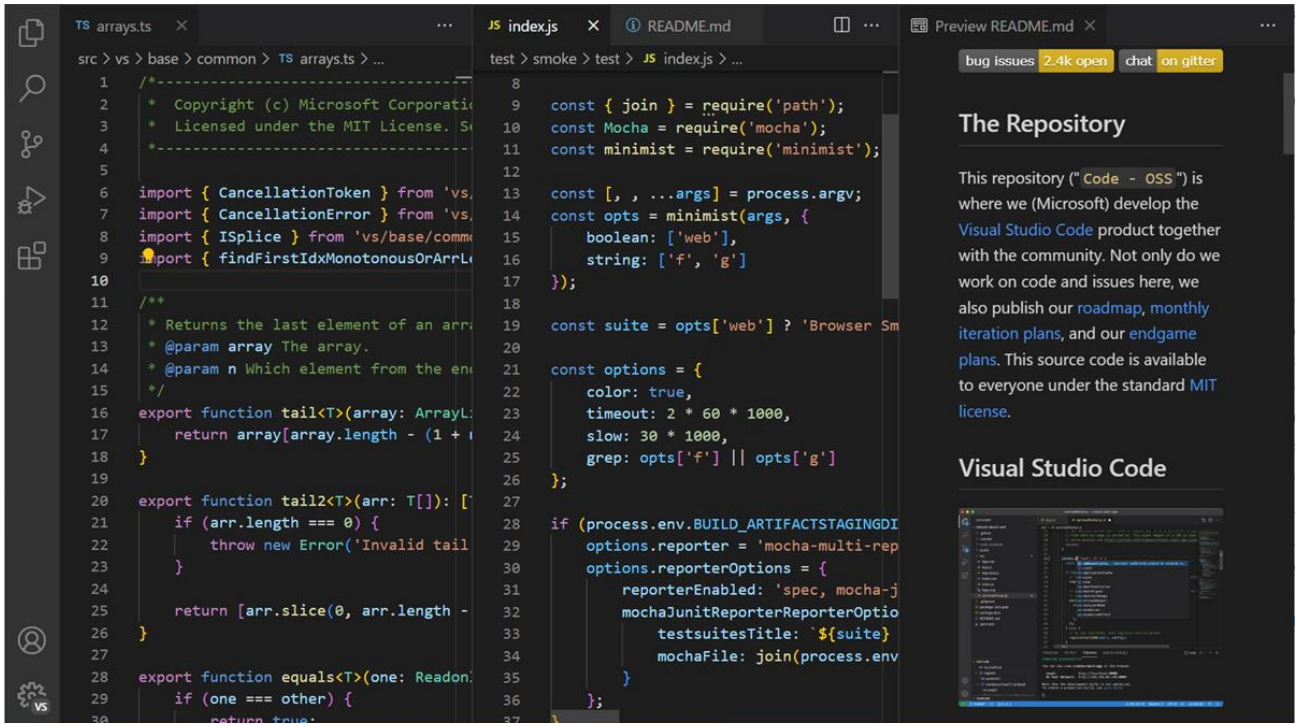


Рисунок 3.4 – Робочий простір середовища Visual Studio Code

Для забезпечення математичних обчислень та машинного навчання використано наступні спеціалізовані бібліотеки:

– TensorFlow / Keras для побудови рекурентної глибокої нейромережі (LSTM). Бібліотека забезпечує тензорне обчислення матриць часових рядів та має вбудовані алгоритми оптимізації [21, 25].

– scikit-learn для реалізації статичного класифікатора (MLPClassifier), а також для стандартизації координатного простору (StandardScaler) [22, 24].

– OpenCV (cv2) для побудови асинхронного графічного інтерфейсу (модуль DashboardUI) та вікон збору даних [27].

– PySerial для організації швидкого обміну даними між Python-скриптом та COM-портом (USB) плати Arduino [31].

Середовище розробки мікроконтролера. Для написання коду виконавчого пристрою та прошивки чипа ATmega32U4 використовувалось офіційне середовище Arduino IDE v2 продемонстроване на рисунку 3.5 [33]. Воно побудоване на базі компілятора GCC і дозволяє використовувати оптимізований діалект мови C/C++.

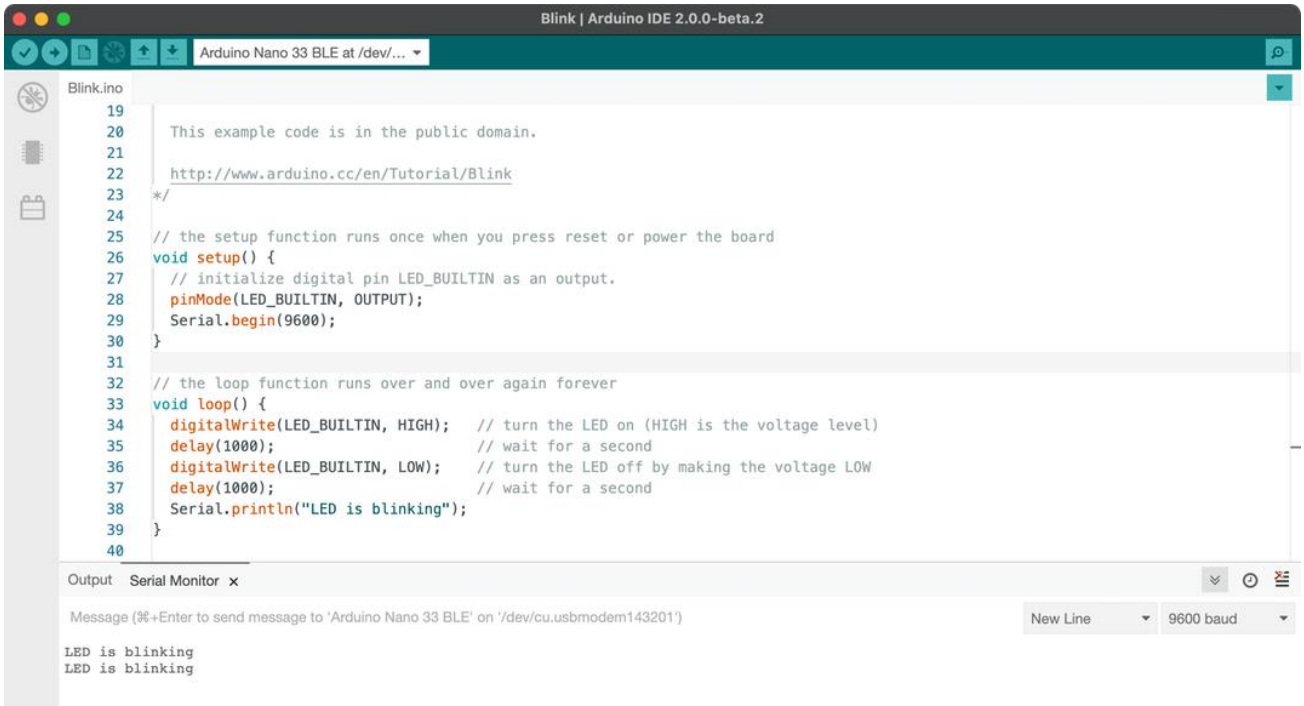


Рисунок 3.5 – Інтерфейс середовища Arduino IDE

У програмному коді мікроконтролера використано вбудовану бібліотеку <Servo.h>, яка абстрагує складні апаратні налаштування таймерів контролера та

дозволяє керувати кутом сервоприводу за допомогою простого методу інкапсуляції.

3.2 Реалізація ключових модулів системи

Практична реалізація людино-машинного інтерфейсу безконтактного керування базується на принципах модульної архітектури та слабкого зв'язку (loose coupling) між обчислювальними компонентами [24]. Такий підхід дозволяє ізолювати процеси фізичної реєстрації просторових координат, процедури статистичного аналізу, контури інтелектуального розпізнавання та блоки безпосередньої взаємодії з апаратним периферійним обладнанням [27, 34].

Програмне забезпечення системи розроблено у вигляді ієрархічного комплексу взаємопов'язаних модулів мовами Python та C++, функціонування яких підпорядковане єдиній логіці кінцевого автомата. Базовим обчислювальним середовищем виступає ЕОМ, яка виконує ресурсомісткі математичні операції класифікації та часового аналізу, тоді як мікроконтролерний модуль виконує роль інертного виконавчого органу, що детерміновано відпрацьовує отримані інструкції в асинхронному режимі [23, 31].

3.2.1 Модулі збору даних та навчання класифікаторів

Формування репрезентативної вибірки (датасету) та подальше тренування моделей штучного інтелекту є фундаментом для забезпечення заданої точності розпізнавання жестів оператора. Процес автоматизованого збору ознакового простору розділено на два ізольовані програмні модулі: collect_data.py (для статичних поз-команд) та collect_dynamic_data.py (для просторово-часових траєкторій руху) [32].

Програмна логіка скрипта collect_data.py базується на створенні об'єкта DataCollectorListener, який успадковує функціонал базового класу leap.Listener. Асинхронний обробник подій on_tracking_event перехоплює просторові координати від інфрачервоного сенсора, виконує обхід вкладеними циклами всіх 5

пальців кисті оператора (hand.digits), вилучає тривимірні координати кінців кожної з 4 фаланг (bone.next_joint) та примусово додає 21-шу опорну точку центру долоні (hand.palm.position) [24].

Сформований 63-вимірний вектор числових дескрипторів лінійно поєднується з поточною міткою класу (CURRENT_CLASS) і записується у файл data/static_gestures.csv. Візуальний контроль процесу збору реалізовано засобами графічної бібліотеки OpenCV.

```
class DataCollectorListener(leap.Listener):
    def __init__(self):
        super().__init__()
        self.current_landmarks = None

    def on_tracking_event(self, event):
        if len(event.hands) > 0:
            hand = event.hands[0]
            landmarks = []

            for digit in hand.digits:
                for bone in digit.bones:
                    landmarks.extend([bone.next_joint.x, bone.next_joint.y, bone.next_joint.z])

            landmarks.extend([hand.palm.position.x, hand.palm.position.y, hand.palm.position.z])

            self.current_landmarks = landmarks
        else:
            self.current_landmarks = None
```

Рисунок 3.6 – Структура обробника подій збору просторових дескрипторів

Для навчання статичного класифікатора було зібрано масив даних обсягом 3000 семплів (по 1000 зразків на кожен із 3 цільових класів):

– Клас 0 — Жест «One Finger (Ch A)»;

– Клас 1 — Жест «Two Fingers (Ch B)»;

– Клас 2 — Жест «Three Fingers (Ch C)».

Навчання статичного класифікатора (train_static.py). Для розпізнавання статичних поз використано архітектуру нейромережі типу багатошаровий перцептрон (MLPClassifier), реалізовану в бібліотеці scikit-learn [22]. Датасет розділяється на тренувальну та тестову вибірки у співвідношенні 80/20. Нейромережа конфігурується двома прихованими повнозв'язними шарами (128 та 64 нейрони відповідно), активаційною функцією relu та максимальним числом

ітерацій градієнтного спуску max_iter=500. Натренована модель бінаризується та зберігається на диск у файл models/static_model.pkl за допомогою модуля pickle.

```
# 3. НАВЧАННЯ МОДЕЛІ
print("\n2. Почалось навчання нейромережі.")
# hidden_layer_sizes=(128, 64)
clf = MLPClassifier(hidden_layer_sizes=(128, 64), max_iter=500, activation='relu', random_state=42)
clf.fit(X_train, y_train)
```

Рисунок 3.7 – Конфігурування структури багатошарового перцептрона в Python

Модуль збору динамічних послідовностей та навчання LSTM-мережі**.** Розпізнавання свайпів вимагає фіксації зміни координат у часі. Скрипт collect_dynamic_data.py забезпечує циклічний запис пакетів даних фіксованої довжини у FRAMES_PER_SEQUENCE = 20 кадрів зі швидкістю приблизно 50 FPS (крок затримки time.sleep(0.02)) [25, 28]. Усі 20 кадрів вирівнюються в один довгий плоский вектор розмірністю $20 \times 63 = 1260$ числових елементів, у кінець якого додається мітка динамічного класу (0 — свайп праворуч, 1 — свайп ліворуч). Загальний обсяг зібраного датасету для файлу data/dynamic_gestures.csv становить 400 рухів [30].

Скрипт навчання train_dynamic.py реалізує конвеєрну стандартизацію ознак за допомогою класу StandardScaler. Матриця ознак X трансформується у тривимірний тензор із формою [-1, 20, 63], що відповідає вимогам рекурентних шарів [24]. Архітектура глибокої нейромережі, побудована на базі TensorFlow/Keras, містить рекурентний шар LSTM на 64 комірки, шар виключення нейронів Dropout(0.2) для регуляризації та запобігання перенавчанню, проміжний повнозв'язний шар Dense (32 нейрони) та вихідний шар Softmax для розрахунку розподілу ймовірностей за класами [21, 25]. Модель оптимізується алгоритмом Adam (learning_rate=0.001) протягом 50 епох та експортується у форматі models/dynamic_model.h5.

```
print("3. Побудова LSTM-мережі")
model = Sequential([
    LSTM(64, return_sequences=False, activation='relu', input_shape=(FRAMES_PER_SEQUENCE, FEATURES_PER_FRAME)),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(NUM_CLASSES, activation='softmax')
])
```

Рисунок 3.8 – Специфікація тензорних шарів моделі LSTM

3.2.2 Програмне ядро системи та взаємодія з мікроконтролером

Центральним диспетчером людино-машинного інтерфейсу виступає головний обчислювальний скрипт main.py. Його архітектура спроєктована за принципом неблокуючого операційного циклу з високою частотою опитування сенсора (затримка time.sleep(0.01) забезпечує частоту контуру ~100 Гц) [31].

Етап ініціалізації програмного ядра. При запуску системи виконується послідовне завантаження десеріалізованих бінарних об'єктів конфігурації за допомогою pickle (static_model.pkl та scaler.pkl), а також тензорної моделі Keras (dynamic_model.h5). Одночасно ініціюється зв'язок із платою розширення DFRobot на базі ATmega32U4 через екземпляр класу serial.Serial на системному порті COM3 зі швидкістю BAUD_RATE = 9600 [31, 33]. Після успішного встановлення прапорців зв'язку запускається графічний дашборд DashboardUI та асинхронний слухач координатного сенсора SystemListener.

```
def main():
    print("1. Завантаження моделей.")
    try:
        with open(STATIC_MODEL_PATH, 'rb') as f:
            static_clf = pickle.load(f)
        with open(SCALER_PATH, 'rb') as f:
            scaler = pickle.load(f)
        dynamic_model = load_model(DYNAMIC_MODEL_PATH)
    except Exception as e:
        print(f"Помилка завантаження: {e}")
        return

    print(f"2. Підключення до Arduino ({COM_PORT})...")
    try:
        arduino = serial.Serial(COM_PORT, BAUD_RATE, timeout=0.1)
        time.sleep(2)
    except Exception as e:
        print(f"Помилка Arduino: {e}")
        return
```

Рисунок 3.9 – Блок конфігурування системного оточення при старті ядра

Обробка логічних режимів та фільтрація помилкових спрацювань. Програмне ядро здатне функціонувати в одному з двох взаємовиключних режимів (current_mode), перемикання між якими реалізовано через перехоплення коду клавіші m:

– робота контуру STATIC полягає в тому, що сирі просторові координати поточного кадру, якщо їхня розмірність дорівнює 63, перетворюються у двовимірну матрицю (1, 63) та передаються методу predict_proba. Отриманий вектор ймовірностей аналізується за умовою порогу впевненості STATIC_CONFIDENCE_THRESHOLD = 0.90. Якщо поріг подолано, запускається лічильник стабілізації (static_gesture_counter). Лише за умови утримання пози протягом DEBOUNCE_FRAMES = 20 послідовних кадрів, індекс жесту вважається верифікованим і мапується через словник COMMAND_MAP_STATIC у командний байт [23, 31].

```
if conf_static >= STATIC_CONFIDENCE_THRESHOLD:
    ui_gesture_name = gesture_name
    ui_confidence = conf_static * 100

    if predicted_static == stable_static_gesture:
        static_gesture_counter += 1
    else:
        stable_static_gesture = predicted_static
        static_gesture_counter = 1

    if static_gesture_counter == DEBOUNCE_FRAMES:
        if command and command != last_sent_command:
            arduino.write(command)
            ui_log_message = f"Sent: {command.decode()} ({gesture_name})"
            print(ui_log_message)
            last_sent_command = command
else:
    static_gesture_counter = 0
    ui_gesture_name = "Unclear pose"
    ui_confidence = conf_static * 100
```

Рисунок 3.10 – Програмна реалізація фільтра брязкання ознак

– робота контуру DYNAMIC бере вхідні вектори безперервно накопичуються у чергу обмеженої довжини deque(maxlen=20). При заповненні буфера система розраховує евклідову відстань між початковим та кінцевим положенням центру долоні у вікні. Якщо просторовий дельта-зсув перевищує

MIN_MOTION_DISTANCE = 50.0 мм, дані визнаються динамічним актом. Буфер вирівнюється, проходить просторову стандартизацію через scaler.transform і подається на вхід нейромережі LSTM [24, 25]. При перевищенні порогу впевненості DYNAMIC_CONFIDENCE_THRESHOLD = 0.85 генерується вихідний байт, черга очищується, а лічильнику dynamic_cooldown присвоюється значення 50, що повністю заморожує динамічний аналіз на наступні 50 кадрів для запобігання дублюванню команд при поверненні руки [27, 28].

Взаємодія з мікроконтролером та виконання команд. Надсилання керуючого сигналу здійснюється методом arduino.write(command). Завдяки апаратній архітектурі чипа ATmega32U4 плати DFRobot Shield, байт миттєво зчитується у низькорівневому контурі C++ (скрипт прошивки Arduino). Мікроконтролер виконує умовне розгалуження через оператор if (incomingByte == 'X') та змінює стани логічних прапорців state1, state2, state3 [33].

```
void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();

    if (incomingByte == 'A') {
      currentChannel = CHANNEL_1;
      Serial.println("Channel: 1");
    }
    else if (incomingByte == 'B') {
      currentChannel = CHANNEL_2;
      Serial.println("Channel: 2");
    }
    else if (incomingByte == 'C') {
      currentChannel = CHANNEL_3;
      Serial.println("Channel: 3");
    }
    else if (incomingByte == 0 || incomingByte == 1) {
      // Process signal for selected channel
      switch (currentChannel) {
        case CHANNEL_1:
          digitalWrite(pin1, incomingByte);
          break;
        case CHANNEL_2:
          digitalWrite(pin2, incomingByte);
          break;
        case CHANNEL_3:
          digitalWrite(pin3, incomingByte);
          break;
        default:
          break;
      }
      Serial.print("Signal: ");
      Serial.println(incomingByte);
    }
  }
}
```

Рисунок 3.11 – Програма обробки бінарного потоку даних на мікроконтролері

При отриманні статичних символів 'A', 'B', 'C' мікроконтролер виконує інверсію відповідного цифрового виходу (digitalWrite(pin, !state)), що призводить до комутації світлодіода або реле, або змінює кут сервоприводу за допомогою об'єкта myServo.write(). При фіксації глобальних динамічних команд 'R' (свайп праворуч) або 'L' (свайп ліворуч), плата реалізує логіку групового керування, миттєво переводячи всі три незалежні канали навантаження в максимальний активний (HIGH / 90°) або пасивний (LOW / 0°) стан відповідно [31].

3.3 Інтерфейс користувача

Проєктування та практична реалізація інтерфейсу користувача (UI) для системи безконтактного керування на базі жестів підпорядковані базовим ергономічним принципам людино-машинної взаємодії (HCI) [29]. Головною метою розробки графічного дашборду є забезпечення оператора інтуїтивно зрозумілим, високоінформативним та асинхронним візуальним зворотним зв'язком у режимі реального часу, що мінімізує когнітивне навантаження під час маніпуляцій [1, 29].

Згідно з сучасними інженерними підходами до побудови інтерфейсів складних інтелектуальних систем, графічна оболонка DashboardUI (реалізована у файлі dashboard.py) функціонує за принципом інформаційного мінімалізму [29]. Візуалізація поточного стану системи здійснюється на темному однорідному тлі, що запобігає втомі зорового апарату оператора. Для швидкої семантичної орієнтації користувача в інтерфейсі застосовано функціональне кольорове кодування: активні системні режими, рівні впевненості моделей та черги логів розподілені за унікальними колірними каналами палітри RGB, що дозволяє миттєво оцінювати статус обчислювального контуру без детального вчитування у текст [11, 29].

3.3.1 Сценарії взаємодії користувача з системою

Логіка взаємодії оператора з розробленим людино-машинним інтерфейсом детермінована набором призначених для користувача сценаріїв, які охоплюють усі

етапи роботи комплексу — від апаратної ініціалізації до штатного завершення сесії [29, 35]. Усі поведінкові паттерни системи описуються п'ятьма базовими сценаріями:

Сценарій 1. Ініціалізація та очікування об'єкта. При запуску головного скрипта main.py графічне вікно «Intelligent Control System» автоматично переходить у стан очікування. Оскільки координатна модель руки у просторі ще не сформована сенсором Leap Motion, інтерфейс виводить на екран текстове повідомлення ui_gesture_name = "Waiting for hand...", обнуляє показник упевненості та фіксує у системному журналі логів статус готовності ui_log_message = "System ready." [27, 29].

Сценарій 2. Керування навантаженням у статичному режимі. Оператор поміщає кисть руки у робоче поле зору оптичного пристрою. Програма автоматично активує режим STATIC (назва режиму підсвічується яскравим зеленим кольором) [23]. Користувач фіксує певну позу (наприклад, піднімає два пальці). Система відображає назву розпізнаного жесту «Two Fingers (Ch B)» та виводить динамічний рівень упевненості моделі у відсотках (наприклад, Confidence: 94.7%). Після утримання пози протягом 20 кадрів на екрані з'являється лог підтвердження: Sent: B (Two Fingers (Ch B)), а на платі DFRobot ATmega32U4 асинхронно інвертується стан реле (пін 4) [31, 33].

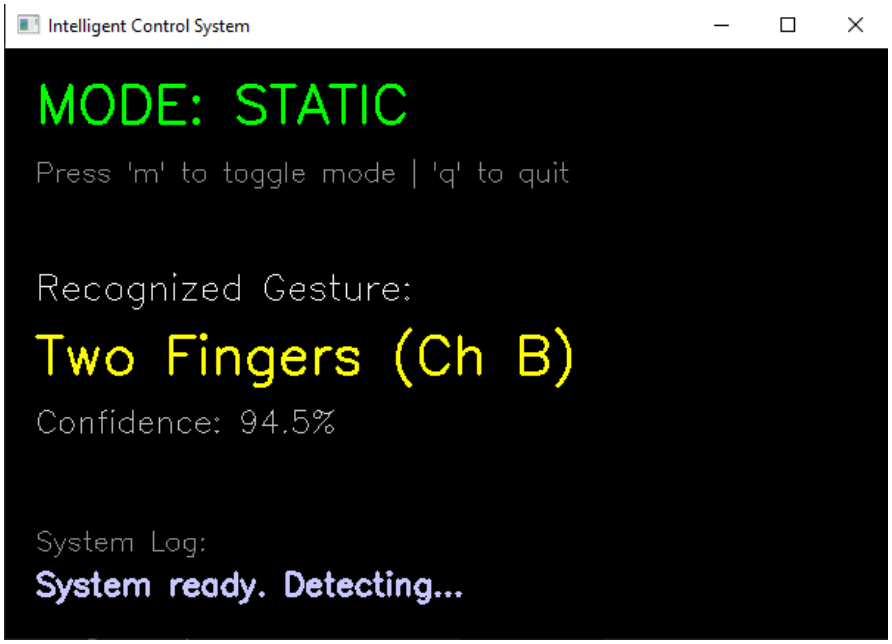


Рисунок 3.12 – Графічний інтерфейс системи в режимі статичного аналізу

Сценарій 3. Асинхронне перемикання операційних режимів. Для переходу до аналізу часових послідовностей оператор натискає клавішу m на клавіатурі ЕОМ. Головне ядро перехоплює код символу, миттєво очищує часовий буфер frame_buffer.clear(), обнуляє лічильники стабілізації та переводить змінну current_mode у стан DYNAMIC [31]. Інтерфейс миттєво реагує на подію, змінюючи колір індикатора режиму на оранжевий, та виводить у лог системне повідомлення Switched to DYNAMIC mode [29].

Сценарій 4. Керування навантаженням у динамічному режимі. У режимі DYNAMIC користувач бачить статус Accumulating buffer..., доки черга FIFO не накопичить перші 20 кадрів. Після заповнення буфера на екрані з'являється індикатор готовності Ready for swipe. Оператор здійснює швидкий змах рукою праворуч. Нейромережа LSTM розпізнає часову траєкторію, дашборд виводить жовтим кольором назву жесту «Swipe Right (ALL ON)», лог фіксує відправку байта Sent: R (Swipe Right (ALL ON)), а виконавчий блок Arduino одночасно вмикає світлодіод, реле та повертає сервопривод SG90 на 90° [21, 31]. Після цього інтерфейс блокується на 50 кадрів, візуалізуючи стан часового кулдауну [27].

Сценарій 5. Штатне переривання та закриття системи. Для завершення сесії взаємодії оператор натискає гарячу клавішу q. Програмний комплекс виходить із нескінченного обчислювального циклу while, виконує безпечне апаратне закриття COM-порту через метод arduino.close(), руйнує графічні вікна бібліотеки OpenCV (cv2.destroyAllWindows()) та звільняє виділені ресурси оперативної пам'яті ЕОМ [31, 35].

3.3.2 UML-діаграма станів інтерфейсу користувача

Для детального опису внутрішньої логіки функціонування графічної оболонки та мапування переходів між інформаційними станами під впливом зовнішніх подій (жестів оператора або сигналів таймерів) було спроєктовано поведінкову UML-діаграму станів (UML State Machine Diagram) [29].

Діаграма відображає життєвий цикл інтерфейсу, що складається з таких суперстанів та станів: Initialization (завантаження вагових коефіцієнтів та ініціалізація Serial-зв'язку), WaitingForHand (інертне сканування простору), StaticMode (контур обробки багатошарового перцептрона), DebounceFilter (накопичення часового порогу стабільності пози), DynamicMode (контур накопичення черги FIFO та LSTM-аналізу), CooldownState (заморожування контуру після свайпу) та Termination [23, 27, 29].

Із діаграмою можна ознайомитись у Додатку Д.

3.4 Тестування та аналіз результатів

Завершальним етапом розробки інтелектуальної системи безконтактного керування є комплексне тестування її компонентів. Метою тестування є верифікація алгоритмів обробки просторових даних, оцінка точності моделей машинного навчання та перевірка логіки формування керуючих команд [24].

Враховуючи архітектурний розподіл системи на програмне ядро (Python) та мікроконтролерний рівень (C++), тестування проводилося методами модульного та інтеграційного аналізу [34]. Оцінка статистичних метрик моделей виконувалася локально на базі зібраних датасетів, тоді як перевірка керуючих сигналів здійснювалася шляхом моніторингу логів та буферів послідовного порту [31].

3.4.1 Тестові сценарії та набори даних

Ядром інтелектуальної системи є натреновані моделі класифікації. Для їх навчання та валідації було попередньо згенеровано два структуровані набори даних (датасети) за допомогою модулів collect_data.py та collect_dynamic_data.py [32].

Формування датасетів відбувалося в реальних умовах експлуатації оптичного сенсора Leap Motion із фіксацією 63-вимірних просторових векторів ознак. Характеристики зібраних наборів даних наведено в таблиці 3.4.

Таблиця 3.4 – Характеристики сформованих наборів просторових даних

Параметр	Статичний датасет (static_gestures.csv)	Динамічний датасет (dynamic_gestures.csv)
Обсяг даних	3000 векторів (кадрів)	400 матриць (часових послідовностей)
Розподіл за класами	3 класи по 1000 семплів	2 класи по 200 семплів
Розмірність ознаки	Вектор: 1×63 (числа типу Float)	Тензор: 20×63 (20 кадрів по 63 координати)
Методологія розбиття	80% тренувальна / 20\% тестова вибірка	80\% тренувальна / 20\% тестова вибірка
Базова модель	Багатошаровий перцептрон (MLPClassifier)	Рекурентна нейромережа (LSTM)

Процедура тестування була поділена на два базові сценарії [24, 27]:

– програмно-алгоритмічний сценарій: локальне тестування метрик точності моделей на тестовій (небаченій під час навчання) вибірці (20% від загального обсягу). Проводився аналіз Accuracy, F1-score та матриці помилок (Confusion Matrix).

– інтеграційно-апаратний сценарій: перевірка спрацювання логіки скрипта main.py шляхом емуляції рухів перед сенсором та контролю генерації байт-кодів (b'A', b'R' тощо) у системний лог дашборду. Завдяки цьому сценарію підтверджується працездатність системи навіть за відсутності фізичного доступу до університетського лабораторного обладнання (зчитування байтів із буфера Serial-інтерфейсу) [31].

3.4.2 Тестування функціоналу розпізнавання жестів та управління пристроєм

Для оцінки якості розпізнавання жестів розробленими моделями було використано стандартні метрики машинного навчання з бібліотеки scikit-learn [22].

Результати тестування статичного класифікатора (MLP). Після запуску модуля train_static.py нейромережа пройшла 500 ітерацій навчання. Результати валідації на тестовій вибірці (600 семплів) показали високий рівень узагальнення даних. Завдяки використанню точних координат кісток (на відміну від піксельних

RGB-зображень), система не зазнає впливу освітлення, що дозволяє досягти стабільної точності (Accuracy).

```
3. Оцінка результатів на тестовій вибірці:
Accuracy (Загальна точність): 99.83%

Детальний звіт (macro-F1 та інші метрики):
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       217
         1.0       1.00      0.99      1.00       197
         2.0       1.00      1.00      1.00       186

    accuracy                           1.00       600
   macro avg       1.00      1.00      1.00       600
weighted avg       1.00      1.00      1.00       600

Матриця помилок (Confusion Matrix):
[[217   0   0]
 [  1 196   0]
 [  0   0 186]]
```

Рисунок 3.13 – Метрики точності статичної моделі розпізнавання поз

Результати тестування динамічного класифікатора (LSTM). Для навчання часової мережі використовувався скрипт train_dynamic.py. Валідація на тестовій вибірці (80 послідовностей) продемонструвала здатність LSTM-шару ефективно виявляти просторові залежності в межах вікна з 20 кадрів [25]. Процедура стандартизації ознак (StandardScaler) перед подачею у мережу мінімізувала вплив індивідуальних анатомічних розмірів руки користувача.

```
5. Оцінка результатів на тестовій вибірці:
3/3 [==============================] - 0s 5ms/step

Детальний звіт:
              precision    recall  f1-score   support

         0.0       0.89      0.77      0.83        44
         1.0       0.76      0.89      0.82        36

    accuracy                           0.82        80
   macro avg       0.83      0.83      0.82        80
weighted avg       0.83      0.82      0.83        80

Матриця помилок (Confusion Matrix):
[[34 10]
 [ 4 32]]
```

Рисунок 3.14 – Метрики точності динамічної моделі розпізнавання свайпів

Тестування логіки формування керуючих команд (Serial Communications). Перевірка мосту взаємодії між модулем ШІ та інтерфейсом мікроконтролера виконувалася під час роботи скрипта main.py. Оскільки алгоритм передбачає жорстке фільтрування випадкових рухів, генерація байт-кодів тестувалася у стрес-режимі (швидка зміна жестів перед сенсором). У таблиці 3.5 наведено результати перевірки логіки кінцевого автомата через моніторинг UI-дашборду.

Таблиця 3.5 – Результати тестування програмного ядра та системи фільтрації

Тестова дія користувача	Очікувана реакція системи	Результат (Лог UI / Serial)	Статус тесту
Утримання жесту «One Finger» менше 20 кадрів	Ігнорування, спрацювання DEBOUNCE_FRAMES	Лог порожній, ui_confidence коливається	Успішно
Утримання жесту «Two Fingers» понад 20 кадрів	Генерація байту b'B'	Sent: b'B' (Two Fingers (Ch B))	Успішно
Виконання швидкого свайпу праворуч у режимі DYNAMIC	Розпізнавання нейромережею, байт b'R', блокування	Sent: b'R' (Swipe Right (ALL ON)), активація dynamic_cooldown = 50	Успішно
Спроба зробити новий свайп під час дії cooldown	Ігнорування (захист від подвійного спрацювання)	Стан буфера заморожений	Успішно

Аналіз логів підтверджує, що програмне ядро коректно обробляє ймовірності класифікаторів та генерує детерміновані сигнали для апаратного рівня (Arduino ATmega32U4), які повністю відповідають розробленій матриці мапування [31, 33].

Із результатами роботи системи, можна ознайомитись на рисунках 3.15 та 3.16.

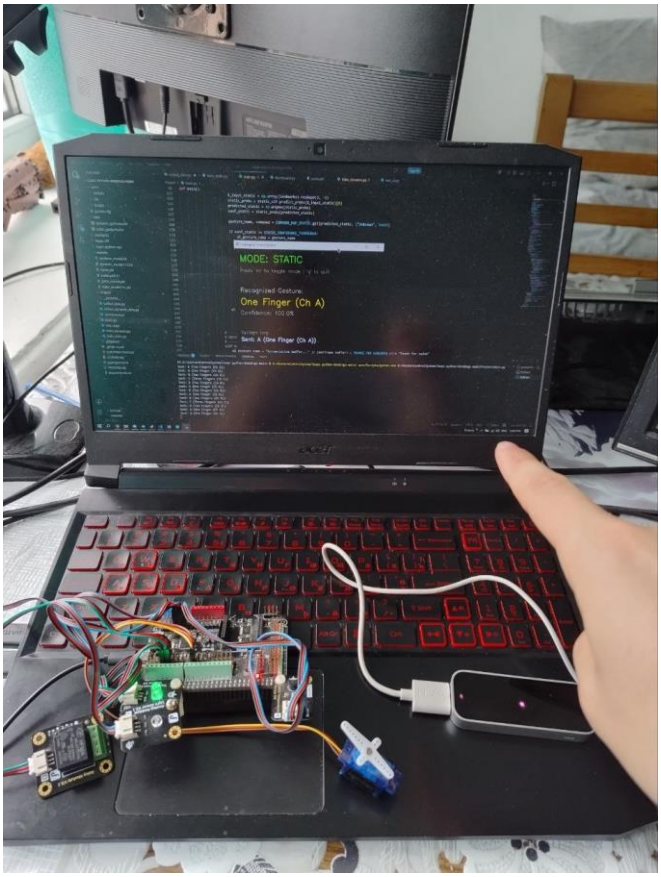


Рисунок 3.15 – Розпізнавання жесту «One Finger» та активація світлодіоду

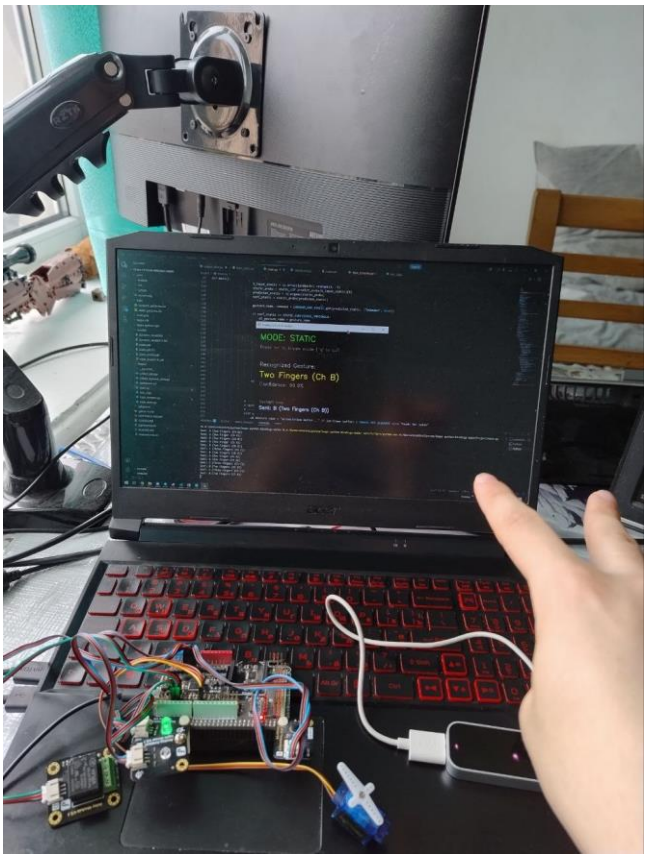


Рисунок 3.16 – Розпізнавання жесту «Two Fingers» та активація реле

3.4.3 Аналіз помилок і обмежень запропонованого підходу

Незважаючи на високі показники точності ізольованих моделей, під час інтеграційного тестування системи в реальному часі було виявлено низку алгоритмічних та апаратних обмежень. Більшість із них є типовими для оптичних систем координатного відстеження та потребували програмної компенсації на етапі розробки [12, 27].

Основні виявлені проблеми, їхній вплив на систему та імплементовані інженерні рішення систематизовано у таблиці 3.6.

Таблиця 3.6 – Аналіз обмежень та інженерних рішень інтелектуального інтерфейсу

Виявлена проблема (Обмеження)	Фізична природа помилки	Впроваджене програмне рішення у код
«Брязкіт» (Chattering) класів на межі кадрів	Нестабільність розпізнавання під час переходу руки з однієї пози в іншу (ШІ видає випадкові класи на частки секунди)	Впровадження програмного лічильника стабільності DEBOUNCE_FRAMES = 20. Команда надсилається лише після стабільного утримання пози
Помилкові повторні спрацювання свайпів	Повернення руки у вихідну позицію після виконання жесту сприймалося нейромережею як новий рух	Розробка механізму «заморожування» контуру DYNAMIC_COOLDOWN_FRAMES = 50. Блокування аналізу на ~0.5 сек після кожної дії
Втрата координат при самоперекритті пальців (Occlusion)	Інфрачервона стереокамера Leap Motion фізично не бачить пальці, якщо вони сховані за долонею (наприклад, стиснуті в кулак)	Впровадження суворої валідації вхідного зрізу if landmarks and len(landmarks) == 63. При втраті хоча б однієї точки кадр відкидається (Drop Frame), запобігаючи краху моделей

Продовження таблиці 3.6

Виявлена проблема (Обмеження)	Фізична природа помилки	Впроваджене програмне рішення у код
Залежність від масштабів руки	Анатомічна різниця в довжині пальців користувачів зміщує абсолютні координати	Впровадження StandardScaler (scaler.transform) перед подачею вектора в LSTM, що нормалізує пропорції
Блокування інтерфейсу при зависанні сенсора	Очікування синхронних даних від сенсора зупиняло графічний дашборд	Використання неблокуючого оператора cv2.waitKey(1) та ізоляція обробника подій SystemListener в окремий асинхронний потік через LeapC API

Результати тестування доводять, що спроєктоване алгоритмічне ядро та імплементовані фільтри часового згладжування частково нівелюють апаратні обмеження оптичного сенсора. Система забезпечує стабільне й перетворення просторових жестів на детерміновані мікроконтролерні інструкції з показниками надійності, достатніми для впровадження у практичні завдання безконтактного керування.

ВИСНОВКИ

У кваліфікаційній роботі вирішено науково-практичне завдання розробки надійної програмно-апаратної системи безконтактного керування обладнанням на основі розпізнавання просторово-часових жестів. За результатами роботи сформульовано такі висновки:

1) Проведено аналіз сучасних систем комп'ютерного зору та безконтактної взаємодії, що дозволило обрати апаратну базу, та сформулювати постановку задачі. Як апаратну базу обрано оптичний сенсор Leap Motion для високоточного зчитування координат та мікроконтролерну плату DFRobot (ATmega32U4) для миттєвого відпрацювання апаратних команд.

2) Розроблено алгоритмічний конвеєр збору та попередньої обробки даних, який формує 63-вимірний вектор ознак. Впровадження методів просторової центризації та Z-стандартизації дозволило зробити систему інваріантною до анатомічних масштабів руки користувача.

3) Сформовано репрезентативні набори даних (3000 статичних та 400 динамічних семплів) та реалізовано класифікатор на базі багатошарового перцептрона. Модель з високою точністю розпізнає статичні пози для ізольованого керування каналами навантаження.

4) Розроблено модуль розпізнавання динамічних траєкторій з використанням рекурентної глибокої нейромережі , яка успішно аналізує часові матриці фіксованої довжини у 20 кадрів для ініціалізації глобальних команд.

5) Створено гібридне програмне ядро з імплементацією алгоритмів часової фільтрації: дебаунсингу та кулдауну. Це усунуло ефект «брязкання» класів та хибні подвійні спрацювання.

6) Розроблено зручний асинхронний графічний інтерфейс користувача , який функціонує за принципом інформаційного мінімалізму та забезпечує візуальний контроль поточних режимів, упевненості нейромереж і системних логів.

7) Проведено комплексне тестування розробленого рішення, яке підтвердило його стабільність, відповідність функціональним вимогам і безпомилкову

взаємодію інтелектуальних моделей із мікроконтролерним рівнем у реальному часі.

Практична цінність одержаних результатів полягає у створенні готового до експлуатації програмно-апаратного комплексу. Сфери його потенційного використання охоплюють медичні установи (стерильні зони), промислові підприємства з агресивними середовищами та комплекси «розумного дому», де традиційний тактильний контакт із перемикачами є небажаним або небезпечним.

Рекомендації для подальших досліджень включають міграцію обчислювального ядра системи на автономні портативні мікрокомп'ютери, наприклад, Raspberry Pi 5, для реалізації парадигми граничних обчислень , а також використання новітніх трансформерних архітектур для розширення бібліотеки складними дворучними маніпуляціями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Білоусов І. Г. Метод розпізнавання жестів для інтерактивного керування комп’ютером з урахуванням контекстної адаптації: кваліфікаційна робота магістра: 123. Харків: ХНУРЕ, 2025. 78 с.

2. Комар В. А. Моделі інтеграції голосових команд та жестів для управління "розумним" будинком: кваліфікаційна робота магістра: 123. Тернопіль: ЗУНУ, 2024. 82 с.

3. 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 «Штучний інтелект» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 57 с.

4. Тихомиров В. І. Система відео розпізнавання жестів мови глухонімих з використанням нейронних мереж у реальному часі: кваліфікаційна робота магістра: 123. Харків: ХНУРЕ, 2023. 65 с.

5. Щур А. С. Підсистема перекладу жестової мови в кіберфізичній системі: магістерська дисертація: 126. Київ: КПІ ім. Ігоря Сікорського, 2024. 92 с..

6. Alabdullah B. I., Ansar H., Mudawi N. A., et al. Smart Home Automation-Based Hand Gesture Recognition Using Feature Fusion and Recurrent Neural Network. *Sensors*. 2023. Vol. 23(17). 7523.

7. Gillian N. E., Paradiso J. A. The Gesture Recognition Toolkit. *Journal of Machine Learning Research*. 2014. Vol. 15. P. 3483–3487.

8. Hobbs T., Ali A. Smart Home Control Using Real-Time Hand Gesture Recognition and Artificial Intelligence on Raspberry Pi 5. *Electronics*. 2025. Vol. 14(20). 3976.

9. Linardakis M., Varlamis I., Papadopoulos G. T. Survey on Hand Gesture Recognition from Visual Input. arXiv preprint arXiv:2501.11992. 2025.

10. Lu C., Zhang H., Pei Y., et al. Online Hand Gesture Detection and Recognition for UAV Motion Planning. *Machines*. 2023. Vol. 11(2). 210.

11. Moysiadis V., Katikaridis D., Benos L., et al. An Integrated Real-Time Hand Gesture Recognition Framework for Human–Robot Interaction in Agriculture. *Applied Sciences*. 2022. Vol. 12(16). 8160.

12. Oudah M., Al-Naji A., Chahl J. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. *Journal of Imaging*. 2020. Vol. 6(8). 73.

13. Sarma D., Bhuyan M. K. Methods, Databases and Recent Advancement of Vision-Based Hand Gesture Recognition for HCI Systems: A Review. SN Computer Science. 2021. Vol. 2. 436.

14. Sarma D., Bhuyan M. K. Methods, Databases and Recent Advancement of Vision-Based Hand Gesture Recognition for HCI Systems: A Review. *SN Computer Science*. 2021. Vol. 2. 436.

15. Sen A., Mishra T. K., Dash R. Deep learning based Hand gesture recognition system and design of a Human-Machine Interface. International Journal of Intelligent Systems and Applications. 2022. Vol. 14(3). P. 12–25.

16. Teslyuk V., Chornenkyi V., Tsapiv V., Kazymyra I. Investigating Hand Gesture Recognition Models: 2D CNNs vs. Visual Transformers. *Lviv Polytechnic National University*. 2024.

17. Touchless Sensing and Gesture Recognition Market by Technology, Product, Industry and Geography - Global Forecast to 2030. *MarketsandMarkets*. URL: https://www.marketsandmarkets.com/Market-Reports/touchless-sensing-gesturing-market-369.html (дата звернення: 13.03.2026).

18. Zhang F., Bazarevsky V., Vakunov A., et al. MediaPipe Hands: On-device Real-time Hand Tracking. arXiv preprint arXiv:2006.10214. 2020.

19. 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.

20. ДСТУ 3008:2015. Національний стандарт України. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2016-07-01]. Київ: ДП «УкрНДНЦ», 2016. 25 с.

21. ДСТУ 8302:2015. Національний стандарт України. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2016-07-01]. Київ: КР «УкрНДНЦ», 2016. 17 с.

22. Sahoo A. K., Udgata S. K., Saxena S. L. Hand gesture recognition using machine learning algorithms. ResearchGate. 2020. Vol. 12(4). P. 153–161.

23. Al-Shabi A., et al. A Novel Hybrid Deep Learning Architecture for Dynamic Hand Gesture Recognition. ResearchGate. 2024. Vol. 16(2). 1042.

24. Duprey M., et al. Hand Gesture Recognition Using Skeletal Data. HAL Open Science. 2018. P. 45–52.

25. Tseng Y.-H., Hsu C.-W., Leu Y.-G. Real-Time Recognition of Dual-Arm Motion Using Joint Direction Vectors and Temporal Deep Learning. *Engineering Proceedings*. 2024. Vol. 120(1). 75.

26. Raschka S., Mirjalili V. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. 3rd ed. Birmingham: Packt Publishing, 2019. 541 p.

27. Chollet F. Deep Learning with Python. 2nd ed. Shelter Island, NY: Manning Publications, 2021. 504 p.

28. Monk S. Programming Arduino: Getting Started with Sketches. 2nd ed. New York: McGraw-Hill Education, 2016. 192 p.

29. Cooper A., Reimann R., Cronin D., Noessel C. About Face: The Essentials of Interaction Design. 4th ed. Indianapolis, IN: John Wiley & Sons, 2014. 720 p.

30. Python Software Foundation. Python standard library: collections — Container datatypes (deque). Python 3.12 Documentation. 2026. URL: https://docs.python.org/3/library/collections.html#collections.deque (дата звернення: 18.05.2026).

31. pySerial Team. pySerial’s documentation: The API — serial.Serial(). Read the Docs. 2024. URL: https://pyserial.readthedocs.io/en/latest/pyserial_api.html (дата звернення: 18.05.2026).

32. Ultraleap. Leap Motion SDK: LeapC Guide. Ultraleap Developer Docs. 2025. URL: https://docs.ultraleap.com/api-reference/tracking-api/leapc-guide.html (дата звернення: 18.05.2026).

33. Arduino LLC. Arduino Language Reference: Communication — Serial. Arduino Docs. 2026. URL: https://www.arduino.cc/reference/en/language/functions/communication/serial/ (дата звернення: 18.05.2026).

34. Ultraleap. LeapC Python Bindings: Official Python wrapper for the Leap Motion Tracking Software. GitHub repository. 2024. URL: https://github.com/ultraleap/leapc-python-bindings (дата звернення: 18.05.2026).

35. Leus V. “Concept of an intelligent hardware control system based on gesture recognition,” in Proc. 1st Int. Sci. Pract. Conf. ICSPM 2026: Intelligent Computing Systems and Project Management, Ternopil, Ukraine, May 21–22, 2026, 2026.

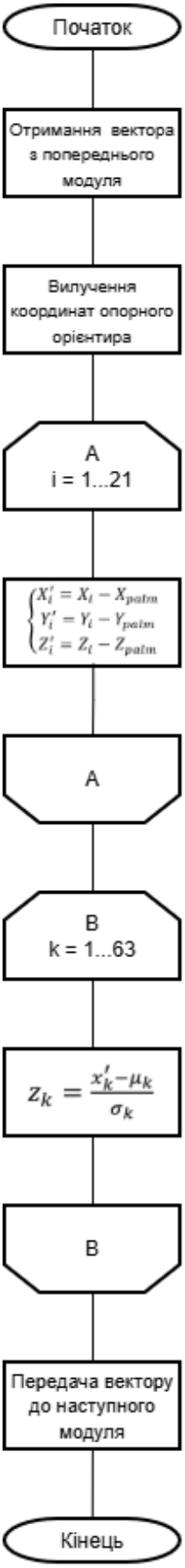
36. Леус В. Система керування жестами на основі контролера Leap Motion та мікроконтролера Arduino Leonardo. *Збірник тез доповідей студентської науково-практичної конференції, «Інтелектуальні інформаційні технології в прикладних дослідженнях* (IITAR-2025), 27-29 травня 2025 р., Тернопіль, с.295-298

Додаток А

Блок-схема алгоритму захоплення та первинної валідації просторових даних

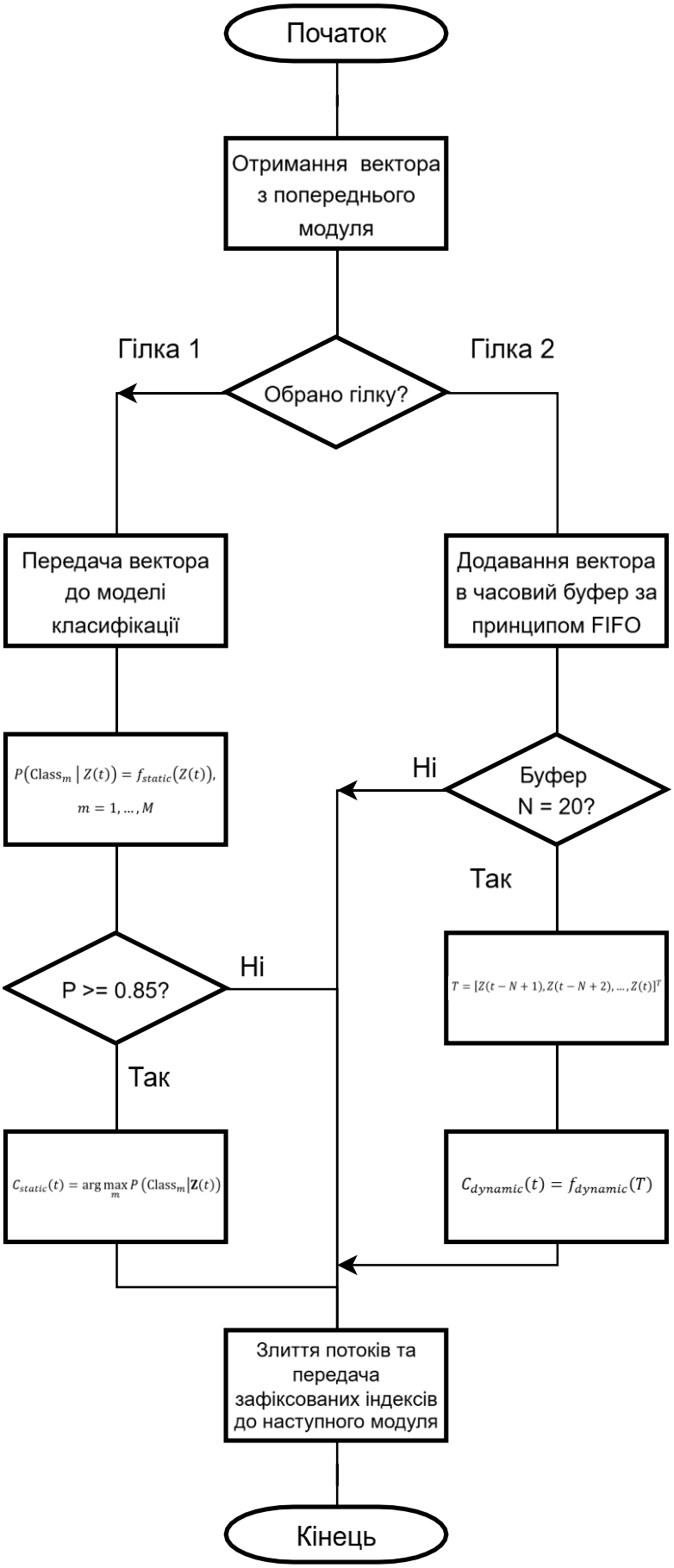
Додаток Б

Блок-схема алгоритму трансформації та стандартизації ознакового простору



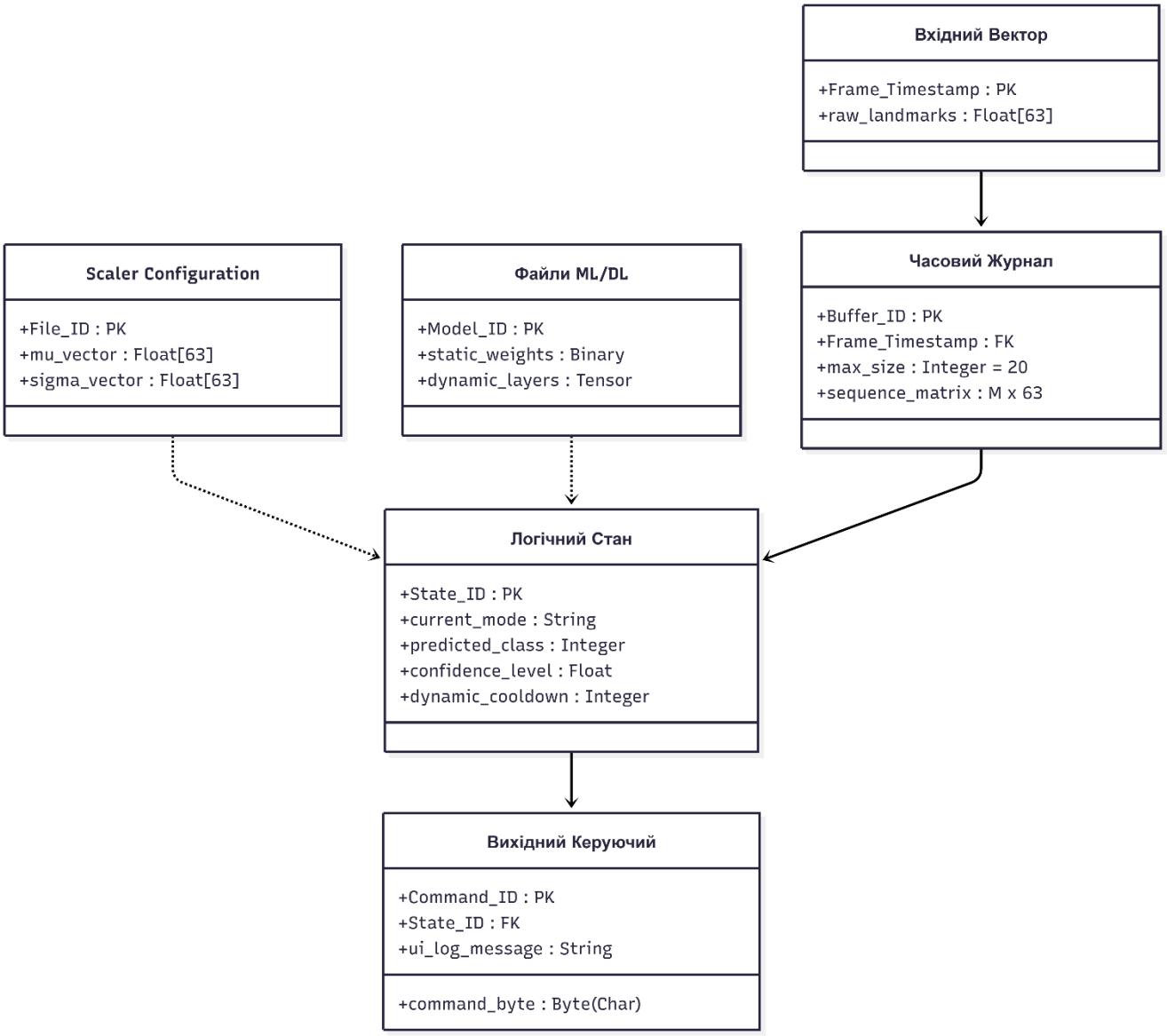
Додаток В

Блок-схема алгоритму гібридної статико-динамічної класифікації жестів



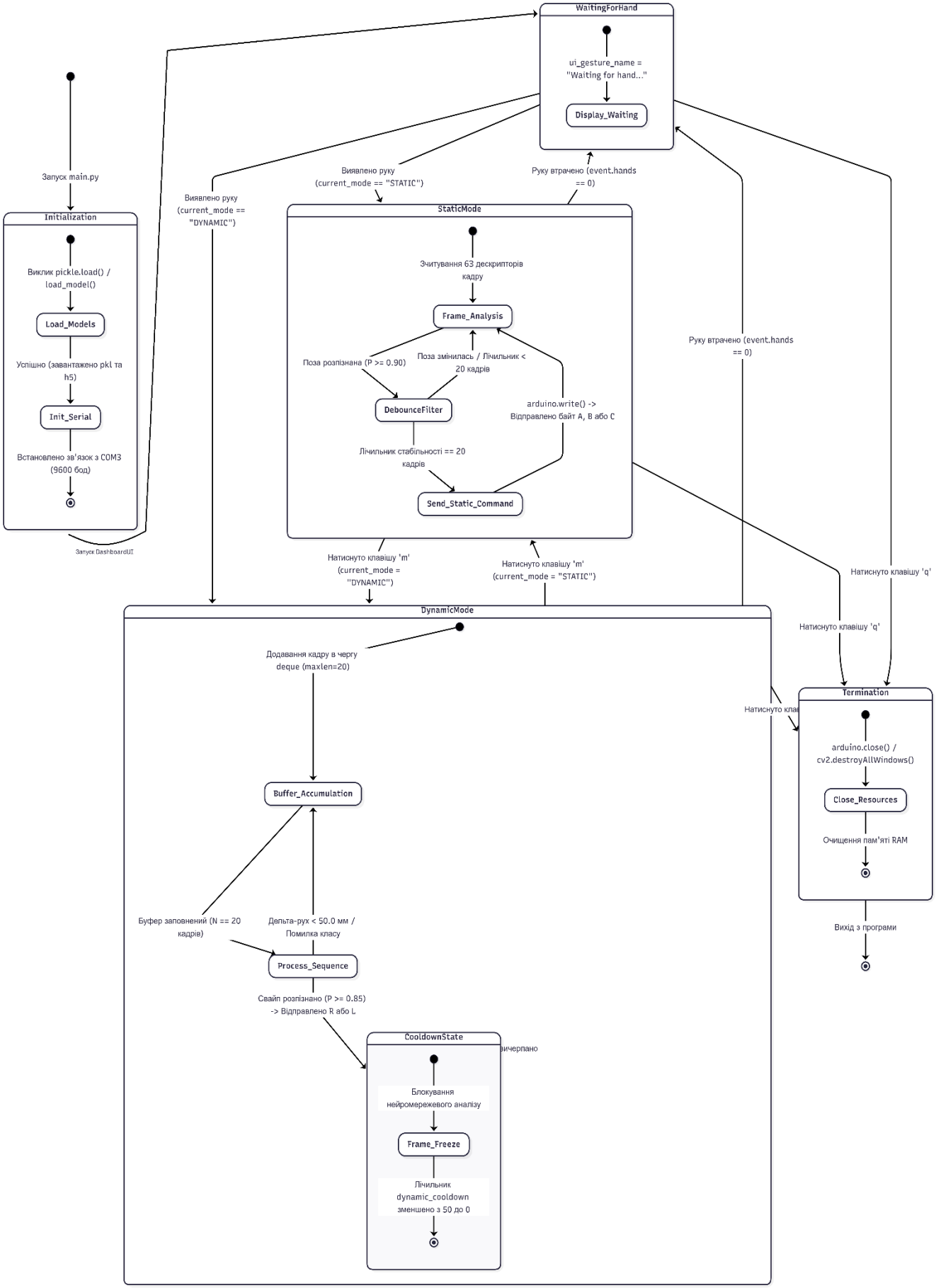
Додаток Г

Діаграма структури даних



Додаток Д

UML-діаграма станів людино-машинного інтерфейсу системи



Додаток Е

Копія публікації результатів дослідження

Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Матеріали

І Міжнародної науково-практичної конференції студентів і молодих вчених

«ICSPM 2026: Intelligent Computing Systems and Project Management / Інтелектуальні обчислювальні системи та управління проєктами»

21–22 травня 2026 р.

м. Тернопіль - 2026

УДК 004.8:005.8](063)

Присвячено 60-річчю Західноукраїнського національного університету

ICSPM 2026: Інтелектуальні обчислювальні системи та управління проєктами : збірник тез доповідей І Міжнародної науково-практичної конференції студентів і молодих вчених (Тернопіль, 21–22 травня 2026 року). – Тернопіль : ЗУНУ, 2026. – 190 с.

До збірника увійшли тези доповідей учасників І Міжнародної науково-практичної конференції студентів і молодих вчених «ICSPM 2026: Intelligent Computing Systems and Project Management / Інтелектуальні обчислювальні системи та управління проєктами», що відбулася на базі кафедри інформаційно-обчислювальних систем і управління факультету комп'ютерних інформаційних технологій Західноукраїнського національного університету.

Матеріали відображають результати досліджень за такими напрямами:

1. Інтелектуальні обчислення та програмні системи.
2. Проєктно-орієнтоване управління та процеси прийняття рішень.
3. Штучний інтелект, аналітика даних і кібернетика.
4. Прикладні технології та інформаційно-комунікаційні системи.
5. Сучасні інформаційні технології в економіці, праві та освіті.

Рекомендовано до друку на засіданні кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету (протокол №12 від 26.05.2026 р.).

За зміст наукових праць, точність наведених цитувань, перекладу та достовірність фактологічних матеріалів відповідальність несуть автори публікацій. У збірнику збережено авторську стилістику, пунктуацію та орфографію.

ЗМІСТ

UDC 004.93

CONCEPT OF AN INTELLIGENT HARDWARE CONTROL SYSTEM BASED ON GESTURE RECOGNITION

Vitalii Leus, student,
v.leus@st.wunu.edu.ua

Scientific supervisor: Oleksandr Osolinskyi,
Associate Professor, Department of Information
and Computing Systems and Control,
West Ukrainian National University,
oso@wunu.edu.ua

At the current stage of information technology development, a global transformation of human-computer interaction (HCI) interfaces is taking place, with the creation of natural, intuitive communication methods becoming a priority. Global trends indicate the gradual replacement of traditional manipulators with intelligent contactless interfaces based on computer vision. In scientific literature, hand gesture recognition is considered a fundamental task of human-machine interaction, enabling the control of complex equipment without physical contact with input devices [2, 5].

Despite significant progress and the deployment of efficient frameworks for hand skeleton tracking, such as MediaPipe Hands, unresolved problems and challenges still remain. The primary difficulties arise when systems operate in real-world environments: dynamic lighting, complex backgrounds, partial occlusion by other objects, as well as the limited computational resources of mobile platforms [1, 6]. Consequently, developing robust and energy-efficient algorithms that can reliably function within industrial environments or the "Smart Home" concept is of high relevance, where touchless control is essential for improving hygienic safety and ergonomics of work processes [3].

To select the technological foundation of the system, a formalized efficiency assessment of existing hardware solutions was implemented using the weighted sum method. The integral indicator $Score$ is calculated by the formula, where the parameters are evaluated on a 10-point scale:

$$Score = C_1 \times W_1 + C_2 \times W_2 + C_3 \times W_3 + C_4 \times W_4 , \quad (1)$$

where W_1 — the level of overcoming technological challenges (the difficulty of mitigating obstacles);
W_2 — the availability and prevalence of tools;
W_3 — the quality of technical specifications (accuracy, speed);
W_4 — the universality of application;
C_n — the weighting coefficient.

The weighting coefficients are determined according to the system development priorities: $C_1 = 0{,}3$ (robustness to noise), $C_2 = 0{,}2$ (availability), $C_3 = 0{,}4$ (performance) and $C_4 = 0.1$ (universality). Part of the comparison results is presented in Table 1.

Table 1 – Comparison of primary hand gesture recognition means

Data Source	**Challenges (W1)**	**Tools (W2)**	**Specifications (W3)**	**Application (W4)**	**Evaluation (Score)**
Webcameras, Kinect	Field of view (7)	Available (10)	High accuracy (8)	HCI, household (9)	8.2
Leap Motion	Lighting (6)	Specialized (8)	Joint tracking (10)	Sign language (8)	8.2
Depth sensors	Range (6)	Medium (7)	3D information (9)	Gaming (8)	7.6
Smart gloves	Comfort (4)	Expensive (6)	Hand accuracy (9)	Industry (8)	7.2
Electromyography (EMG)	Placement (4)	Special (5)	Muscle activity (7)	Prosthetics (6)	5.6

Based on the analysis results, the highest integral evaluation (8.2 points) was achieved by computer vision-based systems utilizing standard webcameras and the Leap Motion controller. The selected approach is based on transitioning from simple visual image analysis and skin color segmentation to multi-level data processing with the construction of skeletal models. This allows for separating the pose capture process from the command interpretation logic. The conceptual diagram of the proposed solution is shown in Figure 1.

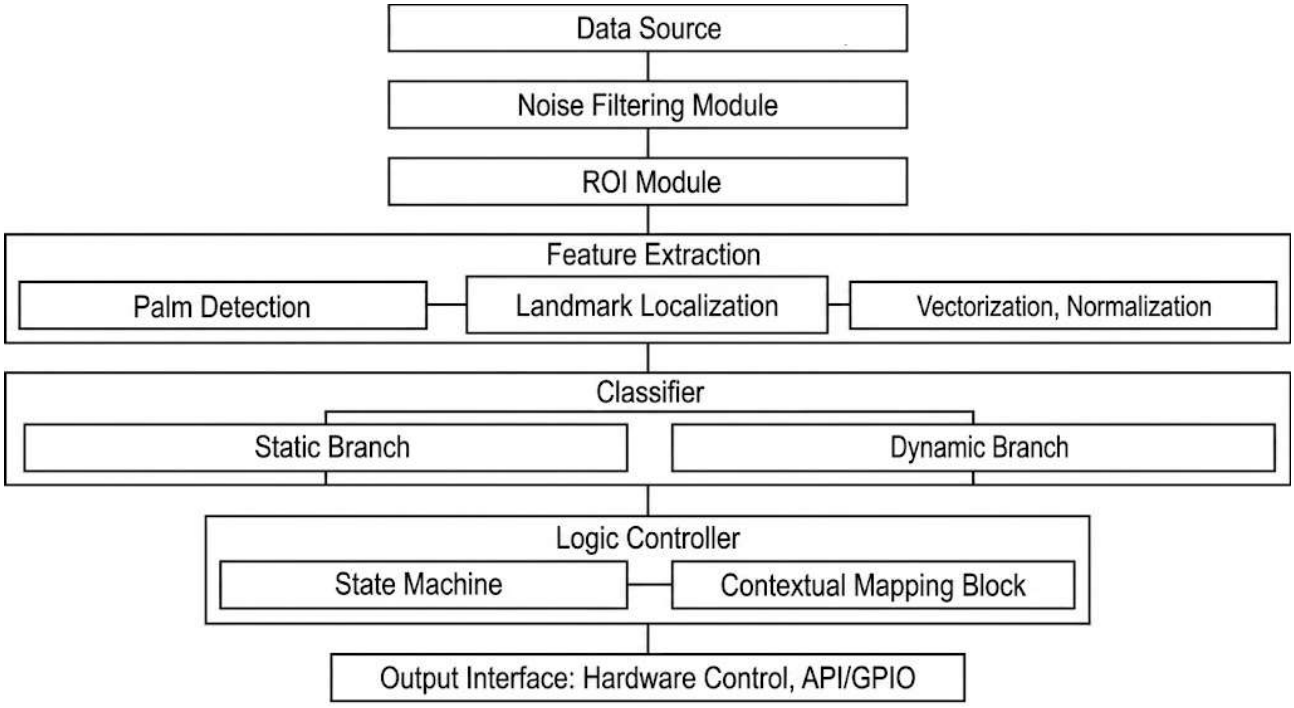


Figure 1 – System architecture concept

According to the developed concept, the first stage of video stream processing is the noise filtering and Region of Interest extraction module, which allows for automatic focusing on dynamic changes in the frame and ignoring background interference. The next level is the structural abstraction and feature extraction block, which includes a palm detector and a landmark localizer. Representing a gesture as a set of spatial vectors followed by their normalization ensures the system's invariance to the hand's distance from the camera and its orientation [5].

The central intelligent element of the system is the classifier, whose logic is divided into two branches. The static branch is responsible for recognizing the pose in the current frame using lightweight machine learning models, for example a one-dimensional convolutional neural network operating on the feature vector. Concurrently, the dynamic branch analyzes the trajectory of movements through a temporal sequence buffer, applying recurrent neural networks, which allows for differentiating complex gestures that share similar initial phases but have distinct completion trajectories [3, 4].

To transform the classification results into control signals, a logical controller is integrated into the architecture. It consists of a state machine that filters out false positives using confidence thresholds and time delay (debounce) mechanisms, and a contextual mapping block. Contextual adaptation allows for changing the meaning of the same gesture depending on the active operation mode of the equipment, making the interface more ergonomic. Final commands are transmitted to the output interface to control the target equipment via an API or input/output ports [6].

Experimental verification of the architecture involves conducting tests for both static and dynamic branches on datasets collected under various lighting conditions and partial occlusion. To evaluate the quality of the algorithms, Accuracy and macro-F1 metrics can be utilized, with latency and FPS metrics to confirm the feasibility of stable real-time operation.

The developed architecture provides a qualitatively new level of human-machine interaction reliability due to the synergy of spatial pose modeling methods and adaptive decision-making logic. This establishes a solid foundation for implementing contactless intelligent interfaces in home automation systems and industrial cyber-physical complexes.

References

1. Alabdullah, B. I., Ansar, H., Mudawi, N. A., et al. (2023). Smart home automation-based hand gesture recognition using feature fusion and recurrent neural network. Sensors, 23(17), 7523. https://www.mdpi.com/1424-8220/23/17/7523

2. Linardakis, M., Varlamis, I., & Papadopoulos, G. T. (2025). Survey on hand gesture recognition from visual input. arXiv preprint arXiv:2501.11992. https://arxiv.org/abs/2501.11992

3. Oudah, M., Al-Naji, A., & Chahl, J. (2020). Hand gesture recognition based on computer vision: A review of techniques. Journal of Imaging, 6(8), 73. https://www.mdpi.com/2313-433X/6/8/73

4. Research on mobile machine learning platforms for human gesture recognition in human-machine interaction systems. (2025). Technology Audit and Production Reserves, 2(2). https://journals.uran.ua/tarp/article/view/325423

5. Sarma, D., & Bhuyan, M. K. (2021). Methods, databases and recent advancement of vision-based hand gesture recognition for HCI systems: A review. SN Computer Science, 2, 436. https://www.researchgate.net/publication/354203395_Methods_Databases_and_Recent_Advancement_of_Vision-Based_Hand_Gesture_Recognition_for_HCI_Systems_A_Review

6. Sen, A., Mishra, T. K., & Dash, R. (2022). Deep learning based hand gesture recognition system and design of a human-machine interface. arXiv preprint arXiv:2207.03112. https://arxiv.org/abs/2207.03112