МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,
д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Мультимедійна клавіатура на основі датчика жестів PAJ7620

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія»

Здобувач _____Станіслав ГАНУЛІЧ
підпис

«__» _____ 202__ р.

Керівник ст. викладач кафедри КІ

_____ Борис САЛТОВСЬКИЙ
підпис

«__» _____ 202__ р.

Миколаїв – 2025

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Гануліча Станіслава Олексійовича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Мультимедійна клавіатура на основі датчика жестів PAJ7620

Затверджена наказом по ЧНУ ім. Петра Могили від 30.10.2024 № 297.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є реалізація апаратного та програмного забезпечення мультимедійної клавіатури на основі датчика жестів PAJ7620.

4. Перелік питань, що підлягають розробці:
Проаналізувати поточні технології та продукти для детектування жестів
Розробити апаратну частину мультимедійної клавіатури.
Розробити програмне забезпечення, яке оброблятиме жести та передаватиме відповідні мультимедійні команди на комп’ютер.
Розробити та провести тестування мультимедійної кравіатури.

5. Перелік графічних матеріалів
Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Борис САЛТОВСЬКИЙ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Станіслав ГАНУЛІЧ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Мультимедійна клавіатура на основі датчика жестів PAJ7620

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	30.10.2024	30.10.2024	Виконано
2	Огляд літератури за темою роботи	01.11.2024	19.01.2025	Виконано
3	Складання календарного плану КБР	20.01.2025	25.02.2025	Виконано
4	Аналіз предметної області	26.02.2025	15.03.2025	Виконано
5	Розробка проєктних рішень	16.03.2025	17.04.2025	Виконано
6	Моделювання та конструювання АПЗ	18.04.2025	03.05.2025	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	04.05.2025	15.05.2025	Виконано
8	Відгук керівника КБР	13.06.2025	15.06.2025	Виконано
9	Оформлення КБР та презентації	15.05.2025	09.06.2025	Виконано
10	Попередній захист	22.05.2025	04.06.2025	Виконано
11	Рецензування	13.06.2025	15.06.2025	Виконано
12	Завершення оформлення КБР та презентації	10.06.2025	12.06.2025	Виконано
13	Захист кваліфікаційної бакалаврської роботи	25.06.2025	25.06.2025	Виконано

Керівник роботи _____ Борис САЛТОВСЬКИЙ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Станіслав ГАНУЛІЧ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Мультимедійна клавіатура на основі датчика жестів PAJ7620»
Студент 405 гр.: Ганулiч Станіслав Олексійович
Керівник: старший викладач Салтовський Борис Григорович

Дана кваліфікаційна бакалаврська робота присвячена розробці апаратно-програмної системи мультимедійної клавіатури з безконтактним керуванням на основі датчика жестів PAJ7620 та мікроконтролера Arduino Pro Micro. Об'єктом дослідження є процес взаємодії людини з комп'ютером за допомогою жестових інтерфейсів, зокрема методи безконтактного керування мультимедійними функціями. Предмет – апаратно-програмна реалізація мультимедійної клавіатури на базі сенсора жестів з підтримкою емуляції HID-пристрою.

Метою є створення інтуїтивного та ефективного альтернативного пристрою введення для управління медіаконтентом через природні рухи рук. Практична значимість полягає у впровадженні інноваційного рішення для розважальних систем, стрімінгу та презентаційної діяльності.

У першому розділі кваліфікаційної бакалаврської роботи проведено аналіз сучасних методів безконтактного керування мультимедійними пристроями та огляд сенсорних технологій. Виділено переваги жестових інтерфейсів над традиційними методами введення, зокрема їхню інтуїтивність та мінімізацію фізичного контакту з пристроями.

У другому розділі обґрунтовано вибір апаратного забезпечення – датчика жестів PAJ7620 та мікроконтролера Arduino Pro Micro. Розроблено архітектуру апаратно-програмного комплексу для обробки жестових команд та емуляції HID-пристрою. Запропоновано алгоритм розпізнавання жестів та їх перетворення у мультимедійні команди. Побудовано схему взаємодії компонентів системи та визначено критерії оцінки функціональності.

У третьому розділі реалізовано програмне забезпечення для Arduino, що забезпечує зчитування жестів із сенсора PAJ7620, їхню обробку та передавання відповідних HID-команд комп'ютеру. Описано структуру коду, алгоритми калібрування та логіку прийняття рішень. Проведено тестування, що показали високу точність розпізнавання жестів та зручність користування у різних сценаріях застосування.

Пояснювальна записка містить 64 сторінок (без додатків), 40 рисунків, 4 таблиці, 21 джерел та 2 додатки.

***Ключові слова**: жестовий інтерфейс, PAJ7620, Arduino Pro Micro, мультимедійна клавіатура, безконтактне керування, датчик жестів, HID, USB.*

ABSTRACT

of the Bachelor's Thesis

«Multimedia keyboard based on the RAJ7620 gesture sensor»

Student: Stanislav Hanulich

Supervisor: Senior Lecturer Boris Saltovsky

This bachelor's thesis is devoted to the development of a hardware and software system for a multimedia keyboard with contactless control based on the PAJ7620 gesture sensor and the Arduino Pro Micro microcontroller. The object of research is the process of human-computer interaction using gesture interfaces, in particular, methods of contactless control of multimedia functions. The subject is the hardware and software implementation of a multimedia keyboard based on a gesture sensor with support for HID device emulation.

The goal is to create an intuitive and effective alternative input device for managing media content through natural hand movements. The practical significance lies in the implementation of an innovative solution for entertainment systems, streaming and presentation activities.

The bachelor's thesis first chapter analyzes current methods of contactless control of multimedia devices and reviews sensor technologies. The advantages of gesture interfaces over traditional input methods are highlighted, including their intuitiveness and minimization of physical contact with devices.

The second chapter justifies the choice of hardware - the PAJ7620 gesture sensor and the Arduino Pro Micro microcontroller. The architecture of the hardware and software complex for processing gesture commands and emulating a HID device is developed. An algorithm for recognizing gestures and converting them into multimedia commands is proposed. A scheme of interaction between the system components is constructed and the criteria for evaluating the functionality are determined.

The third chapter implements the software for Arduino, which provides reading gestures from the PAJ7620 sensor, processing them, and transmitting the corresponding HID commands to the computer. The code structure, calibration algorithms, and decision-making logic are described. Experimental studies have been conducted to demonstrate the high accuracy of gesture recognition and ease of use in various application scenarios.

The explanatory note contains 64 pages (without appendices), 40 figures, 4 tables, 21 references and 2 appendices.

Keywords: gesture interface, PAJ7620, Arduino Pro Micro, multimedia keyboard, contactless control, gesture sensor, HID, USB.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІЧ – інфрачервоне

HID – Human Interface Device

I2C – Inter-Integrated Circuit

UART – Universal Asynchronous Receiver-Transmitter

USB – Universal Serial Bus

ВСТУП

У сучасну епоху цифрової трансформації питання зручності, швидкості та інтуїтивності взаємодії людини з комп’ютером стає одним із ключових у розробці нових інтерфейсів. Особливої актуальності набувають безконтактні способи керування, які дозволяють мінімізувати фізичний контакт з пристроями та звільнити користувача від залежності від традиційних методів введення. У цій роботі розглядається проєкт створення мультимедійної клавіатури, яка використовує датчик жестів PAJ7620 і мікроконтролер Arduino Pro Micro, пропонуючи новий підхід до управління медіаконтентом через природні рухи рук.

Мультимедійні пристрої стали невід'ємною частиною як побутового, так і професійного середовища. Водночас потреба в швидкому доступі до функцій управління – таких як регулювання гучності, перемикання треків, пауза, відтворення, заглушення звуку – зростає пропорційно до кількості використовуваних платформ, сервісів і застосунків. Звичні клавіатури та миші не завжди відповідають цим вимогам, особливо в умовах обмеженого простору або при необхідності частих перемикань між задачами. Це відкриває шлях для альтернативних рішень, зокрема жестових інтерфейсів.

Науково-практичне значення роботи полягає у дослідженні та впровадженні інноваційного методу управління мультимедійними функціями комп’ютера на основі безконтактного інтерфейсу, реалізованого за допомогою сенсора жестів PAJ7620. Розроблена система демонструє можливість створення ефективного альтернативного пристрою введення, який поєднує інтуїтивно зрозуміле управління та технічну простоту реалізації. Практичне значення полягає у можливості застосування пристрою в таких галузях, як створення цифрового контенту, презентаційна діяльність.

Об’єктом дослідження є процес взаємодії людини з комп’ютером за допомогою жестових інтерфейсів, зокрема методи і засоби безконтактного керування мультимедійними функціями. Особливу увагу зосереджено на підходах до розробки інтерфейсів користувача, які дозволяють підвищити ергономічність,

ефективність і доступність управління цифровими пристроями, використовуючи сучасні сенсорні технології та мікроконтролери.

Предметом дослідження є апаратно-програмна реалізація мультимедійної клавіатури, що функціонує на базі сенсора жестів PAJ7620 та мікроконтролера Arduino Pro Micro з підтримкою емуляції пристрою HID.

Метою роботи є розробка та реалізація мультимедійної клавіатури на основі датчика жестів PAJ7620 з використанням мікроконтролера Arduino Pro Micro, яка дозволяє здійснювати управління мультимедійними функціями комп’ютера за допомогою жестових команд без фізичного контакту з пристроєм.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналіз сучасних методів безконтактного керування комп’ютером та огляд сенсорів жестів;

– обґрунтувати вибір апаратного забезпечення для реалізації мультимедійної клавіатури, зокрема датчика PAJ7620 та мікроконтролера Arduino Pro Micro;

– розробити архітектуру апаратно-програмного комплексу для обробки жестових команд та емуляції HID-пристрою;

– реалізувати програмне забезпечення, яке забезпечує зчитування жестів, їхню обробку та передавання мультимедійних команд комп’ютеру;

– провести тестування функціональності системи, оцінити точність розпізнавання жестів.

Особливе практичне значення розробленої системи проявляється у домашньому середовищі де воно розкривається через інтеграцію з розважальними системами. Сучасні вітальні перетворюються на мультимедійні центри з численними пристроями, що формують єдину екосистему. Жестове управління дозволяє користувачу легко перемикатися між різними функціями – змінювати гучність музики, перемотувати відео чи переключати джерела контенту – не шукаючи відповідний пульт серед десятка схожих.

Окремої уваги заслуговує практичне застосування мультимедійної клавіатури в індустрії цифрового контенту, зокрема серед стрімерів та творців онлайн-відео. У цій галузі, де одночасне керування декількома додатками та перемикання між сценами трансляції є критично важливим, безконтактний інтерфейс відкриває нові можливості для підвищення якості контенту. Стрімер може плавно змінювати налаштування звуку, перемикати ракурси камер чи активувати спеціальні ефекти за допомогою простих жестів, не перериваючи спілкування з аудиторією та не створюючи відволікаючих звуків клавіатури під час запису. Це не лише підвищує професійність трансляцій, але й дозволяє креаторам зосередитися на творчому процесі та взаємодії з глядачами, що є ключовим фактором успіху в сучасному медіапросторі.

Інтуїтивність жестових команд робить взаємодію з розважальними системами природною навіть для користувачів без технічної підготовки. Змахнув рукою вліво – перейшов до попереднього треку, підняв долоню – призупинив відтворення. Така безпосередність взаємодії повертає нас до природних способів комунікації, роблячи технології дійсно орієнтованими на людину.

Таким чином, розроблена мультимедійна клавіатура на основі датчика жестів стає мостом між технічною функціональністю та людською природністю, забезпечуючи новий рівень комфорту та ефективності як у професійному середовищі презентаційної діяльності, так і в контексті домашніх розважальних систем.

Гануліч С., Салтовський Б. Розробка мультимедійної клавіатури на базі датчика PAJ7620. Ольвійський форум – 2025: стратегії країн Причорноморського регіону в геополітичному просторі : тези доп. XXII Міжнар. наук. конф., Миколаїв, 16–21 черв. 2025 р. Миколаїв : ЧНУ ім. Петра Могили, 2025 (подано до друку).

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ КЕРУВАННЯ МУЛЬТИМЕДІЙНИМИ ФУНКЦІЯМИ ЗА ДОПОМОГОЮ ЖЕСТІВ

1.1 Актуальність розробки

Актуальність розробки мультимедійної клавіатури на основі датчика жестів PAJ7620 обумовлена низкою важливих факторів сучасності. По-перше, в умовах глобальних епідеміологічних викликів зростає попит на технології, які мінімізують фізичний контакт із поверхнями спільного користування. Безконтактні інтерфейси дозволяють значно знизити ризики передачі патогенів через пристрої введення, що особливо критично в публічних просторах, медичних установах та освітніх закладах.

Паралельно з цим спостерігається стрімке зростання індустрії медіарозваг, виробництва цифрового контенту та онлайн-комунікацій. Сучасна людина щоденно взаємодіє з різноманітними мультимедійними пристроями – від смартфонів і планшетів до розумних телевізорів та систем домашніх кінотеатрів. У цьому середовищі виникає потреба в універсальних та інтуїтивно зрозумілих інтерфейсах керування, які б дозволяли швидко перемикатися між різними пристроями та функціями без необхідності використання численних пультів та контролерів.

Технологічний ландшафт також сприяє розвитку безконтактних інтерфейсів. Мініатюризація електронних компонентів, зниження їхньої вартості та підвищення продуктивності створюють сприятливі умови для впровадження просунутих сенсорних систем у масові продукти. Датчики жестів, подібні до PAJ7620, стають достатньо доступними та точними для використання в споживчій електроніці, що відкриває шлях для створення бюджетних рішень з високою функціональністю.

Важливим аспектом актуальності є також тенденція до інклюзивності та доступності технологій. Для людей з обмеженими можливостями або специфічними потребами традиційні пристрої введення можуть становити серйозну перешкоду. Безконтактні інтерфейси розширюють можливості взаємодії

з цифровим світом для широкого кола користувачів, включаючи тих, хто має фізичні обмеження.

Сфера професійної діяльності також активно трансформується під впливом нових інтерфейсних рішень. Презентації, конференції, віддалені наради стали невід'ємною частиною ділової культури, а в умовах глобалізації та переходу до гібридних форматів роботи зростає потреба в технологіях, які забезпечують природну та безперебійну комунікацію. Жестові інтерфейси дозволяють спікерам зосередитися на змісті виступу та взаємодії з аудиторією, а не на технічних аспектах керування презентацією.

Не менш значущим є розвиток індустрії стрімінгу та виробництва цифрового контенту. Творці онлайн-відео, стрімери, влогери постійно шукають способи підвищення якості своїх трансляцій та оптимізації робочого процесу. Безконтактне керування мультимедійними функціями дозволяє їм створювати більш професійний контент, плавно перемикаючись між сценами та налаштуваннями без відволікаючих звуків клавіатури чи переривання спілкування з аудиторією.

Таким чином, розробка мультимедійної клавіатури на основі датчика жестів відповідає ключовим технологічним, соціальним та економічним тенденціям сучасності, пропонуючи інноваційне рішення для актуальних викликів у сфері взаємодії людини з комп'ютером.

1.2 Аналіз існуючих рішень керування комп'ютером жестами

У сучасному світі, де технології стрімко розвиваються та трансформують наше повсякденне життя, інтерфейси взаємодії між людиною та комп'ютером набувають дедалі більшого значення. Традиційні пристрої введення, такі як клавіатура та миша, хоча й залишаються домінуючими, поступово доповнюються та в деяких контекстах замінюються більш інтуїтивними та природними способами взаємодії. Серед таких інновацій особливе місце займають системи керування за допомогою жестів – технології, що дозволяють користувачам маніпулювати цифровим контентом через рухи тіла, зокрема рук та пальців, без прямого фізичного контакту з пристроєм.

Технологія жестового керування пройшла значний шлях еволюції від експериментальних лабораторних розробок до масових комерційних продуктів. Її витоки сягають 1980-х років, коли перші дослідники комп'ютерних інтерфейсів почали експериментувати з системами, здатними розпізнавати прості жести. Однак справжній прорив у цій галузі відбувся на початку 2000-х років із розвитком більш потужних алгоритмів комп'ютерного зору та обробки зображень, а також зі зниженням вартості та підвищенням якості сенсорних пристроїв.

Ринок сучасних рішень для жестового керування умовно можна розділити на кілька категорій, кожна з яких має свої особливості, переваги та обмеження. Перша категорія – це системи, засновані на оптичному розпізнаванні за допомогою камер. Найбільш відомим прикладом такої технології є Microsoft Kinect, випущений у 2010 році як доповнення до ігрової консолі Xbox 360.

Рисунок 1.1 – Камера Kinect [17]

Kinect об'єднував RGB-камеру з інфрачервоним проєктором та спеціальним сенсором для створення тривимірної карти простору, що дозволяло точно відстежувати рухи всього тіла користувача.

Рисунок 1.2 – ІЧ зображення камери Kinect [17]

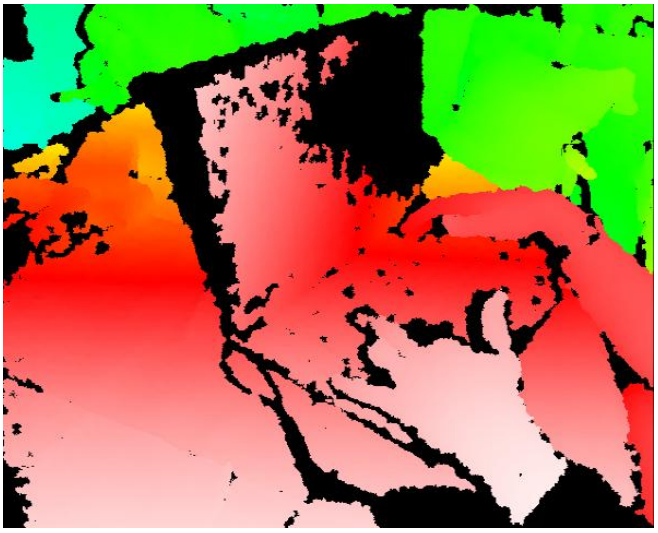

Рисунок 1.3 – Карта глибини камери Kinect [17]

Хоча Microsoft зрештою припинила виробництво Kinect для споживчого ринку, технологія знайшла застосування в багатьох галузях.

Intel RealSense представляє собою іншу потужну оптичну платформу, що поєднує традиційну RGB-камеру з датчиками глибини для точного відстеження об'єктів у тривимірному просторі.

Рисунок 1.4 – Intel RealSense [6]

Ця технологія дозволяє розпізнавати не лише великі рухи тіла, але й дрібні жести пальців, що відкриває нові можливості для створення інтуїтивних інтерфейсів керування.

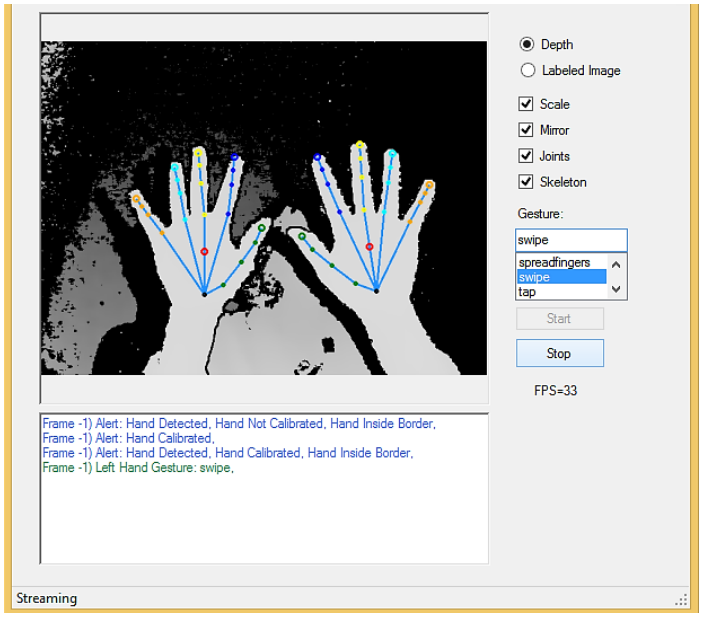

Рисунок 1.5 – Intel RealSense SDK: Hands Viewer [6]

Системи на базі Intel RealSense активно використовуються в роботехніці, медичній візуалізації та віртуальній реальності.

Leap Motion пропонує компактний пристрій, оптимізований саме для відстеження рухів рук та пальців. Використовуючи інфрачервоні світлодіоди та камери, цей пристрій створює детальну модель рук користувача в реальному часі, дозволяючи їм взаємодіяти з віртуальними об'єктами з високою точністю.

Рисунок 1.6 – Leap Motion [9]

Особливо актуальною ця технологія стала з розвитком систем віртуальної та доповненої реальності, де природність взаємодії відіграє ключову роль у забезпеченні ефекту присутності. Ця система кріпиться до шолому віртуальної реальності, та считує рухи рук перед собою.

Друга категорія жестових інтерфейсів – це рішення, засновані на носимих пристроях. Myo Armband від Thalmic Labs був одним із перших комерційно успішних продуктів у цій категорії.

Рисунок 1.7 – Myo Armband [11]

Цей браслет реєстрував електричну активність м'язів передпліччя для визначення жестів руки та пальців. Хоча точність такого підходу була нижчою порівняно з оптичними системами, Myo не вимагав прямої видимості між пристроєм та користувачем і міг працювати в умовах різного освітлення.

Третя категорія – це радарні технології, серед яких особливо виділяється Project Soli від Google.

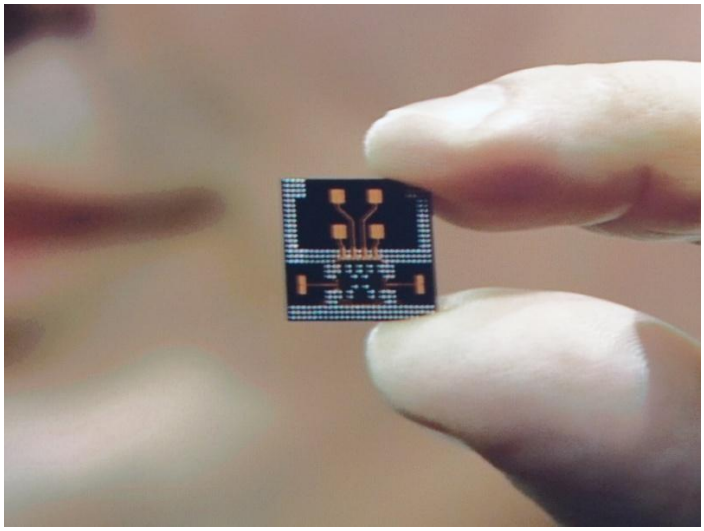

Рисунок 1.8 – Project Soli [18]

Ця мініатюрна радарна система здатна розпізнавати дрібні рухи пальців на відстані до 30 см від пристрою. Перевагою радарних технологій є їхня здатність працювати через тонкі матеріали та в умовах, де оптичні системи могли б давати збої. Технологія Project Soli знайшла застосування в смартфонах Google Pixel, дозволяючи користувачам керувати деякими функціями телефону без прямого дотику до екрана.

Окрім апаратної частини, критичним компонентом будь-якої системи жестового керування є програмне забезпечення для розпізнавання та інтерпретації жестів. У цій галузі також спостерігаються значні інновації. Алгоритми комп'ютерного зору, особливо засновані на глибоких нейронних мережах, демонструють вражаючі результати в розпізнаванні складних жестів у реальному часі. Бібліотеки OpenCV, TensorFlow та MediaPipe надають розробникам потужні інструменти для реалізації жестових інтерфейсів у різних застосунках.

Згідно з прогнозами аналітичних агенцій, ринок технологій жестового керування продовжить стрімке зростання в найближчі роки. Особливо перспективними виглядають гібридні системи, що поєднують переваги різних технологій розпізнавання – оптичних, радарних, електроміографічних – для досягнення оптимального балансу між точністю, зручністю та енергоефективністю.

В контексті мікроконтролерних рішень, таких як Arduino з датчиком PAJ7620, що є фокусом даного дослідження, комерційні системи жестового керування можуть здаватися надмірно складними та дорогими. Однак вони задають важливі орієнтири щодо можливостей та обмежень жестових інтерфейсів. Розуміння того, як реалізовані професійні рішення, дозволяє більш ефективно проєктувати спрощені системи, що зберігають ключові переваги жестового керування при значно нижчій складності та вартості.

Технології жестового керування комп'ютером пройшли значний шлях розвитку та диверсифікації. Від експериментальних систем до масових продуктів, від простих рухів до складних жестів – ця еволюція відображає загальну тенденцію до створення більш природних та інтуїтивних інтерфейсів. Аналіз існуючих рішень

дозволяє не лише оцінити поточний стан галузі, але й визначити перспективні напрямки для подальших досліджень та розробок, особливо в контексті спеціалізованих систем для керування мультимедійними функціями.

1.3 Принцип дії оптичного датчика жестів PAJ7620

У світі вбудованих систем, де кожен міліметр простору та мікроват енергії має значення, датчик жестів PAJ7620 представляє собою унікальне поєднання компактності, енергоефективності та функціональності. Цей мініатюрний оптичний сенсор, розроблений компанією PixArt Imaging Inc., став справжнім проривом у демократизації технологій жестового керування, дозволивши навіть простим мікроконтролерним системам розпізнавати жести людини без необхідності у складних обчислювальних алгоритмах та потужних процесорах.

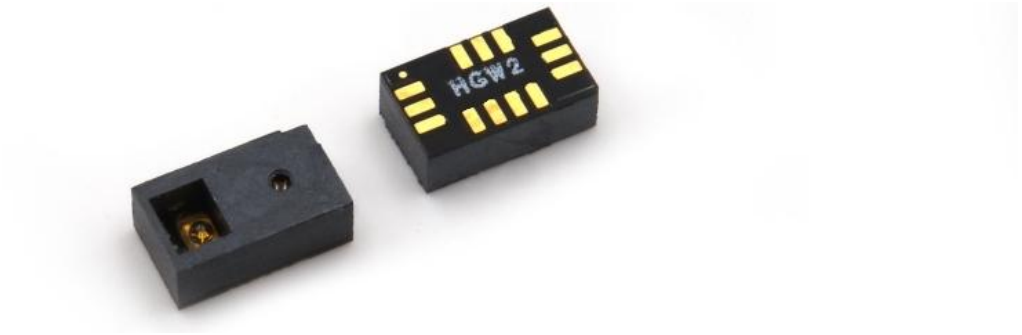

Рисунок 1.9 – Датчик жестів PAJ7620U2 [12]

PAJ7620 базується на технології інфрачервоного (ІЧ) оптичного розпізнавання. Це система, що інтегрує ІЧ-випромінювачі, масив фотодіодів та спеціалізовану систему обробки сигналів. Саме ця інтеграція дозволяє датчику виконувати розпізнавання жестів автономно, передаючи на мікроконтролер вже готові результати, а не первинні дані, які потребують подальшої обробки.

Принцип дії PAJ7620 базується на технології оптичного потоку – методі вимірювання руху об'єктів у полі зору сенсора на основі змін інтенсивності світла, що фіксується матрицею фотодіодів. Датчик активно випромінює інфрачервоне світло через вбудовані світлодіоди, яке відбивається від руки користувача та повертається до сенсора. Спеціалізована матриця з 30x30 фотодіодів, яка фіксує це

відбите світло, створюючи примітивне, але інформативне зображення об'єкта перед сенсором.

Ключовою особливістю PAJ7620 є його здатність аналізувати послідовні кадри таких зображень для виявлення руху. Вбудований процесор датчика обчислює вектори руху між послідовними кадрами, визначаючи напрямок, швидкість та характер переміщення об'єкта. Ця інформація обробляється за допомогою вбудованих алгоритмів розпізнавання образів, які порівнюють отримані дані з заздалегідь запрограмованими шаблонами жестів.

Архітектурно PAJ7620 можна розділити на три основні функціональні блоки. Перший блок – це система освітлення, що складається з ІЧ-світлодіодів, які працюють на довжині хвилі близько 940 нм, що знаходиться поза спектром видимого людським оком світла. Це дозволяє датчику функціонувати непомітно для користувача, не відволікаючи його видимим світлом

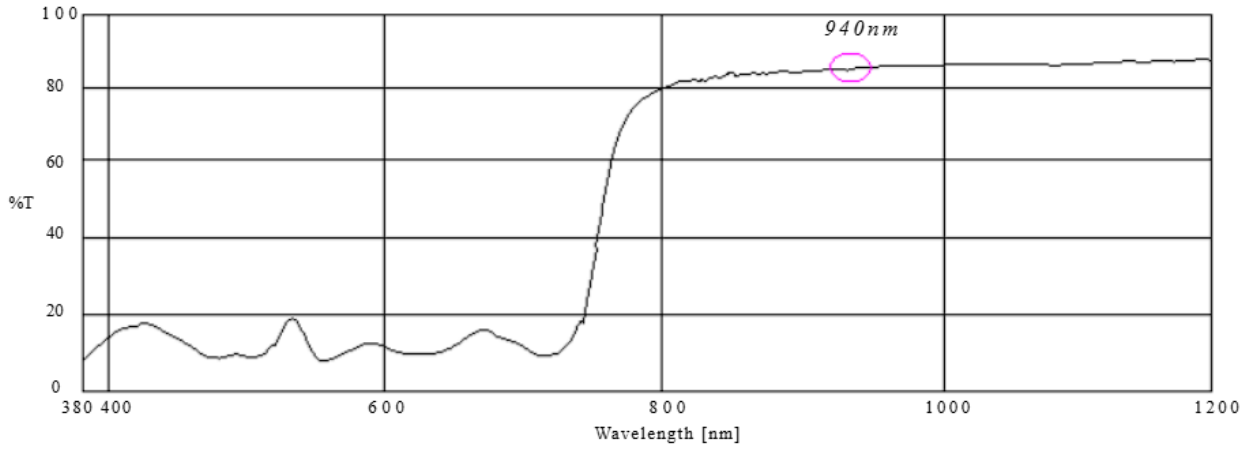

Рисунок 1.10 – Чутливість до ІЧ-випромінення [13]

Другий блок – це матриця фотодіодів, оптимізована для роботи саме з інфрачервоним спектром, що забезпечує високу чутливість та стійкість до перешкод від видимого світла. Третій блок – це процесор обробки сигналів та розпізнавання образів, який виконує алгоритми фільтрації шумів, сегментації зображення та класифікації жестів.

Датчик PAJ7620 спроєктований з урахуванням обмежень типових вбудованих систем. Він працює від напруги 3.3 В, споживаючи при цьому лише близько 5 мА у активному режимі, що робить його дуже економічним

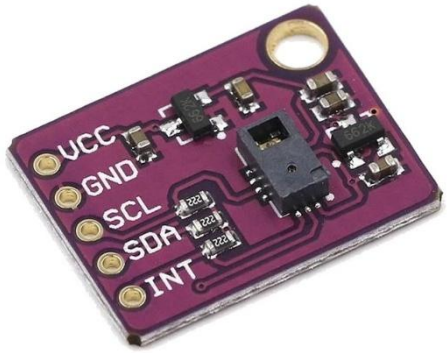

Рисунок 1.11 – Модуль з датчиком PAJ7620U2 [12]

Стандартна конфігурація PAJ7620 дозволяє розпізнавати вісім базових жестів: рух вгору, вниз, вліво, вправо, вперед (наближення до сенсора), назад (віддалення від сенсора), за годинниковою стрілкою, проти годинникової стрілки та коливальний рух. Цього набору жестів достатньо для реалізації базового управління мультимедійними функціями, такими як регулювання гучності, перемикання треків чи каналів, паузи та відновлення відтворення.

Важливою характеристикою PAJ7620 є його робочий діапазон. Оптимальна відстань для розпізнавання жестів становить від 5 до 20 см від сенсора, що робить його ідеальним для інтеграції у пристрої, з якими користувач взаємодіє на близькій відстані. Кут огляду сенсора становить приблизно 60°, що забезпечує достатню свободу для виконання жестів без необхідності точного позиціонування руки.

У підсумку, оптичний датчик жестів PAJ7620 представляє собою компактне, енергоефективне та доступне рішення для інтеграції базового жестового керування у вбудовані системи. Його принцип дії, заснований на аналізі інфрачервоного оптичного потоку, дозволяє надійно розпізнавати набір стандартних жестів без необхідності у складних обчисленнях. Простота інтеграції через інтерфейс I2C та наявність готових програмних бібліотек роблять цей датчик ідеальним вибором для проєктів на базі Arduino Pro Micro, спрямованих на розробку систем керування мультимедійними функціями за допомогою жестів.

1.4 Можливості використання Arduino Pro Micro

Arduino Pro Micro вирізняється серед інших плат Arduino насамперед своєю архітектурою. Мікроконтролер ATmega32u4, який лежить в основі плати, інтегрує USB-контролер безпосередньо в чіп, що усуває необхідність у додатковому перетворювачі USB-UART. Ця особливість має кардинальне значення для проєктів, що потребують безпосередньої взаємодії з комп'ютером, оскільки дозволяє платі емулювати різноманітні периферійні пристрої введення, такі як клавіатура або миша.

Рисунок 1.12 – Arduino Pro Micro [14]

Технічні характеристики Arduino Pro Micro включають:

– мікроконтролер ATmega32u4 з тактовою частотою 16 МГц;

– 32 кбайт флеш-пам'яті (з яких 4 кбайт використовуються для завантажувача);

– 2.5 кбайт SRAM та 1 кбайт EEPROM;

– 18 цифрових входів/виходів (з яких 5 можуть використовуватися як ШІМ-виходи);

– 9 аналогових входів;

– інтерфейси UART, I2C та SPI для комунікації з периферійними пристроями.

Ці характеристики забезпечують достатню обчислювальну потужність і пам'ять для обробки даних з датчика жестів PAJ7620, інтерпретації розпізнаних жестів та їх конвертації у відповідні команди для керування мультимедійними функціями.

Емуляція HID-пристроїв як ключова перевага. Найбільш значущою перевагою Arduino Pro Micro для реалізації мультимедійної клавіатури на основі жестів є здатність емулювати HID-пристрої. Завдяки вбудованому USB-контролеру, Pro Micro може функціонувати як стандартна комп'ютерна клавіатура, миша або джойстик без необхідності в додатковому апаратному забезпеченні. Ця можливість відкриває широкі перспективи для створення альтернативних інтерфейсів введення.

Наприклад, жест «змахування вправо» може бути інтерпретований як команда «наступний трек» у медіаплеєрі, що технічно реалізується як натискання клавіші «Next Track» або комбінації «Ctrl+Right». Аналогічно, жести можуть відповідати командам регулювання гучності, паузи/відтворення, перемотування та інших мультимедійних функцій.

Програмна гнучкість та розширюваність. Arduino Pro Micro програмується за допомогою стандартного середовища розробки Arduino IDE, яке використовує мову програмування, засновану на C/C++. Це забезпечує широкі можливості для реалізації складних алгоритмів обробки даних з датчика жестів, їх інтерпретації та конвертації в команди керування мультимедійними програмами. Що дасть змогу реалізувати:

– налаштування чутливості розпізнавання жестів відповідно до вимог користувача;

– створення персоналізованих профілів з різними наборами жестів для різних мультимедійних програм.

Енергоефективність і автономність. Arduino Pro Micro характеризується відносно низьким енергоспоживанням, що робить її придатною для створення автономних пристроїв, які можуть працювати від батарей протягом тривалого часу. Мікроконтролер ATmega32u4 має декілька режимів енергозбереження, які можуть

бути використані для оптимізації енергоспоживання залежно від інтенсивності використання пристрою.

Для системи керування мультимедійними функціями за допомогою жестів можна реалізувати алгоритми, які переводять пристрій у режим зниженого енергоспоживання під час відсутності активності, і «пробуджують» його при появі руху перед датчиком. Це дозволить створити дійсно автономний пристрій, який можна використовувати від живлення акумулятора

Компактність і можливості інтеграції. Arduino Pro Micro має компактні розміри (68,6 мм × 53,3 мм), що робить її придатною для інтеграції в невеликі корпуси. Це важливо для створення ергономічного пристрою керування мультимедійними функціями, який повинен бути зручним у використанні та не займати багато місця на робочому столі користувача.

Підсумовуючи можна сказати, що Arduino Pro Micro є оптимальною платформою для реалізації мультимедійної клавіатури на основі датчика жестів PAJ7620. Її унікальна здатність емулювати HID-пристрої, сумісність з датчиком PAJ7620 через інтерфейс I2C, програмна гнучкість та можливості розширення функціональності створюють міцну основу для розробки системи керування мультимедійними функціями.

1.5 Алгоритми розпізнавання та інтерпретації жестів у різних системах

Розпізнавання жестів як спосіб взаємодії з електронними пристроями відкриває перед нами дивовижний світ можливостей, де звичні рухи рук набувають нового значення, перетворюючись на команди, що керують цифровим середовищем. Ця технологія, яка ще кілька десятиліть тому здавалася фантастикою, сьогодні стає невід'ємною частиною повсякденного життя, інтегруючись у смартфони, ігрові консолі, системи розумного дому та мультимедійні пристрої. Саме алгоритми розпізнавання та інтерпретації жестів становлять інтелектуальне ядро таких систем, перетворюючи абстрактні рухи на конкретні дії в цифровому світі.

Історія розвитку алгоритмів розпізнавання жестів відображає еволюцію підходів до вирішення складної проблеми перетворення просторових рухів у цифрову інформацію. Перші системи базувалися на простих порогових значеннях та евристичних правилах, що дозволяли розпізнавати лише найпростіші жести в контрольованих умовах. З часом з'явилися статистичні методи, які використовували ймовірнісні моделі для інтерпретації рухів, а пізніше – системи на основі комп'ютерного зору, що аналізували послідовності зображень для виявлення патернів руху. Сучасні алгоритми розпізнавання жестів часто поєднують різні підходи, використовуючи як прості детерміністичні методи для базових жестів, так і складні моделі машинного навчання для розпізнавання більш нюансованих рухів.

Одним з ключових викликів при розробці алгоритмів розпізнавання жестів є забезпечення їх стійкості до індивідуальних особливостей виконання жестів різними користувачами. Люди виконують одні й ті ж жести з різною швидкістю, амплітудою та точністю, що створює значну варіативність вхідних даних. Ефективні алгоритми повинні абстрагуватися від цих індивідуальних особливостей, зосереджуючись на ключових характеристиках жесту.

Інтерпретація розпізнаних жестів – це наступний логічний крок, який перетворює абстрактний рух на конкретну дію. Ця інтерпретація значною мірою залежить від контексту використання та поточного стану системи. У мультимедійних системах інтерпретація жестів часто реалізується через систему маппінгу, де кожному розпізнаному жесту або комбінації жестів відповідає певна команда або послідовність команд. Цей маппінг може бути статичним, заздалегідь запрограмованим, або динамічним, що змінюється залежно від контексту чи навіть навчається на основі поведінки користувача.

Контекстна інтерпретація жестів є особливо цікавим напрямком розвитку алгоритмів. Вона дозволяє одному й тому ж жесту мати різне значення залежно від поточної ситуації або активної програми. Така контекстна чутливість робить інтерфейс більш інтуїтивним та природним, дозволяючи користувачу застосовувати ті самі природні рухи в різних ситуаціях.

Алгоритми розпізнавання комбінацій жестів та послідовностей відкривають ще ширші можливості для взаємодії з мультимедійними системами. Подібно до того, як комбінації клавіш розширюють функціональність клавіатури, послідовності та комбінації жестів дозволяють реалізувати більш складні команди без необхідності вводити додаткові базові жести. Наприклад, жест «змахування вправо» з подальшим утриманням руки в полі зору датчика може інтерпретуватися як команда перемотування вперед, де тривалість утримання визначає швидкість перемотування. Такі комплексні інтерпретації вимагають більш складних алгоритмів, які враховують не лише тип жесту, але й часові характеристики, послідовність та взаємозв'язок різних рухів.

Мультимодальність – ще один перспективний напрямок розвитку алгоритмів розпізнавання жестів, який передбачає поєднання даних від різних типів сенсорів для підвищення точності та розширення можливостей інтерфейсу. Наприклад, комбінація інфрачервоного датчика жестів PAJ7620 з акселерометром або гіроскопом може дозволити розпізнавати не лише рухи руки в площині перед датчиком, але й просторові жести, що включають нахили, обертання та переміщення в тривимірному просторі. Алгоритми злиття даних від різних сенсорів повинні враховувати різну природу, частоту та точність цих даних, синхронізувати їх у часі та витягувати комплексні характеристики, що відображають суть жесту.

В історичній перспективі ми стоїмо на порозі нової ери взаємодії з комп'ютерами, де жести стануть так само природними і звичними, як натискання клавіш або рухи мишею. Вони стають все більш адаптивними, контекстно-чутливими та персоналізованими, враховуючи індивідуальні особливості, переваги та звички користувачів.

1.6 Специфікація вимог до програмного забезпечення мультимедійної клавіатури на основі датчика жестів

Загальний опис. Програмне забезпечення призначене для роботи з системою безконтактного управління мультимедійними функціями комп'ютера на базі датчика жестів PAJ7620 та мікроконтролера Arduino Pro Micro. Система має

забезпечувати розпізнавання жестів користувача та їх перетворення на відповідні мультимедійні команди, які передаються на комп'ютер через інтерфейс HID.

Функціональні вимоги до програмного забезпечення:

– ініціалізація та налаштування датчика жестів PAJ7620;

– зчитування та розпізнавання жестів користувача (мінімум 8 базових жестів);

– перетворення розпізнаних жестів у команди мультимедійного управління;

Нефункціональні вимоги:

– латентність розпізнавання жестів: не більше 500 мс;

– точність розпізнавання жестів: не менше 90 % в нормальних умовах освітлення;

– стабільна робота без збоїв протягом кількох годин безперервного використання;

– сумісність з операційними системами: Windows 10/11;

Інтерфейси та архітектура:

– взаємодія з датчиком PAJ7620 через інтерфейс I2C;

– підключення до комп'ютера через USB-інтерфейс;

– використання бібліотеки Wire.h для комунікації через I2C.

Тестування та критерії приймання:

– успішне розпізнавання всіх визначених жестів з точністю не менше 90 %;

– коректна передача мультимедійних команд;

– стабільна робота протягом тривалого часу без перезавантаження.

Згідно з визначеними вимогами буде розроблена система безконтактного мультимедійного управління, що інтегрує датчик жестів PAJ7620 з мікроконтролером Arduino Pro Micro для реалізації високоточного розпізнавання рухів користувача. Створене програмне забезпечення забезпечить швидку обробку жестів та їх конвертацію

Висновки до розділу 1

У першому розділі було проведено комплексний аналіз сучасних технологій жестового керування, що активно розвиваються у напрямку створення природних і зручних інтерфейсів взаємодії людини з електронними пристроями. Розглянуто основні підходи до реалізації таких систем, зокрема оптичні, електроміографічні та радарні рішення, кожне з яких має свої переваги, обмеження та сфери застосування.

Також проаналізовано технічні можливості плати Arduino Pro Micro, яка завдяки інтегрованому USB-контролеру забезпечує просту емуляцію HID-пристроїв, що робить її ідеальним варіантом для створення систем мультимедійного керування на основі жестів. У поєднанні з датчиком PAJ7620, ця платформа дозволяє створити доступне, енергоефективне та зручне рішення для управління комп’ютером або іншими пристроями без фізичного контакту.

Таким чином, результати теоретичного аналізу підтверджують доцільність обраного підходу до реалізації системи керування мультимедійними функціями за допомогою жестів, який поєднує ефективність, простоту реалізації та функціональну гнучкість.

2 ПРОЄКТУВАННЯ МУЛЬТИМЕДІЙНОЇ КЛАВІАТУРИ НА ОСНОВІ PAJ7620

2.1 Алгоритм функціонування системи розпізнавання жестів

Розробка ефективної системи керування мультимедійними функціями за допомогою жестів вимагає не лише оптимального підбору апаратних компонентів, але й розробки продуманого алгоритму функціонування, що забезпечить точне розпізнавання команд користувача та їхню правильну інтерпретацію. У цьому розділі було детально розглянуто алгоритмічну основу системи, представлену на структурній схемі, проаналізуємо кожен етап обробки жестових команд.

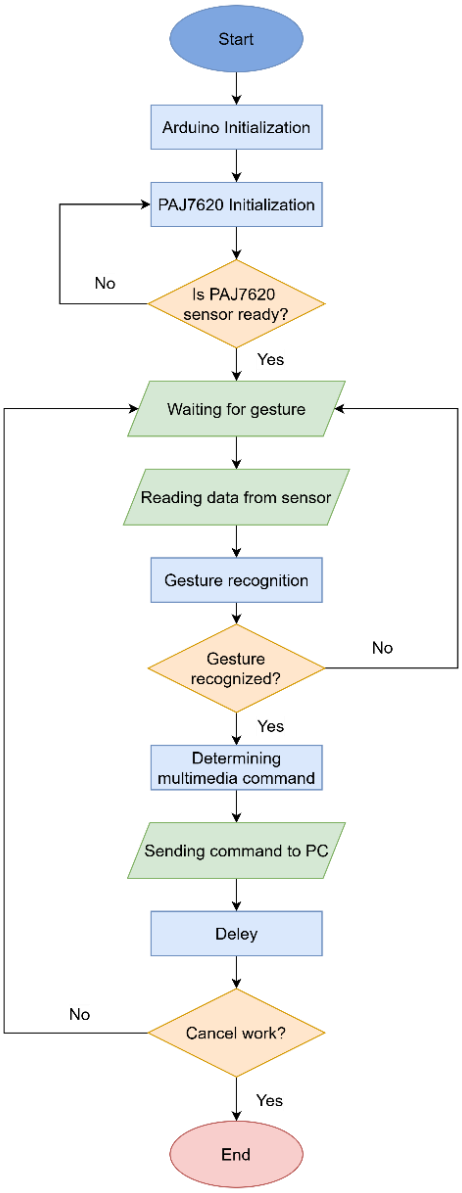

Рисунок 2.1 – Блок-схема алгоритму

Блок-схема представляє комплексну візуалізацію алгоритму роботи системи, покликаного забезпечити надійне розпізнавання жестів і відповідне виконання команд. Процес починається з ініціалізації як Arduino Pro Micro, так і датчика PAJ7620, встановлення необхідних протоколів зв'язку та параметрів конфігурації. На цьому етапі відбувається налаштування всіх апаратних та програмних компонентів до робочого стану:

Ініціалізація мікроконтролера Arduino Pro Micro – налаштування тактової частоти, конфігурація пінів введення/виведення, встановлення початкових значень регістрів. Налаштування інтерфейсу I²C – активація шини I²C, встановлення швидкості передачі даних (зазвичай 100 кГц або 400 кГц), активація внутрішніх підтягуючих резисторів. Ініціалізація датчика PAJ7620 – послідовність команд для пробудження датчика з режиму сну, налаштування його внутрішніх регістрів для оптимальної роботи:

Ретельна ініціалізація кожного модуля є фундаментом для стабільної роботи системи. Особливу увагу приділено налаштуванню датчика PAJ7620, оскільки коректність його функціонування безпосередньо впливає на точність розпізнавання жестів. Послідовність команд ініціалізації відповідає специфікації виробника та оптимізована для роботи в умовах керування мультимедійними функціями.

Після ініціалізації система входить в безперервний робочий цикл, де активно очікує на жести користувача. При виявленні руху датчик PAJ7620 фіксує дані жесту, які потім зчитуються Arduino Pro Micro для обробки. На етапі розпізнавання отримані дані порівнюються з попередньо визначеними шаблонами жестів, що зберігаються в системі.

Після завершення ініціалізації система переходить до наступного кроку, де на цьому етапі датчик активно сканує простір перед собою, використовуючи інфрачервоне випромінювання для виявлення об'єктів та їх руху.

Процес збору даних про жести включає:

– активне сканування: датчик випромінює інфрачервоні імпульси з частотою близько 240 Гц;

– фіксація відбитого світла: матриця 8x8 фотодіодів реєструє інтенсивність відбитого світла, формуючи примітивне «зображення» руки користувача;

– попередня обробка даних: вбудований процесор датчика виконує фільтрацію шумів та обчислення вектора руху об'єкта між послідовними кадрами;

– класифікація жестів: на основі аналізу напрямку, швидкості та характеру руху датчик визначає, який саме жест виконано з передбаченого набору.

На цьому етапі мікроконтролер Arduino Pro Micro активно взаємодіє з датчиком PAJ7620 для отримання результатів розпізнавання жестів. Процес зчитування даних через I²C включає наступні кроки:

– формування запиту: мікроконтролер ініціює комунікацію з датчиком, використовуючи його 7-бітну адресу (зазвичай 0x73);

– вказівка регістру: мікроконтролер вказує адресу регістра, з якого потрібно прочитати дані (для PAJ7620 регістр жестів має адресу 0x43);

– зчитування даних: мікроконтролер зчитує байт даних з вказаного регістра, що містить інформацію про розпізнаний жест;

– повторне опитування для підвищення надійності розпізнавання мікроконтролер може зчитувати дані кілька разів протягом короткого інтервалу часу.

Важливим аспектом цього етапу є оптимізація частоти опитування датчика. Занадто часте опитування може призвести до помилкового розпізнавання одного жесту як кількох послідовних, тоді як занадто рідке опитування може призвести до пропуску швидких жестів. У нашій системі встановлено інтервал опитування 100 мс, що протестовано і визначено як оптимальний баланс між чутливістю та стійкістю до помилок.

Після успішного розпізнавання алгоритм визначає, яка мультимедійна команда відповідає ідентифікованому жесту. Потім ці команди передаються на підключений комп'ютер через інтерфейс HID, імітуючи стандартні входи мультимедійної клавіатури, які керують такими функціями, як відтворення/пауза, збільшення/зменшення гучності, наступний/попередній трек та інші.

Емуляція HID-команд – Arduino Pro Micro використовує бібліотеку HID для емуляції натискання відповідних мультимедійних клавіш, що розпізнаються операційною системою комп'ютера.

Після виконання необхідних дій алгоритм повертається до етапу збору даних про жести, утворюючи нескінченний цикл роботи системи до моменту її вимкнення.

2.2 Схема підключення плати Arduino Pro Micro та датчика жестів PAJ7620

У створенні системи безконтактного управління мультимедійними функціями комп'ютера ключову роль відіграє правильне підключення апаратних компонентів. Розглянемо детально схему підключення датчика жестів PAJ7620 до мікроконтролера Arduino Pro Micro, а також особливості інтеграції даної системи з комп'ютерною платформою.

Запропонована система складається з двох основних апаратних компонентів: мікроконтролера Arduino Pro Micro та датчика жестів PAJ7620. Кожен з цих елементів має свої унікальні характеристики, що дозволяють реалізувати необхідний функціонал.

Arduino Pro Micro представляє собою компактну плату на базі мікроконтролера ATmega32U4, що працює на частоті 16 МГц. Ключовою перевагою цього мікроконтролера є вбудована підтримка USB-інтерфейсу, що дозволяє Arduino Pro Micro емулювати роботу HID-пристроїв (Human Interface Device), таких як клавіатура, миша або контролер мультимедіа. Плата має 18 цифрових входів/виходів, з яких 7 можуть використовуватися як ШІМ-виходи, а 12 – як аналогові входи. Компактні розміри (33 × 18 мм) роблять цю плату ідеальним рішенням для проєктів з обмеженим простором розміщення.

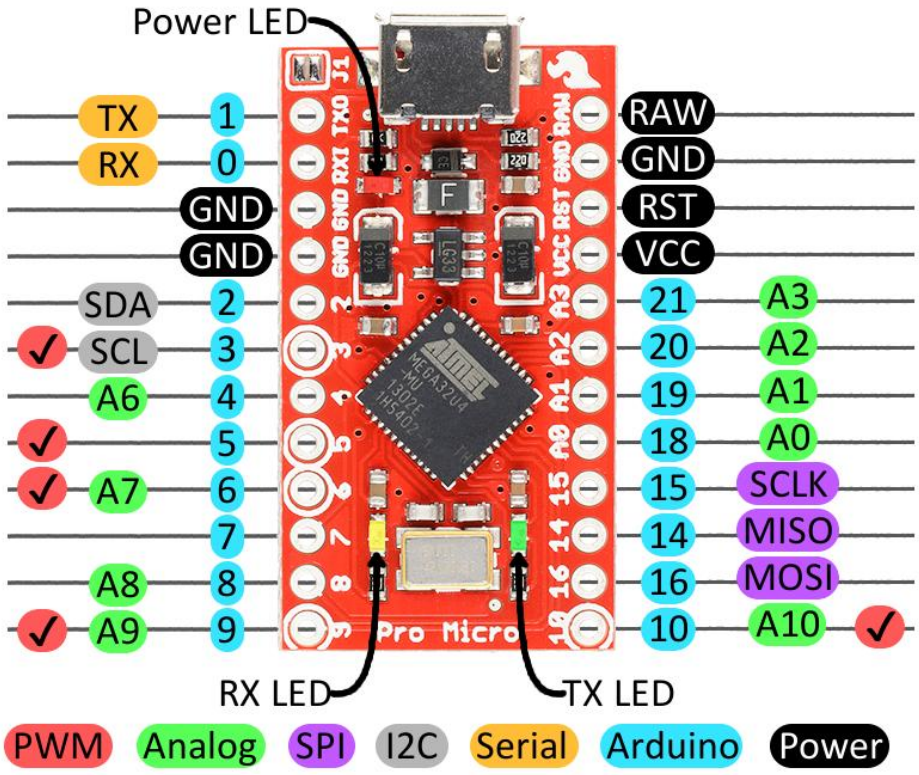

Рисунок 2.2 – Pinout Pro Micro [3]

Датчик жестів PAJ7620 – це інтегральна схема розпізнавання жестів, оснащена інфрачервоним сенсором та вбудованим модулем обробки даних. Датчик здатний розпізнавати до 9 базових жестів: рух вгору, вниз, вліво, вправо, рух вперед, назад, колові рухи за годинниковою та проти годинникової стрілки, а також хвилеподібні рухи. PAJ7620 використовує інтерфейс I2C для комунікації з мікроконтролером, що дозволяє здійснювати передачу даних за допомогою лише двох сигнальних ліній.

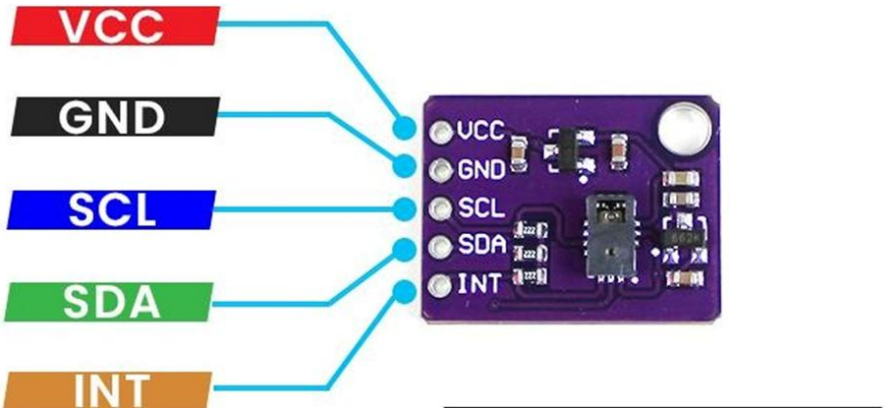

Рисунок 2.3 – Pinout PAJ7620 [12]

Підключення датчика PAJ7620 до Arduino Pro Micro здійснюється через інтерфейс I2C, який забезпечує двонаправлену передачу даних між пристроями з використанням мінімальної кількості виводів.

Схема підключення живлення включає наступні з'єднання:

– вивід VCC датчика PAJ7620 підключається до виводу VCC (5V) Arduino Pro Micro;

– вивід GND датчика з'єднується з виводом GND Arduino Pro Micro.

Схема підключення передачі даних включає наступні з'єднання:

– вивід SCL (Serial Clock Line) датчика підключається до піна 3 (SCL) Arduino Pro Micro;

– вивід SDA (Serial Data Line) датчика з'єднується з піном 2 (SDA) Arduino Pro Micro.

Загальну принципову схему пристрою продемонстровано на рисунку нижче.

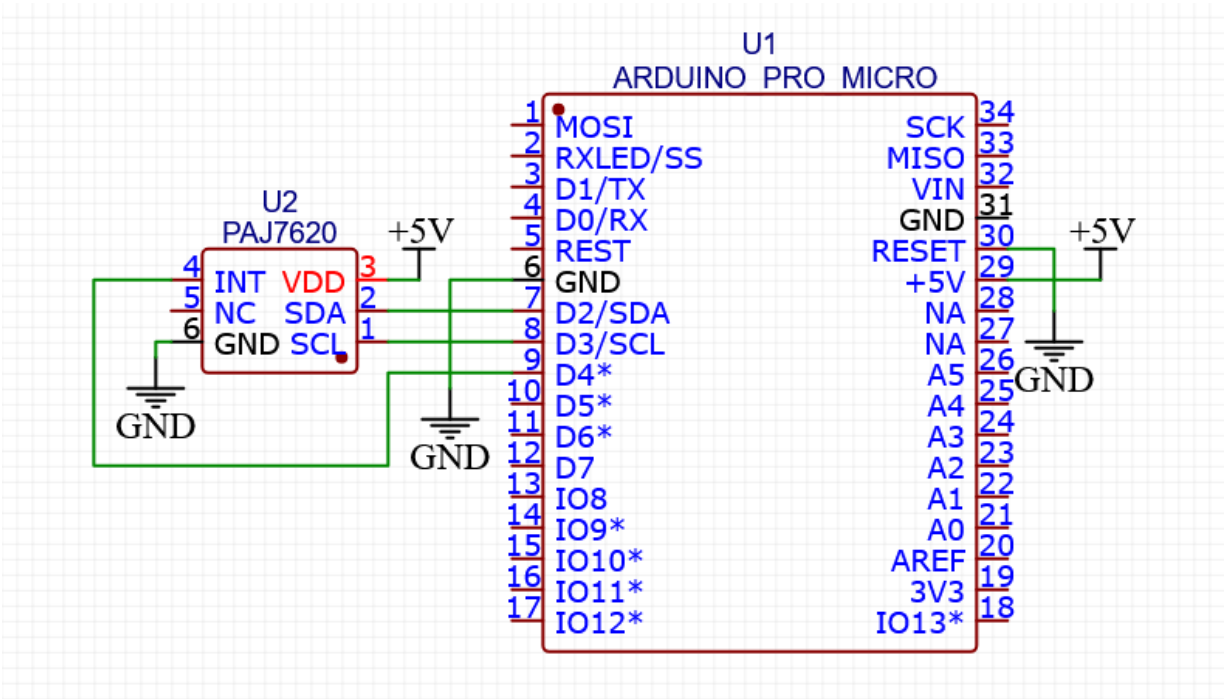

Рисунок 2.4 – Принципова схема

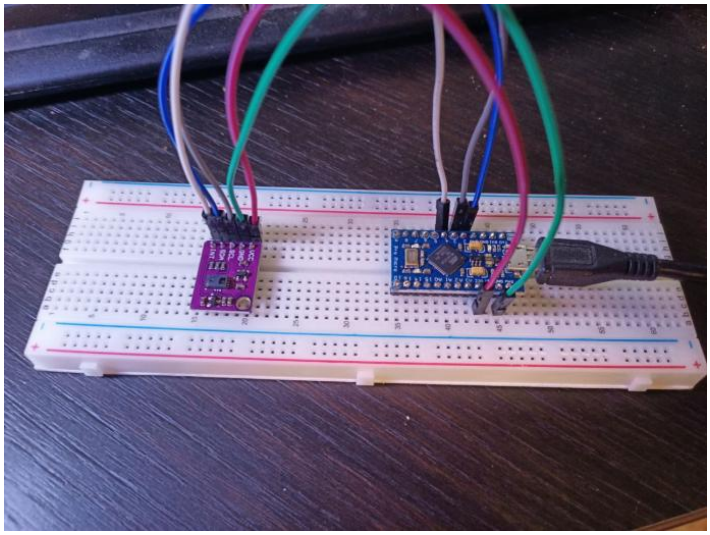

Рисунок 2.5 – Загальний вигляд прототипу пристрою

У підсумку, розроблена схема підключення плати Arduino Pro Micro та датчика жестів PAJ7620 забезпечує оптимальне балансування між простотою реалізації та надійністю роботи, що робить її придатною для використання як в аматорських, так і в професійних проєктах створення систем безконтактного управління.

2.3 Функціональна структура десктопного застосунку

Основною функцією застосунку є перетворення жестів у системні команди, зокрема комбінації клавіш або запуск зовнішніх програм.

Застосунок функціонує у фоновому режимі, забезпечуючи постійний прийом даних через послідовний порт. Після отримання коду жесту від пристрою, програма виконує пошук у внутрішньому списку асоціацій. Якщо знайдено відповідність, запускається пов’язана дія. Таким чином реалізується логіка асоціювання жестів з командами.

Користувач має змогу налаштувати ці відповідності за допомогою графічного інтерфейсу. Усі конфігурації зберігаються у вигляді JSON-файлу. Застосунок підтримує редагування, додавання нових асоціацій, а також відновлення стандартних параметрів.

Для формалізованого опису функціональності застосунку використано діаграму варіантів використання (use case diagram). Вона ілюструє основні взаємодії між користувачем і програмою.

На рисунку 2.6 зображено діаграму варіантів використання. У центрі діаграми розміщено актор – користувача, який ініціює основні дії у програмі.

Доступні варіанти використання наведені у вигляді прямих зв’язків між актором та відповідними функціями:

– додати комбінацію клавіш. Користувач вводить новий жест або обирає існуючий, після чого вказує комбінацію клавіш або команду, яка має виконуватись. Програма оновлює список відповідностей у пам’яті та зберігає зміни у конфігураційному файлі;

– редагувати комбінацію клавіш. Дозволяє змінювати вже існуючі відповідності. Користувач може оновити дію, яка пов’язана з конкретним жестом, або повністю змінити призначення цього жесту;

– скинути до дефолтних налаштувань. Програма замінює поточний список відповідностей на базову конфігурацію, передбачену розробником. Це забезпечує відновлення працездатного стану у разі помилок під час налаштування.

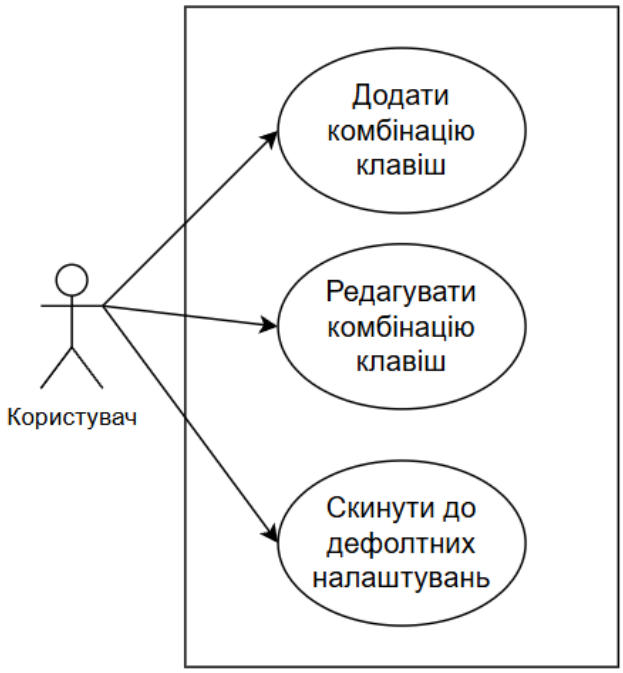

Рисунок 2.6 – Діаграма варіантів використання

У структурі застосунку реалізовано зберігання асоціацій між жестами та відповідними діями у вигляді конфігураційного файлу. Для формального представлення цієї структури використовується ER-діаграма, яка ілюструє взаємозв’язки між сутностями, що беруть участь у збереженні і обробці конфігурації. Основними сутностями є Gesture, Action та ConfigFile.

Сутність Gesture представляє жест, що надходить до програми у вигляді символьного або числового коду, що визначає напрямок або тип жесту, зчитаного з сенсора. Вона має один атрибут gesture_code, який ідентифікує тип вхідного сигналу. Цей код уніфікований згідно з протоколом мікроконтролера і є основою для подальшого відображення на дію.

Сутність Action описує дію, яку програма повинна виконати у відповідь на отриманий жест. Основним атрибутом є key_combination, який містить опис комбінації клавіш, що буде емітовано в операційній системі. Наприклад, це може бути Ctrl+C, Alt+F4 або будь-яка інша клавіатурна послідовність.

Сутність ConfigFile є контейнером для набору жестів та відповідних їм дій. Вона описує конкретну конфігурацію, яка застосовується у застосунку. Один ConfigFile може містити декілька жестів та дій, утворюючи пари. Така реалізація дозволяє створювати індивідуальні профілі керування.

Візуальна схема зв’язків між сутностями представлена на рисунку 2.7.

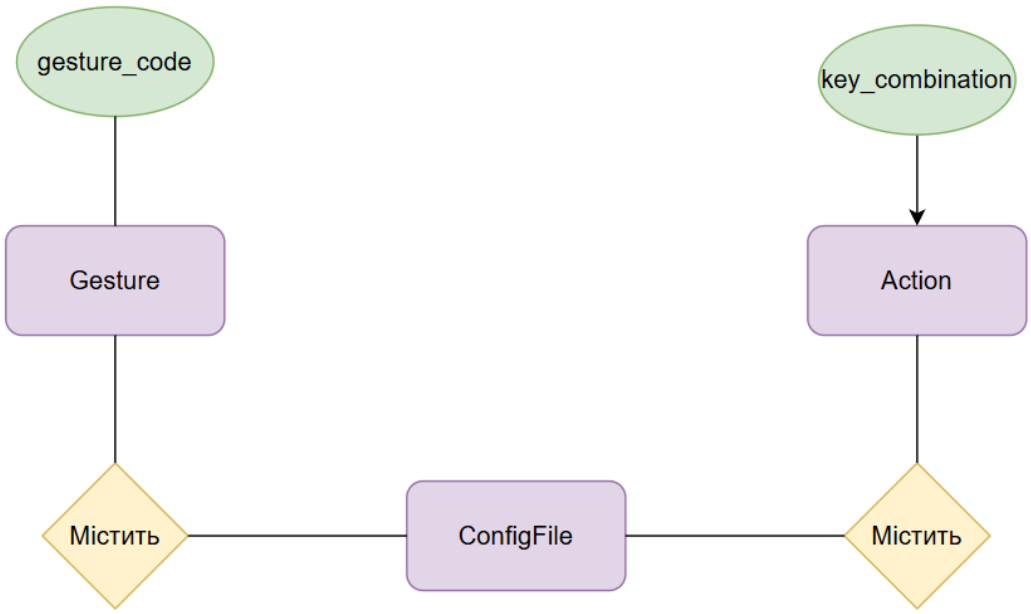

Рисунок 2.7 – ER-діаграма

У рамках цієї моделі не встановлюється прямий зв’язок між Gesture та Action, оскільки їх зв’язок опосередкований сутністю ConfigFile, що служить як джерело структурованих відповідностей. Це спрощує зміну конфігурації – для завантаження нового профілю достатньо підключити інший конфігураційний файл без перезапуску програми або перезапису коду.

Це дозволяє підтримувати пряме відображення між позиціями списку жестів та списку дій. Порядковий номер у списку використовується для встановлення асоціації між подією та дією без необхідності створення додаткової таблиці зв’язків або структур даних.

Файл конфігурації у застосунку використовується для збереження відповідностей між розпізнаними жестами та комбінаціями клавіш, які повинні бути згенеровані у відповідь на ці жести. Формат зберігання – текстовий, структура представлена у вигляді об’єкта JSON. Це дозволяє легко обробляти, редагувати, зберігати та завантажувати налаштування за допомогою стандартних засобів мови програмування.

Кожен запис у файлі містить пару «ключ–значення», де ключем є назва жесту, а значенням – рядок, що описує комбінацію клавіш у форматі, який підтримується бібліотекою емуляції натискання клавіш на комп’ютері.

У табл. 2.1 подано перелік жестів, які підтримуються системою, та комбінації клавіш, що асоційовані з ними за замовчуванням

Таблиця 2.1 – Жести системи

Жест	**Код у файлі**	**Комбінація клавіш (за замовчуванням)**
Проведення вліво	«Left»	Ctrl+Left
Проведення вправо	«Right»	Ctrl+Right
Проведення вгору	«Up»	Ctrl+Up
Проведення вниз	«Down»	Ctrl+Down
Рух вперед	«Forward»	Ctrl+F
Рух назад	«Backward»	Ctrl+B

Жест	Код у файлі	Комбінація клавіш (за замовчуванням)
Оберт вправо	«RotateRight»	Ctrl+R
Оберт вліво	«RotateLeft»	Ctrl+L

Кожен запис у файлі містить пару «ключ–значення», де ключем є назва жесту, а значенням – рядок, що описує комбінацію клавіш у форматі, який підтримується бібліотекою емуляції натискання клавіш на комп’ютері.

2.4 Проєктування 3D-моделі корпусу

Сучасний розвиток технологій CAD-проєктування відкрив нові можливості для інженерів та дизайнерів, особливо з появою хмарних платформ для тривимірного моделювання. Серед численних рішень особливе місце займає онлайн-сервіс Onshape, який був обраний для проєктування корпусу системи розпізнавання жестів з огляду на його унікальні переваги та функціональні можливості.

Вибір Onshape як основного інструменту проєктування був зумовлений кількома ключовими факторами. По-перше, хмарна архітектура сервісу забезпечує доступ до проєкту з будь-якого пристрою, що має підключення до інтернету, без необхідності встановлення громіздкого програмного забезпечення. Це особливо важливо в умовах сучасного мобільного робочого процесу, коли розробка може вестися з різних локацій та на різних платформах.

По-друге, Onshape пропонує професійний рівень функціональності, який раніше був доступний лише в дорогих desktop-рішеннях. Параметричне моделювання, розширені можливості створення складних геометричних форм та інтуїтивний інтерфейс роблять цю платформу ідеальним вибором для проєктування точних технічних деталей.

Проєктування корпусу розпочалося з концептуального підходу, що передбачав створення двокомпонентної конструкції. Така архітектурна концепція була обрана не випадково – розділення на основу та кришку забезпечує

максимальну функціональність при мінімальній складності виготовлення. Основа призначалася для розміщення ключового електронного компонента – плати Arduino Pro Micro, тоді як кришка мала інтегрувати датчик PAJ7620 та забезпечувати захист внутрішніх елементів.

Особливу увагу під час проєктування було приділено ергономіці та функціональності. Основа корпусу була ретельно спроєктована з урахуванням габаритних розмірів Arduino Pro Micro, при цьому забезпечувався зручний доступ до USB-порту для програмування та живлення пристрою. Це потребувало точних розрахунків та врахування товщини стінок корпусу, щоб гарантувати надійне кріплення плати без перешкоджання доступу до інтерфейсів.

Проєктування кришки потребувало створення спеціалізованого отвору для датчика PAJ7620. Геометрія цього отвору мала забезпечувати точне позиціонування сенсора відносно зовнішнього середовища, гарантуючи оптимальний кут огляду та мінімізуючи можливість потрапляння сторонніх об'єктів всередину корпусу.

Використання Onshape дозволило реалізувати складні геометричні рішення з високою точністю. Параметричне моделювання забезпечило можливість швидкого внесення змін у конструкцію без необхідності повного перепроєктування, що особливо важливо на етапі прототипування. Візуалізаційні можливості платформи дозволили оцінити естетичні аспекти дизайну та забезпечити гармонійний вигляд готового виробу.

Результатом проєктування стала 3D-модель, що повністю відповідала технічним вимогам проєкту. Двокомпонентна конструкція забезпечила не лише функціональність, але й технологічність виготовлення – обидві частини корпусу могли бути надруковані без підтримуючих структур, що спростило процес 3D-друку та підвищило якість поверхонь.

Досвід використання Onshape для проєктування корпусу системи розпізнавання жестів продемонстрував ефективність сучасних хмарних CAD-рішень. Поєднання професійного функціоналу, доступності та зручності використання робить цю платформу оптимальним вибором для розробки складних

технічних виробів, де критично важливими є точність, швидкість розробки та можливість спільної роботи.

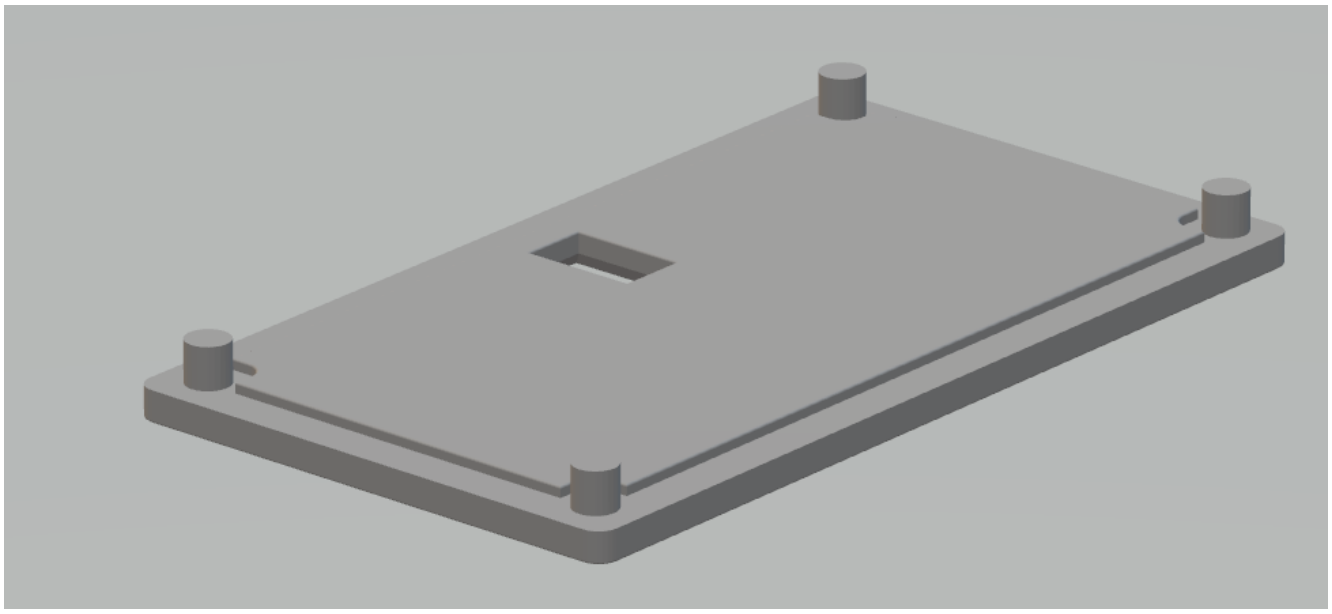

Рисунок 2.8 – 3D-модель кришки

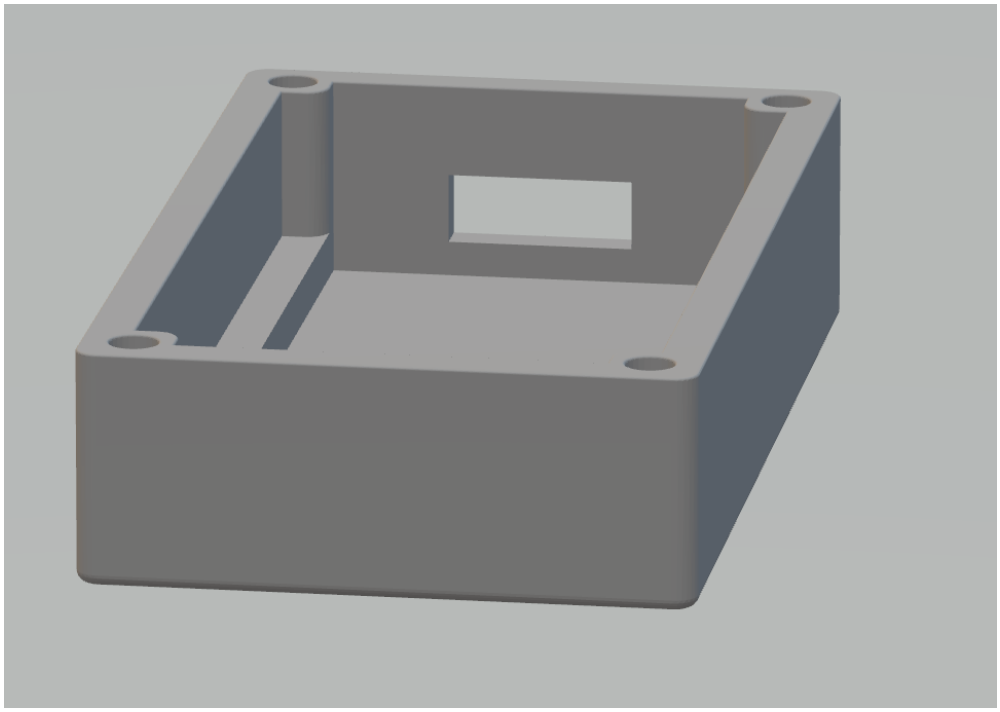

Рисунок 2.9 – 3D-модель корпусу

Для реалізації проєкту був використаний професійний слайсер Bambu Studio, що дозволив оптимально підготувати модель до друку. Програмне забезпечення автоматично розрахувало оптимальне розташування деталей на робочому столі.

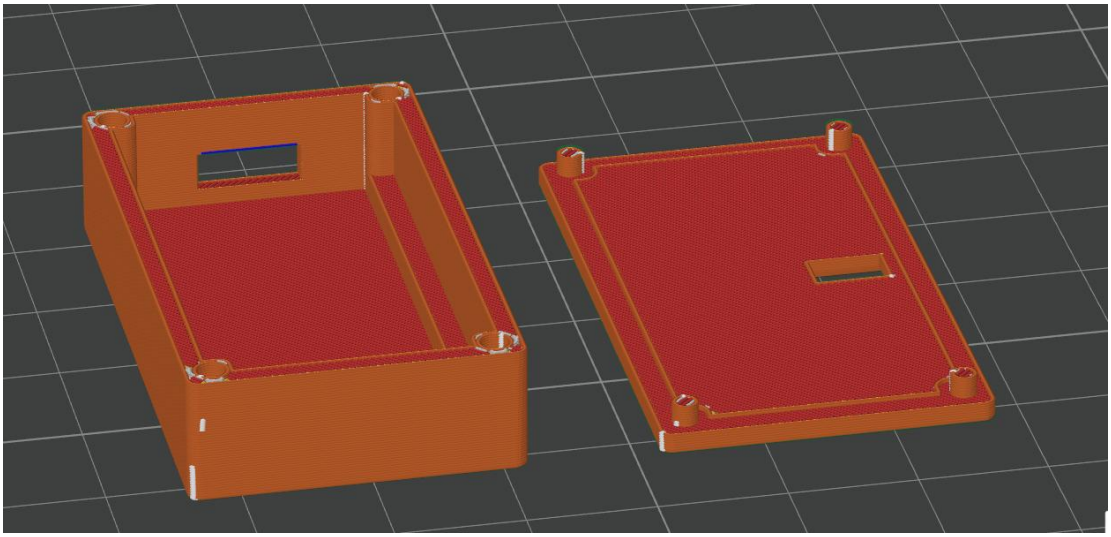

Рисунок 2.10 – Корус нарізаний в слайсері

Друк корпусу здійснювався на високопродуктивному 3D-принтері Bambu Lab X1 Carbon, який забезпечив виняткову точність та якість виготовлення. Принтер продемонстрував стабільну роботу протягом всього процесу друку, автоматично контролюючи температурні режими та швидкість подачі матеріалу.

Рисунок 2.11 – Принтер Bambu Lab X1 Carbon

При виборі матеріалу для виготовлення компонентів системи розпізнавання жестів постало питання про оптимальний тип пластику для 3D-друку. Серед широкого спектру доступних матеріалів найбільш поширеними є PLA, ABS, PETG, кожен з яких має свої унікальні характеристики та області застосування.

PLA традиційно вважається найпростішим матеріалом для початківців у 3D-друці. Він легко друкується при відносно низьких температурах (190-220°C), має мінімальну усадку. Однак його основним недоліком є крихкість та низька термостійкість – деталі з PLA можуть деформуватися вже при температурі 60°C.

ABS демонструє значно кращі механічні властивості порівняно з PLA. Він більш міцний, стійкий до ударів та витримує температури до 100°C. Проте ABS має істотні недоліки: схильність до значної усадки під час охолодження, що призводить до деформації та розтріскування великих деталей, необхідність використання закритого корпусу принтера та підігрітого столу, а також виділення шкідливих парів під час друку. Ці фактори ускладнюють процес виготовлення та потребують додаткового обладнання.

У цьому контексті PETG постає як оптимальне рішення, що поєднує переваги інших матеріалів, мінімізуючи їхні недоліки. PETG успадковує міцність ABS, зберігаючи при цьому простоту використання PLA. Цей матеріал практично не дає усадки під час охолодження, що критично важливо для виготовлення точних деталей складної геометрії.

Нетоксичність матеріалу додатково підтверджує його безпечність для використання у приміщеннях. Температурний режим друку PETG (220-250°C для екструдера та 60-85°C для столу) є компромісним між простотою PLA та вимогливістю ABS. Практичні результати підтвердили правильність вибору – час друку моделі склав лише 18 хвилин, а якість отриманих деталей повністю відповідала технічним вимогам проєкту. Відмінна адгезія між шарами забезпечила міцність конструкції, а мінімальна схильність до деформацій гарантувала точність геометричних розмірів.

Таким чином, вибір PETG для виготовлення компонентів системи розпізнавання жестів був обґрунтованим рішенням, що забезпечило оптимальний

баланс між якістю, простотою виготовлення та експлуатаційними характеристиками готового виробу. Цей матеріал дозволив реалізувати технічні вимоги проєкту при мінімальних затратах часу та ресурсів.

Готовий корпус продемонстрував ефективність спроєктованого внутрішнього простору. Arduino Pro Micro розмістилася в спеціально передбаченому відсіку основи, з легким доступом до USB-роз'єму через бічний отвір.

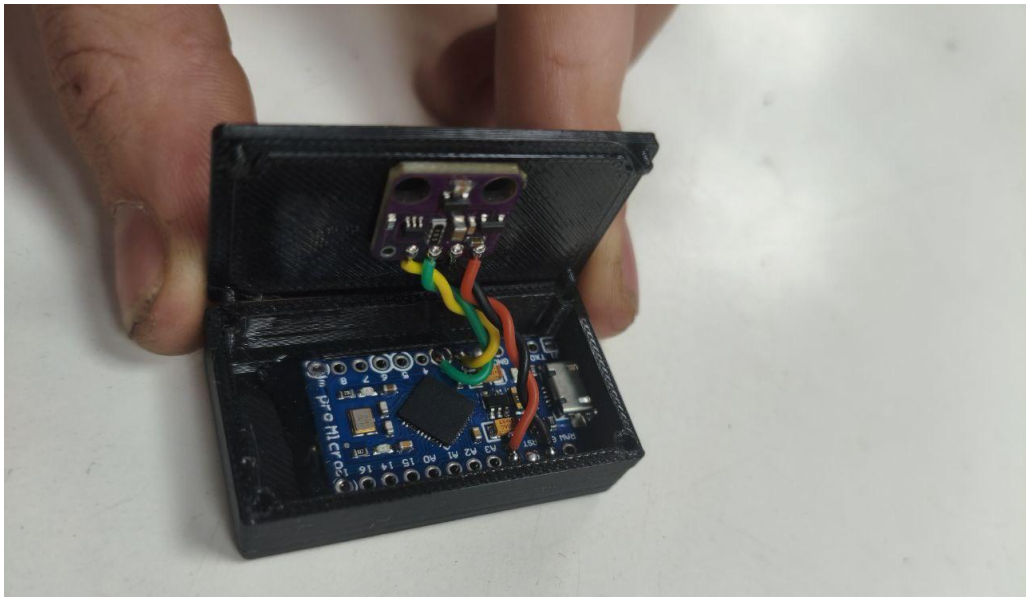

Рисунок 2.12 – Вигляд розпаяних компонентів

Датчик PAJ7620 був інтегрований у верхню частину кришки, забезпечуючи оптимальну зону детекції жестів. Компактні розміри та продумана форма корпусу забезпечують стабільне положення на робочій поверхні.

Рисунок 2.13 – Готовий пристрій

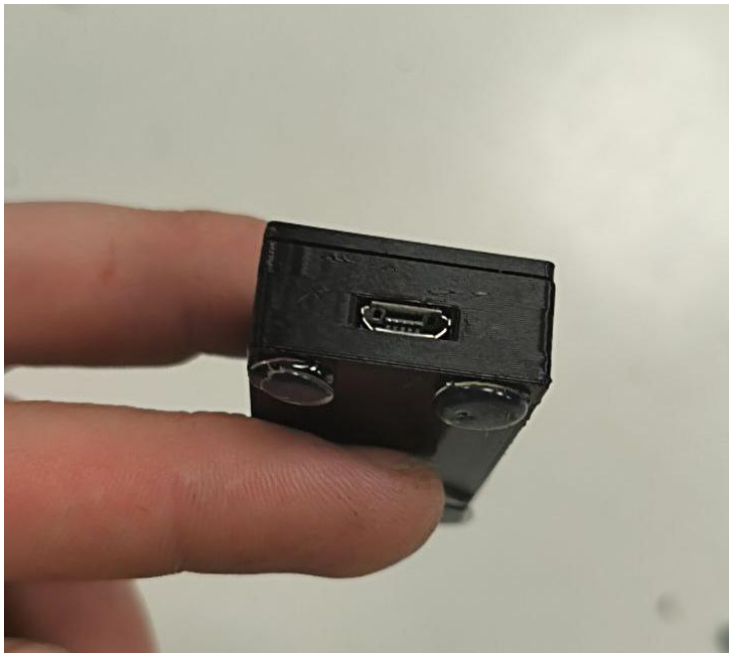

Рисунок 2.14 – Вигляд з боку USB

Для забезпечення кращої стійкості та запобігання ковзанню пристрою по робочій поверхні було прийнято рішення додати краплі термоклею по кутах основи корпусу. Це технічне рішення виявилося простим та ефективним способом створення антиковзного покриття без необхідності використання спеціалізованих матеріалів.

Термоклей був обраний завдяки його доступності, простоті нанесення та відмінним адгезійним властивостям до поверхні PETG пластику. Краплі термоклею, нанесені по кутах основи, створюють додаткові точки опори з підвищеним коефіцієнтом тертя, що значно покращує стабільність положення пристрою під час використання.

Така модифікація особливо важлива для системи розпізнавання жестів, оскільки будь-яке переміщення корпусу може вплинути на точність детекції рухів користувача. Антиковзні елементи забезпечують надійну фіксацію пристрою на столі.

Завершений проєкт 3D-друку корпусу мультимедійної клавіатури став успішним прикладом інтеграції сучасних технологій проєктування та виробництва, створивши функціональний та естетично привабливий пристрій для управління мультимедійними додатками.

Висновки до розділу 2

Другий розділ дослідження присвячено детальному розгляду технічної реалізації системи безконтактного керування мультимедійними функціями за допомогою жестів. Проведений аналіз демонструє комплексний підхід до розробки, що охоплює як апаратну, так і програмну складові системи.

Розроблений алгоритм функціонування системи розпізнавання жестів представляє собою добре структурований процес, що починається з ретельної ініціалізації всіх компонентів та переходить до безперервного циклу обробки жестових команд. Особливу увагу приділено оптимізації взаємодії між мікроконтролером Arduino Pro Micro та датчиком PAJ7620, де встановлено оптимальний інтервал опитування в 100 мс, що забезпечує баланс між чутливістю системи та стійкістю до помилкових спрацьовувань.

Аналіз схеми підключення показав доцільність використання інтерфейсу I²C для комунікації між компонентами, що дозволяє мінімізувати кількість з'єднань та забезпечує надійну передачу даних. Вибір Arduino Pro Micro як основного мікроконтролера виправданий його вбудованою підтримкою USB-інтерфейсу та можливістю емуляції HID-пристроїв, що є критично важливим для взаємодії з операційною системою комп'ютера.

Функціональна структура десктопного застосунку відображає продуманий підхід до створення користувацького інтерфейсу. Реалізація фонового режиму роботи забезпечує постійну готовність системи до прийому команд, а використання JSON-формату для зберігання конфігурацій дозволяє легко налаштовувати відповідності між жестами та системними командами.

Діаграма варіантів використання чітко визначає основні функціональні можливості системи: додавання нових комбінацій клавіш, редагування існуючих асоціацій та відновлення стандартних налаштувань. Така структура забезпечує гнучкість у налаштуванні системи під індивідуальні потреби користувача.

Проєктування та виготовлення 3D-корпусу стало важливим етапом створення завершеної системи. Розроблена двокомпонентна конструкція з основи

та кришки забезпечує не лише захист електронних компонентів, але й оптимальні умови для роботи датчика жестів. Внутрішній простір корпусу організовано таким чином, що Arduino Pro Micro та датчик PAJ7620 надійно зафіксовані у відповідних відсіках з легким доступом для обслуговування. Зовнішній дизайн корпусу поєднує функціональність та естетичність, створюючи завершений продукт, що гармонійно вписується в сучасне робоче середовище.

Представлена система восьми жестів із відповідними комбінаціями клавіш за замовчуванням демонструє практичність розробленого рішення. Кожен жест має логічне призначення, що інтуїтивно зрозуміле користувачеві та ефективно використовує можливості датчика.

Загалом, технічне рішення, представлене в другому розділі, характеризується оптимальним балансом між складністю реалізації та функціональністю системи. Використання стандартних протоколів комунікації, відкритих апаратних платформ та сучасних підходів до розробки програмного забезпечення робить систему доступною для широкого кола користувачів та придатною для подальшого розвитку й удосконалення.

3 РЕАЛІЗАЦІЯ ПРОТОТИПУ МУЛЬТИМЕДІЙНОЇ КЛАВІАТУРИ

3.1 Написання програми для першої взаємодії з сенсором PAJ7620

Програма починається з підключення необхідних бібліотек. Основна бібліотека DFRobot_PAJ7620U2.h забезпечує взаємодію з датчиком жестів PAJ7620U2, який здатний розпізнавати рухи руки в просторі.

GestureRecognize_Keyboard.ino

```
#include "DFRobot_PAJ7620U2.h"
#include <Wire.h>

DFRobot_PAJ7620U2 paj;

void setup()
{
  Serial.begin(115200);
  delay(300);
  Serial.println("Gesture recognition system started");

  while(paj.begin() != 0){
    Serial.println("PAJ7620U2 initialization failed!");
    delay(500);
  }

  Serial.println("PAJ7620U2 ready");
  paj.setGestureHighRate(false);

}

void loop()
{
  DFRobot_PAJ7620U2::eGesture_t gesture = paj.getGesture();

  if(gesture != paj.eGestureNone ){
    String description = paj.gestureDescription(gesture);
    String data = "";


    if(description == "Left") data = "Left";
    else if(description == "Right") data = "Right";
    else if(description == "Up") data = "Up";
    else if(description == "Down") data = "Down";
    else if(description == "Forward") data = "Forward";
    else if(description == "Backward") data = "Backward";
    else if(description == "Clockwise") data = "RotateRight";
    else if(description == "Anti-Clockwise") data = "RotateLeft";


    if(data != "") {
      Serial.println(data);
    }

    delay(200);
  }
}
```

Рисунок 3.1 – Програмний код

Додатково підключається стандартна бібліотека Wire.h для роботи з I2C протоколом, через який відбувається комунікація з датчиком. Створюється глобальний об'єкт paj класу DFRobot_PAJ7620U2, який слугуватиме інтерфейсом для взаємодії з апаратним датчиком.

Функція *setup()* відповідає за початкову конфігурацію системи. Спочатку ініціалізується серійне з'єднання зі швидкістю 115200 для виведення інформації в монітор порту. Після короткої затримки система повідомляє про початок роботи системи розпізнавання жестів.

Ключовою частиною ініціалізації є цикл, який намагається підключитися до датчика PAJ7620U2. Якщо підключення не вдається, система виводить повідомлення про помилку і чекає 500 мілісекунд перед наступною спробою. Цей механізм забезпечує надійність запуску системи навіть у випадку тимчасових проблем з підключенням датчика.

Після успішного підключення система налаштовує датчик через виклик *setGestureHighRate(false)*, що встановлює нормальну швидкість розпізнавання жестів замість високої частоти. Це рішення забезпечує баланс між чутливістю та стабільністю роботи.

Функція *loop()* містить основну логіку програми, яка виконується безперервно. На початку кожної ітерації система отримує інформацію про поточний жест через виклик *paj.getGesture()*, який повертає значення типу eGesture_t.

Якщо виявлено будь-який жест (відмінний від eGestureNone), система переходить до його обробки. Спочатку отримується текстовий опис жесту через метод *gestureDescription()*, після чого запускається процес перетворення опису в стандартизований формат даних.

Центральною частиною програми є блок умовних операторів, який перетворює описи жестів у стандартизовані команди. Система розпізнає вісім основних типів жестів: рух ліворуч та праворуч, рух вгору та вниз, рух вперед та назад, а також обертальні рухи за годинниковою стрілкою та проти неї.

Після успішної ідентифікації та стандартизації жесту система виводить результат через серійний порт. Важливою деталлю є перевірка, чи не є рядок data порожнім перед виведенням, що запобігає виведенню непотрібної інформації.

Розглянутий програмний код демонструє практичне застосування сенсора PAJ7620 у поєднанні з платформою Arduino для розпізнавання жестів. За допомогою небагатьох десятків рядків коду реалізується виявлення 8 типів рухів. Тепер можна перейти до тестування роботи датчика у різних умовах експлуатації.

3.2 Тестування ефективності розпізнавання

Методика тестувального дослідження була обрана відповідно до робочих характеристик датчика жестів PAJ7620 було розроблено комплексну методику тестування для визначення точності розпізнавання жестів системою безконтактного управління в різних умовах експлуатації. Головною метою тустування було встановлення залежності точності розпізнавання жестів від відстані між рукою користувача та датчиком, а також визначення оптимального робочого діапазону системи.

Для проведення тестування було використано прототип системи безконтактного управління, що складався з датчика PAJ7620, підключеного до мікроконтролера Arduino Pro Micro згідно з розробленою схемою підключення.

Тестування проводилось у приміщенні з контрольованим рівнем освітлення (500–600 люкс), що відповідає типовим умовам офісного приміщення. Для підвищення достовірності результатів тестування проводився трьома різними людьми, кожен з яких мав різний досвід взаємодії з системами жестового керування.

У ході тестування кожен з восьми базових жестів (рух вгору, вниз, вліво, вправо, вперед, назад, по годинниковій та проти годинникової стрілки) виконувався по 50 разів на кожному інтервалі відстані від датчика. Інтервали відстані були обрані з кроком 5 см, починаючи від 5 см і закінчуючи 30 см від поверхні датчика.

Для кожного виконаного жесту фіксувались наступні параметри:

– тип жесту, що виконувався;
– відстань від руки до датчика;
– результат розпізнавання (правильно/неправильно);
– час розпізнавання (мс);
– рівень достовірності розпізнавання (за даними датчика).

Результати тестування точності розпізнавання жестів системою безконтактного управління наведені в табл. 3.1, де представлена залежність середньої точності розпізнавання від відстані між рукою оператора та датчиком для кожного з восьми базових жестів.

Таблиця 3.1 – Результати тестування точності розпізнавання жестів (%)

Відстань, см	**Вгору, %**	**Вниз, %**	**Вліво, %**	**Вправо, %**	**Вперед, %**	**Назад, %**	**По год., %**	**Проти год., %**	**Середня, %**
5	86	84	100	100	100	100	98	100	**96**
10	81	83	100	100	100	100	98	97	**94,8**
15	71	74	100	100	98	99	95	96	**91,6**
20	66	63	88	89	87	88	75	71	**78,4**
25	55	61	74	77	74	75	67	68	**68,8**
30	4	2	3	4	2	2	1	0	**2,8**

З отриманих результатів можна зробити висновок, що точність розпізнавання жестів системою суттєво залежить від відстані між рукою оператора та датчиком. Найвища точність розпізнавання (близько 91–96 %) досягається на відстані 5–15 см, що робить цей діапазон оптимальним для роботи системи.

При збільшенні відстані до 20–25 см спостерігається поступове зниження точності розпізнавання до 65–80 %. Подальше збільшення відстані до 30 см призводить до втрати системою здатність коректно розпізнавати жести (середня точність 2–3 %).

Додатково був проведений аналіз впливу швидкості виконання жестів на точність їх розпізнавання. Результати цього аналізу представлені в табл. 3.2, де

наведені середні значення точності розпізнавання для різних швидкостей виконання жестів на відстані 10 см від датчика.

Таблиця 3.2 – Вплив швидкості жестів на точність розпізнавання (%)

Швидкість жесту	**Вгору, %**	**Вниз, %**	**Вліво, %**	**Вправо, %**	**Вперед, %**	**Назад, %**	**По год.,%**	**Проти год.,%**	**Середня, %**
Повільна	86	84	100	100	100	100	100	100	**96**
Нормальна	81	83	100	100	100	100	99	98	**95**
Швидка	82	80	78	76	70	68	60	56	**71,3**

З наведених даних видно, що швидкість виконання жестів також суттєво впливає на точність їх розпізнавання. Найкращі результати досягаються при повільному та нормальному темпі виконання жестів (95–96 % точності), тоді як швидке виконання жестів призводить до значного падіння точності розпізнавання (до 71,3 %).

Окрім точності розпізнавання, важливим параметром системи безконтактного управління є латентність розпізнавання жестів, що визначає час відгуку системи на дії користувача. В ході тестувань було проведено вимірювання часу розпізнавання різних жестів системою на різних відстанях від датчика.

Таблиця 3.3 – Середній час розпізнавання жестів (мс)

Відстань, см	**Середній час розпізнавання, мс**	**Стандартне відхилення, мс**
5	234	12
10	252	14
15	246	18
20	302	40
25	343	44

На основі результатів тестування можна зробити наступні практичні висновки та рекомендації щодо використання системи безконтактного управління:

– оптимальний робочий діапазон: Система демонструє найкращі результати в діапазоні відстаней 5–15 см від датчика, де точність розпізнавання жестів близька до 100 %. При проєктуванні інтерфейсу системи рекомендується

організувати взаємодію з користувачем таким чином, щоб жести виконувались саме в цьому діапазоні;

– максимальна дальність дії: Максимальна дальність дії системи становить близько 25 см, однак на відстанях більше 15 см точність розпізнавання значно падає. При необхідності роботи на таких відстанях рекомендується використовувати лише простіші жести (рух вгору, вниз, вліво, вправо), які розпізнаються з вищою точністю порівняно зі складнішими жестами;

– швидкість виконання жестів: Для забезпечення високої точності розпізнавання користувачам рекомендується виконувати жести в повільному або нормальному темпі. Швидкі жести часто розпізнаються некоректно, особливо на більших відстанях від датчика.

Проведене тестування підтверджує, що розроблена система безконтактного управління відповідає заявленим вимогам щодо точності розпізнавання жестів (не менше 90 % в нормальних умовах освітлення) та латентності (не більше 500 мс), за умови її використання в оптимальному робочому діапазоні відстаней (5–15 см).

3.3 Розробка десктопної програми з графічним інтерфейсом користувача

Для розробки графічного інтерфейсу користувача персонального комп’ютера було використано інструмент Qt Designer. Цей засіб дозволяє швидко створювати візуальні компоненти та компонувати їх у віконні форми без необхідності ручного кодування.

Спочатку було спроєктовано основну форму інтерфейсу, яка включає елементи керування для налаштування жестів та відповідних їм дій. Вигляд розробленої форми, створеної в Qt Designer, наведено на рисунку 3.2.

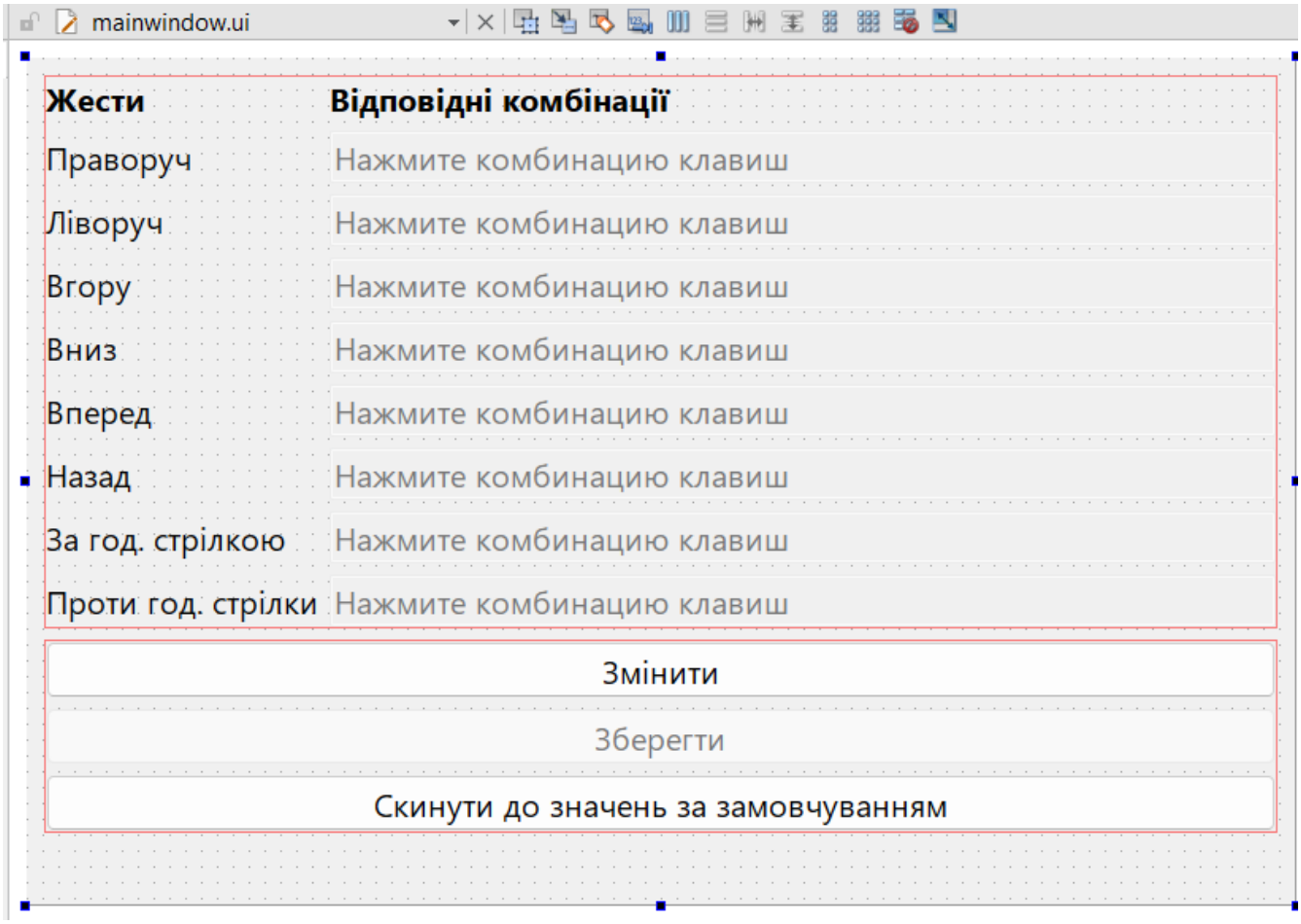

Рисунок 3.2 – Інтерфейс користувача програмного забезпечення для налаштування жестів

У розробці графічного інтерфейсу користувача було застосовано елементи Label для підпису назв жестів, Sequence – для полів вводу комбінацій клавіш, а також кнопки для взаємодії користувача з програмою. Для розміщення елементів на формі використано сітковий менеджер розташування (grid layout). Кожному елементу призначено зрозумілі ідентифікатори, що полегшує подальшу роботу з інтерфейсом. Структура цих елементів у вигляді ієрархічного дерева представлена на рисунку 3.3.

Об'єкт	Клас
MainWindow	QMainWindow
centralwidget	QWidget
formLayout	QFormLayout
keySequenceEditBackward	QKeySequenceEdit
keySequenceEditDown	QKeySequenceEdit
keySequenceEditForward	QKeySequenceEdit
keySequenceEditLeft	QKeySequenceEdit
keySequenceEditRight	QKeySequenceEdit
keySequenceEditRotateLeft	QKeySequenceEdit
keySequenceEditRotateRight	QKeySequenceEdit
keySequenceEditUp	QKeySequenceEdit
label	QLabel
label_10	QLabel
label_11	QLabel
label_2	QLabel
label_3	QLabel
label_4	QLabel
label_5	QLabel
label_6	QLabel
label_7	QLabel
label_8	QLabel
statusLabel	QLabel
verticalLayout_5	QVBoxLayout
pushButtonDefault	QPushButton
pushButtonEdit	QPushButton
pushButtonSave	QPushButton

Рисунок 3.3 – Структура елементів інтерфейсу користувача у вигляді дерева

Перед реалізацією основної частини програми наведено фрагмент файлу збірки CMakeLists.txt, який містить налаштування підключення необхідних бібліотек Qt.

```
find_package(QT NAMES Qt6 Qt5 REQUIRED COMPONENTS Widgets SerialPort LinguistTools)
find_package(Qt${QT_VERSION_MAJOR} REQUIRED COMPONENTS Widgets SerialPort LinguistTools)

set(TS_FILES GestureControl_uk_UA.ts)

set(PROJECT_SOURCES
        main.cpp
        mainwindow.cpp
        mainwindow.h
        mainwindow.ui
        ${TS_FILES}
)

if(${QT_VERSION_MAJOR} GREATER_EQUAL 6)
    qt_add_executable(GestureControl
        MANUAL_FINALIZATION
        ${PROJECT_SOURCES}
        resources.qrc
    )
# Define target properties for Android with Qt 6 as:
#    set_property(TARGET GestureControl APPEND PROPERTY QT_ANDROID_PACKAGE_SOURCE_DIR
#                 ${CMAKE_CURRENT_SOURCE_DIR}/android)
# For more information, see https://doc.qt.io/qt-6/qt-add-executable.html#target-creation

    qt_create_translation(QM_FILES ${CMAKE_SOURCE_DIR} ${TS_FILES})
else()
    if(ANDROID)
        add_library(GestureControl SHARED
            ${PROJECT_SOURCES}
        )
# Define properties for Android with Qt 5 after find_package() calls as:
#    set(ANDROID_PACKAGE_SOURCE_DIR "${CMAKE_CURRENT_SOURCE_DIR}/android")
    else()
        add_executable(GestureControl
            ${PROJECT_SOURCES}
        )
    endif()

    qt5_create_translation(QM_FILES ${CMAKE_SOURCE_DIR} ${TS_FILES})
endif()

target_link_libraries(GestureControl PRIVATE Qt${QT_VERSION_MAJOR}::Widgets Qt${QT_VERSION_MAJOR}::SerialPort)
```

Рисунок 3.4 – Фрагмент файлу CMakeLists для підключення бібліотек Qt

Ці команди забезпечують налаштування залежностей проєкту відповідно до встановленої версії Qt. Таким чином, файл збірки містить усі необхідні інструкції для коректного підключення бібліотек, що використовуються в програмі. Цей фрагмент стосується лише конфігурації збірки і не описує функціонал програми.

Програма складається з одного класу MainWindow, що наслідується від QMainWindow. Цей клас відповідає за основний графічний інтерфейс користувача, а також за взаємодію з послідовним портом.

У заголовочному файлі MainWindow.h визначені конструктор і деструктор класу, а також приватні слоти, які обробляють події інтерфейсу користувача та надходження даних із послідовного порту. Зокрема, передбачено обробку натискань кнопок редагування, збереження та скидання налаштувань.

Окрім слотів, у приватній секції класу оголошено допоміжні методи для ініціалізації послідовного порту, завантаження конфігурації жестів, керування

режимом редагування та симуляції комбінацій клавіш. Для операційної системи Windows реалізовано перевизначення методу *nativeEvent()* для обробки системних подій.

Конструктор класу MainWindow виконує початкову ініціалізацію інтерфейсу користувача та налаштувань програми.

Деструктор класу MainWindow звільняє пам’ять, видаляючи об’єкт інтерфейсу користувача.

```
QSerialPort *serial = nullptr;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    loadGesturesConfig();
    setEditingEnabled(false);
    initSerial();
    QShortcut *altF4Shortcut = new QShortcut(QKeySequence("Alt+F4"), this);
    connect(altF4Shortcut, &QShortcut::activated, this, [this]() {
        qDebug() << "Alt+F4 натиснуто";
    });
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

Рисунок 3.5 – Конструктор і деструктор класу MainWindow з ініціалізацією

Спочатку створюється об’єкт інтерфейсу UI і викликається метод *setupUi()* для встановлення структури вікна згідно з описом у файлі UI, створеному за допомогою Qt Designer.

Далі викликається метод loadGesturesConfig, який відповідає за завантаження конфігурації жестів, що використовуються в програмі. Метод *setEditingEnabled()* встановлює режим редагування інтерфейсу, на початку він відключений, що означає, що користувач не може змінювати налаштування без відповідної активації.

Ініціалізується послідовний порт викликом *initSerial()*, який забезпечує зв’язок програми з мікроконтролером через серійний інтерфейс.

Для запобігання випадковому закриттю програми при введенні комбінації клавіш Alt+F4 створюється об’єкт QShortcut. Він прив’язаний до цієї комбінації і

підключає сигнал активації до лямбда-функції, яка виводить повідомлення в консоль. Це дозволяє програмі ігнорувати стандартну дію Alt+F4, якщо користувач вводить цю комбінацію для призначення її як жесту.

```
40  void MainWindow::initSerial()
41  {
42      qDebug() << "Доступні COM-порти:";
43
44      foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts()) {
45          qDebug() << "Порт:" << info.portName();
46          qDebug() << "Опис:" << info.description();
47          qDebug() << "Виробник:" << info.manufacturer();
48          qDebug() << "Серійний номер:" << info.serialNumber();
49          qDebug() << "Системний шлях:" << info.systemLocation();
50          qDebug() << "------------------------------------------------";
51      }
52
53      // Логіка пошуку Arduino або USB-плати
54      foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts()) {
55          if (info.manufacturer().contains("Arduino") || info.description().contains("USB")) {
56              serial = new QSerialPort(info, this);
57              serial->setBaudRate(QSerialPort::Baud115200);
58              serial->setDataBits(QSerialPort::Data8);
59              serial->setParity(QSerialPort::NoParity);
60              serial->setStopBits(QSerialPort::OneStop);
61              serial->setFlowControl(QSerialPort::NoFlowControl);
62              if (serial->open(QIODevice::ReadOnly)) {
63                  connect(serial, &QSerialPort::readyRead, this, &MainWindow::readSerialData);
64                  ui->statusLabel->setText("З'єднано: " + info.portName());
65                  return;
66              }
67          }
68      }
69
70      ui->statusLabel->setText("Плата не знайдена");
71  }
```

Рисунок 3.6 – Функція *InitSerial()*

Після початкової ініціалізації та налаштувань, основні функції класу MainWindow забезпечують взаємодію програми з апаратною частиною та користувачем. Функція *initSerial()* виконує пошук і підключення до послідовного порту, через який здійснюється прийом даних від мікроконтролера з датчиком жестів. В разі успішного підключення налаштовуються параметри порту (швидкість передачі, формат кадру) і встановлюється обробник подій наявності нових даних.

Даний метод відповідає за зчитування конфігурації жестів з файлу JSON. Якщо локальний файл відсутній, використовується конфігурація за замовчуванням із ресурсів програми. Дані розпарсуються і відображаються у відповідних полях інтерфейсу, що дозволяє користувачу ознайомитись із поточними налаштуваннями.

```
void MainWindow::loadGesturesConfig()
{
    QFile file("gestures.json");
    if (!file.exists()) {
        file.setFileName(":/gestures.json");
    }
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        qDebug() << "Не вдалося відкрити gestures.json";
        return;
    }
    QByteArray jsonData = file.readAll();
    file.close();
    QJsonDocument doc = QJsonDocument::fromJson(jsonData);
    if (doc.isNull() || !doc.isObject()) {
        qDebug() << "Невірний формат JSON";
        return;
    }
    QJsonObject obj = doc.object();

    if (obj.contains("Left")) {
        ui->keySequenceEditLeft->setKeySequence(QKeySequence(obj.value("Left").toString()));
    }
    if (obj.contains("Right")) {
        ui->keySequenceEditRight->setKeySequence(QKeySequence(obj.value("Right").toString()));
    }
    if (obj.contains("Up")) {
        ui->keySequenceEditUp->setKeySequence(QKeySequence(obj.value("Up").toString()));
    }
    if (obj.contains("Down")) {
        ui->keySequenceEditDown->setKeySequence(QKeySequence(obj.value("Down").toString()));
    }
    if (obj.contains("Forward")) {
        ui->keySequenceEditForward->setKeySequence(QKeySequence(obj.value("Forward").toString()));
    }
    if (obj.contains("Backward")) {
        ui->keySequenceEditBackward->setKeySequence(QKeySequence(obj.value("Backward").toString()));
    }
    if (obj.contains("RotateRight")) {
        ui->keySequenceEditRotateRight->setKeySequence(QKeySequence(obj.value("RotateRight").toString()));
    }
    if (obj.contains("RotateLeft")) {
        ui->keySequenceEditRotateLeft->setKeySequence(QKeySequence(obj.value("RotateLeft").toString()));
    }

}
```

Рисунок 3.7 – Метод *loadGesturesConfig()*

Режим редагування налаштувань активується викликом *setEditingEnabled()*, який змінює доступність полів введення і кнопок збереження та скасування змін. Це дозволяє уникнути випадкового редагування параметрів без відповідної дії користувача.

Обробка отриманих від мікроконтролера даних виконується у функції *readSerialData()*. Вона зчитує рядок із порту, порівнює отримане значення з відомими жестами і, за наявності збігу, запускає емуляцію натискання відповідної комбінації клавіш у системі.

```
void MainWindow::readSerialData()
{
    QByteArray data = serial->readAll();
    QString str = QString::fromUtf8(data).trimmed();
    qDebug() << "Отримано жест:" << str;

    QMap<QString, QKeySequenceEdit*> gestureMap = {
        {"Left", ui->keySequenceEditLeft},
        {"Right", ui->keySequenceEditRight},
        {"Up", ui->keySequenceEditUp},
        {"Down", ui->keySequenceEditDown},
        {"Forward", ui->keySequenceEditForward},
        {"Backward", ui->keySequenceEditBackward},
        {"RotateRight", ui->keySequenceEditRotateRight},
        {"RotateLeft", ui->keySequenceEditRotateLeft},
    };

    if (gestureMap.contains(str)) {
#ifdef Q_OS_WIN
        QKeySequence seq = gestureMap[str]->keySequence();
        simulateKeySequence(seq);
#endif
    }
}
```

Рисунок 3.8 – Функція *readSerialData*

Емуляція клавіатурних подій реалізована через функції, які формують і надсилають у систему низькорівневі події натискання та відпускання клавіш. Це дозволяє керувати іншими додатками, виконуючи дії, прив’язані до конкретних жестів. Таким чином, основні функції класу MainWindow формують базову логіку роботи програми, включаючи налаштування інтерфейсу, зв’язок із пристроєм, обробку даних і взаємодію із користувачем.

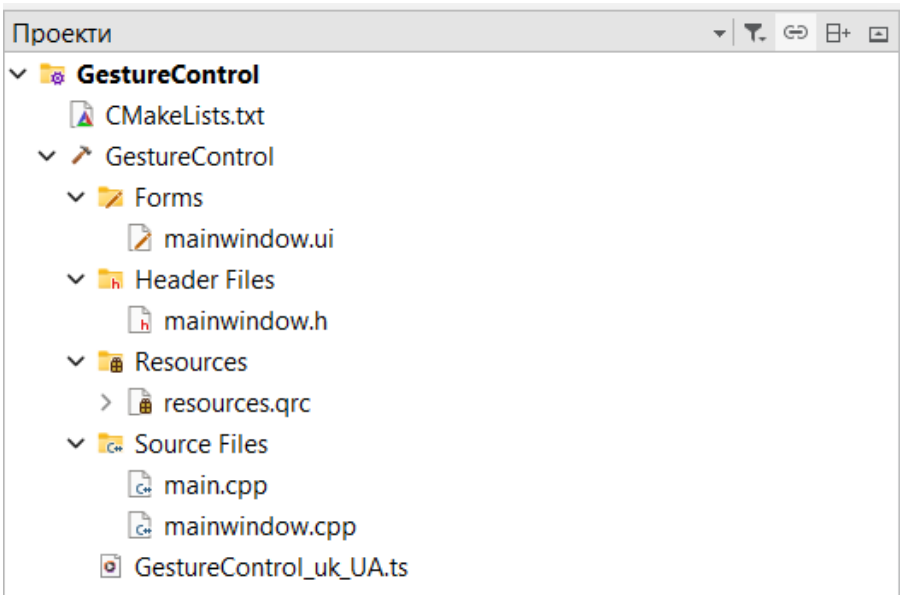

Рисунок 3.9 – Структура файлів проєкту

На рисунку 3.9 зображено загальну структуру проєкту, яка демонструє організацію файлів та їх взаємозв'язки. Проєкт структурований з урахуванням принципів модульності, де кожен компонент виконує чітко визначені функції: заголовкові файли містять декларації класів та функцій, файли реалізації - програмну логіку, а ресурсні файли забезпечують графічний інтерфейс користувача. Така архітектура сприяє зручності розробки, тестування та подальшого супроводу програмного забезпечення.

3.4 Тестування функціоналу системи та аналіз результатів

Тестування функціоналу системи проводиться нативним способом шляхом запуску програми та контролю її стабільності роботи під час взаємодії з апаратним забезпеченням. Метою тестування є перевірка коректності ініціалізації, зв’язку з мікроконтролером, обробки даних і відсутності аварійного завершення роботи програми.

Перевірка починається з підключення апаратної частини до персонального комп’ютера через USB-інтерфейс. Після фізичного підключення необхідно запустити програму. У вікні програми відображено повідомлення про успішне встановлення з’єднання з послідовним портом COM17.

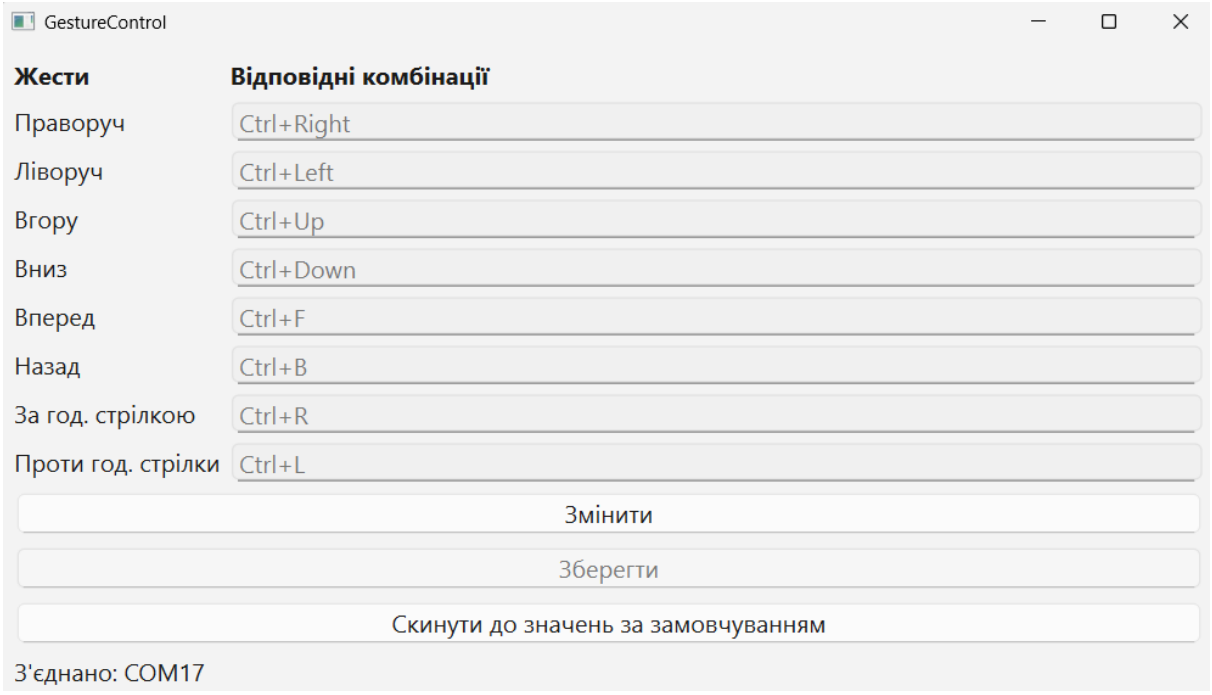

Рисунок 3.10 – Результат запуску програми

Це свідчить про правильну ініціалізацію серійного інтерфейсу і готовність програми до обміну даними з мікроконтролером.

Далі виконується перевірка функціоналу логування програми та надходження повідомлень від мікроконтролера. Логування реалізується у вигляді текстових повідомлень у спеціальному текстовому полі інтерфейсу. Результати логування відображено на рисунку 3.11.

```
------------------------------------------------
Порт: "COM10"
Опис: "USB-SERIAL CH340"
Виробник: "wch.cn"
Серійний номер: ""
Системний шлях: "\\\\.\\COM10"
------------------------------------------------
Отримано жест: "Right"
Отримано жест: "Left"
Отримано жест: "Up"
Отримано жест: "Down"
Отримано жест: "Up"
Отримано жест: "Down"
Отримано жест: "RotateRight"
Отримано жест: "Up"
Отримано жест: "Up"
Отримано жест: "RotateRight"
Отримано жест: "Up"
Отримано жест: "RotateRight"
Отримано жест: "RotateRight"
Отримано жест: "RotateLeft"
Отримано жест: "RotateLeft"
Отримано жест: "Down"
```

Рисунок 3.11 – Логування програми

Після запуску програми у лог записується перелік доступних COM-портів, що дає змогу переконатися у коректній роботі механізму виявлення серійних пристроїв.

Після встановлення з’єднання з Arduino Pro Micro програма починає отримувати дані з датчика жестів. У журналі фіксуються події, які сигналізують про отримання жестів.

Наступним етапом перевірки є тестування можливості зміни комбінації клавіш, що асоціюється з конкретним жестом. У межах цього тесту було змінено стандартні комбінації клаві на цифри від 1 до 8.

Цю зміну здійснено через графічний інтерфейс користувача у таблиці налаштувань. Після активації режиму редагування було обрано відповідні рядки та призначено їм нові комбінації натискання клавіш.

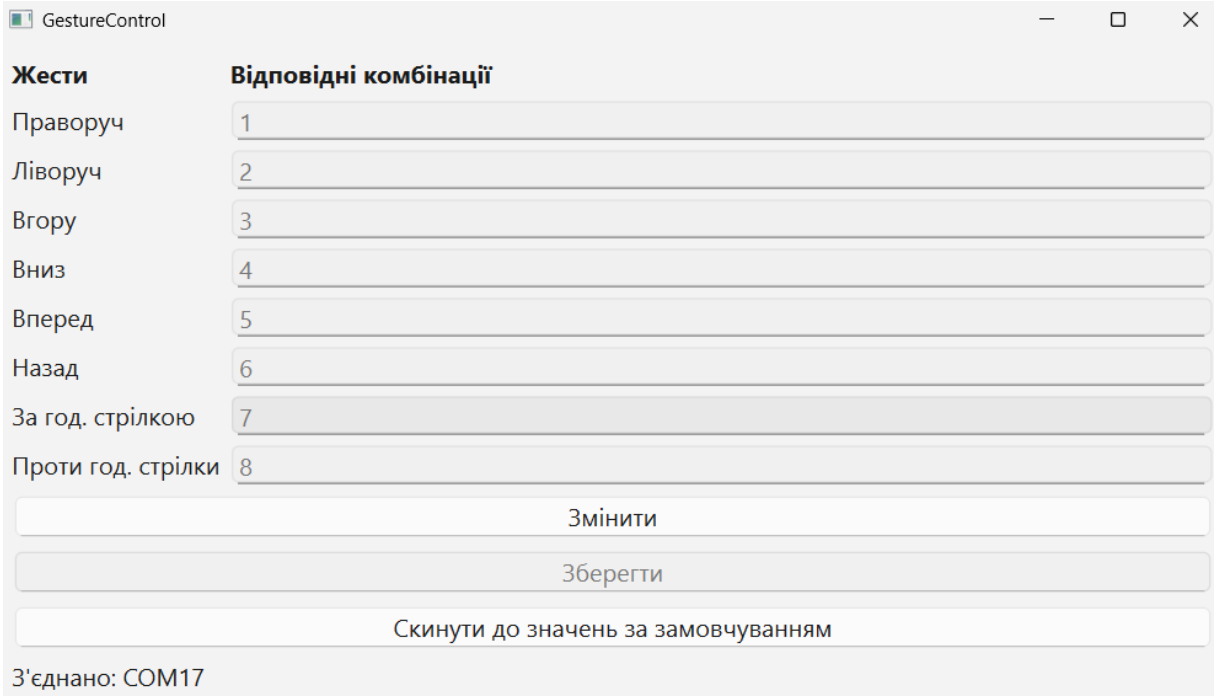

Рисунок 3.12 – Фрагмент інтерфейсу зі зміною комбінацій клавіш

Як видно, у відповідному полі таблиці тепер відображається значення цифр, що підтверджує успішне оновлення конфігурації в інтерфейсі. Це свідчить про працездатність механізму зміни команд, пов’язаних із жестами, а також про коректну роботу функцій збереження та відображення цих змін у UI.

Наступним етапом перевірки є перевірка відправки комбінацій клавіш до відповідних жестів.

На рис. 3.13 відображено результат даного тестування, де ми можемо побачити логування програми, та записаний у текстовий файл рядок «12345678», що свідчить що програма виконує закладені в неї алгоритми згідно вимог.

Рисунок 3.13 – Виконання жестів перед датчиком

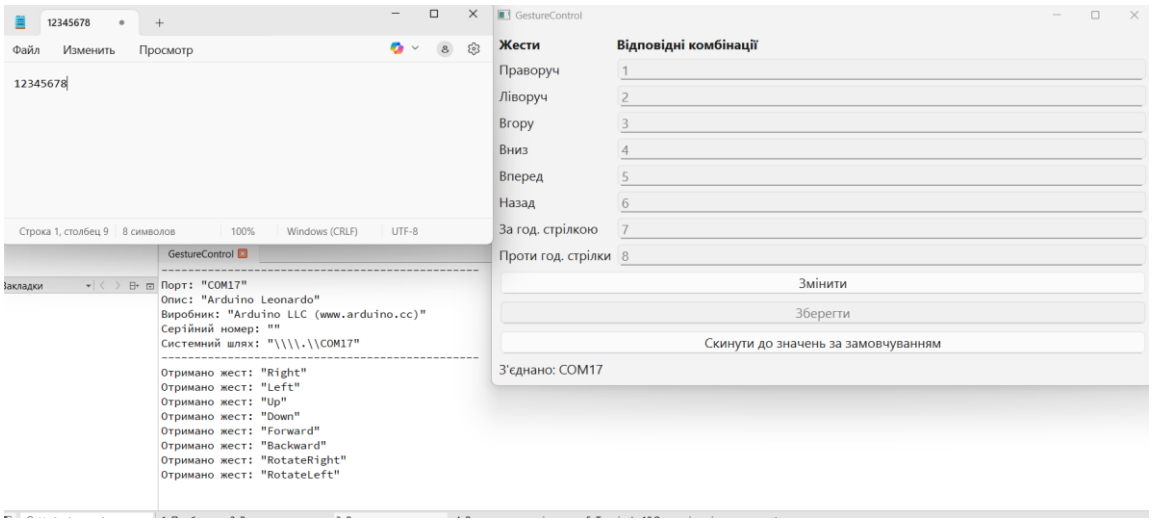

Рисунок 3.14 – Фрагмент інтерфейсу зі зміною комбінацій клавіш

Дане тестування продемонструвало повну функціональність системи розпізнавання жестів та виконання відповідних команд. Програма коректно інтерпретує налаштовані жести, обробляє їх згідно з визначеною конфігурацією та успішно виконує призначені дії. Логування процесу дозволяє відстежувати послідовність виконання операцій, що є важливим для діагностики та подальшої оптимізації системи.

Результати тестування підтверджують стабільність роботи всіх ключових компонентів: модуля розпізнавання жестів, системи обробки команд, механізму

збереження налаштувань та користувацького інтерфейсу. Це дає підстави стверджувати, що розроблена система готова до практичного використання та відповідає встановленим технічним вимогам.

Після цього для всіх інших жестів було встановлено відповідність: кожному жесту призначено першу літеру його назви. Це дозволяє у наступному етапі перевірки відстежити реакцію системи на різні жести й переконатися, що передані з плати дані інтерпретуються коректно згідно з налаштуваннями.

Висновки до розділу 3

Третій розділ став кульмінаційним етапом дослідження – моментом перетворення теоретичних концепцій на функціонуючий прототип мультимедійної клавіатури. Шлях від ідеї до реалізації виявився викликовим процесом, що вимагав не лише технічних знань, а й творчого підходу.

Розробка програмного забезпечення для сенсора PAJ7620 продемонструвала елегантність простого рішення – кілька десятків рядків коду здатні розпізнати вісім типів жестів. Проте за цією простотою криється складна взаємодія апаратних та програмних компонентів.

Тестування ефективності розкрило справжню природу системи. Оптимальний робочий діапазон 5-15 сантиметрів та вимога до помірної швидкості жестів окреслюють межі практичного застосування.

Розробка десктопної програми з Qt Designer трансформувала технічний прототип у користувацький продукт з інтуїтивним інтерфейсом. Фінальне тестування підтвердило життєздатність концепції – успішна демонстрація повного циклу роботи від розпізнавання жесту до виконання команди.

Таким чином, третій розділ не лише завершує технічну частину дослідження, а й відкриває перспективи для подальшого розвитку альтернативних інтерфейсів управління, демонструючи, що майбутнє цифрових технологій може бути значно більш інтуїтивним та природним.

ВИСНОВОК

Даній кваліфікаційній бакалаврській роботі реалізовано прототип мультимедійної клавіатури, яка функціонує на основі датчика жестів PAJ7620 та забезпечує безконтактне передавання команд до персонального комп’ютера. Робота виконувалась з метою дослідження та практичної реалізації системи керування мультимедійними функціями без застосування традиційних пристроїв введення. Та були виконано наступне:

– проведено аналіз сучасних методів безконтактного керування комп’ютером та огляд сенсорів жестів;

– обґрунтувано вибір апаратного забезпечення для реалізації мультимедійної клавіатури, зокрема датчика PAJ7620 та мікроконтролера Arduino Pro Micro;

– розроблено архітектуру апаратно-програмного комплексу для обробки жестових команд та емуляції HID-пристрою;

– реалізовано програмне забезпечення, яке забезпечує зчитування жестів, їхню обробку та передавання мультимедійних команд комп’ютеру;

– проведено тестування функціональності системи, оцінено точність розпізнавання жестів.

У ході аналізу було розглянуто наявні рішення у сфері безконтактного керування, а також досліджено принципи функціонування сенсора PAJ7620. Обґрунтовано вибір апаратної платформи.

Розроблено алгоритм, який реалізує послідовну обробку жестів, визначення їх напрямку, формування відповідної команди та передавання її на комп’ютер. Алгоритм реалізовано у вигляді прошивки для мікроконтролера, що забезпечує безперервне опитування сенсора, інтерпретацію подій жестів та емуляцію натискання клавіш.

Розроблено десктопний застосунок на мові C++ з використанням бібліотеки Qt. Він виконує функції прийому повідомлень з мікроконтролера, логування подій та налаштування відповідностей між жестами й комбінаціями клавіш. Реалізовано

графічний інтерфейс для редагування конфігурації, яка зберігається у форматі JSON та може бути оновлена користувачем.

Під час тестування перевірено коректність встановлення з’єднання з мікроконтролером, стабільність прийому повідомлень про жести, працездатність логування, а також функціональність налаштування й виконання команд. Тестування показало, що програма правильно інтерпретує отримані жести, оновлює конфігурацію відповідно до дій користувача та виконує емуляцію клавіш у відповідності до призначених жестів.

Отриманий прототип підтвердив можливість реалізації безконтактної взаємодії з комп’ютером. Застосунок працює у фоновому режимі та не потребує активної взаємодії з користувачем під час виконання жестових команд.

У результаті реалізовано завершене рішення, яке включає апаратну та програмну частини, забезпечує стабільний обмін даними між пристроями та дозволяє користувачу налаштовувати відповідності між жестами й клавішними командами. Такий підхід може бути масштабований для використання в інших проєктах з безконтактним керуванням, включаючи інтерфейси для людей з обмеженими можливостями, інтерактивні стенди або розумні робочі місця.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aiguo W., Huancheng L., Chundi Z., Huihui C., Chih-Yung C. Fusion of kinematic and physiological sensors for hand gesture recognition. *Multimedia Tools and Applications*. 2024. Vol. 83. P. 1–28. DOI: 10.1007/S11042-024-18283-Z.

2. All Qt Documentation. URL: https://doc.qt.io/all-topics.html (Last accessed: 25.05.2025).

3. Arduino Pro Micro V2 Type-C. URL: https://arduino.ua/prod6103-plata-arduino-pro-micro-v2-type-c?srsltid=AfmBOoquu3GRuuRnlAF3AeyN2MktZ1IeKUTEUqiTnmLpIagDY5jRMVMH (Last accessed: 08.05.2025).

4. Dong-Han M., Chuen-Lin T., Yu-Ling Y., Yi-Ru G., Chern-Sheng L., Chih-Chin C., Che-Ming C. *Design of Digital-Twin Human-Machine Interface Sensor with Intelligent Finger Gesture Recognition. Sensors*. 2023. Vol. 23. P. 1–24. DOI: 10.3390/s23073509.

5. Gesture recognition sensor module PAJ7620. Conrad. URL: https://asset.conrad.com/media10/add/160267/c1/-/gl/002380066DS00/podatkovna-tablica-2380066-iduino-tc-9520264-senzorski-modul-1-st-pogodno-za-komplet-za-razvoj-arduino.pdf (Last accessed: 15.05.2025).

6. Intel RealSense Depth Camera Module D421. URL: https://www.intelrealsense.com/depth-module-d421 (Last accessed: 07.05.2025).

7. Json. JSON. URL: https://www.json.org/json-en.html (Last accessed: 25.05.2025).

8. Kryvonos O. Specialized applied software fritzing and its use for simulation of electronic devices based on arduino platform. *Information technologies in education.* 2020. No. 43. P. 39–51. DOI: 10.14308/ite000718.

9. Leap Motion Controller. URL: https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf (Last accessed: 07.05.2025).

10. Liu Z. Leonardo's USB CDC serial port. Arduino Forum. URL: https://forum.arduino.cc/t/leonardos-usb-cdc-serial-port/260221 (Last accessed: 25.05.2025).

11. MYO armband to muscle into computer control. URL: phys.org/news/2013-04-myo-armband-muscle-video.html (Last accessed: 07.05.2025).

12. PAJ7620U2 Gesture Sensor. URL: https://www.waveshare.com/wiki/PAJ76 20U2_Gesture_Sensor (Last accessed: 08.05.2025).

13. PAJ7620U2: Integrated Gesture Recognition Sensor. URL: https://files.seeedstudio.com/wiki/Grove_Gesture_V_1.0/res/PAJ7620U2_DS_v1.5_05012022_Confidential.pdf (Last accessed: 08.05.2025).

14. Pro Micro & Fio V3 Hookup Guide. URL:https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide/hardware-overview-pro-micro (Last accessed: 07.05.2025).

15. Pro Micro. URL: docs.arduino.cc/hardware/ProMicro/ (Last accessed: 08.05.2025).

16. Software: Arduino IDE. URL: https://www.arduino.cc/en/software (Last accessed: 07.05.2025).

17. Tales In Tech History: Microsoft Kinect. URL: https://www.silicon.co.uk/e-innovation/microsoft-kinect-history-226781 (Last accessed: 07.05.2025).

18. The Rise and Fall of Google's Project Soli: A Promising Vision to a Quiet Fade. URL: https://www.linkedin.com/pulse/rise-fall-googles-project-soli-promising-vision-quiet-krishnan-isrpc (Last accessed: 07.05.2025).

19. Use case diagram. GeeksforGeeks. URL: https://www.geeksforgeeks.org/use-case-diagram/ (Last accessed: 25.05.2025).

20. Матвієнко С. I2C інтерфейс. IT Master. URL: https://itmaster.biz.ua/directory/standarts/i2c.html (дата звернення: 27.05.2025).

21. Яковенко Т. К. Датчики руху на основі мікроконтролера Arduino в інтелектуальних системах безпеки. 2024. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/95538/1/Yakovenko_bak_rob.pdf (дата звернення: 27.05.2025).

ДОДАТОК А

Код програми

Лістинг А.1 – XML-розмітка інтерфейсу користувача

```
<?xml version=“1.0” encoding=“UTF-8”?>
<ui version=“4.0”>
 <class>MainWindow</class>
 <widget class=“QMainWindow” name=“MainWindow”>
  <property name=“geometry”>
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>490</height>
   </rect>
  </property>
  <property name=“font”>
   <font>
    <pointsize>12</pointsize>
   </font>
  </property>
  <property name=“focusPolicy”>
   <enum>Qt::FocusPolicy::NoFocus</enum>
  </property>
  <property name=“windowTitle”>
   <string>GestureControl</string>
  </property>
  <widget class=“QWidget” name=“centralwidget”>
   <property name=“focusPolicy”>
    <enum>Qt::FocusPolicy::NoFocus</enum>
   </property>
   <layout class=“QVBoxLayout” name=“verticalLayout_4”>
    <item>
     <layout class=“QFormLayout” name=“formLayout”>
      <item row=“1” column=“0”>
       <widget class=“QLabel” name=“label”>
        <property name=“text”>
         <string>Праворуч</string>
        </property>
       </widget>
      </item>
      <item row=“2” column=“0”>
       <widget class=“QLabel” name=“label_2”>
        <property name=“text”>
         <string>Ліворуч</string>
        </property>
       </widget>
      </item>
      <item row=“3” column=“0”>
       <widget class=“QLabel” name=“label_3”>
        <property name=“text”>
```

```
   <string>Вгору</string>
  </property>
 </widget>
</item>
<item row="4" column="0">
 <widget class="QLabel" name="label_4">
  <property name="text">
   <string>Вниз</string>
  </property>
 </widget>
</item>
<item row="5" column="0">
 <widget class="QLabel" name="label_5">
  <property name="text">
   <string>Вперед</string>
  </property>
 </widget>
</item>
<item row="6" column="0">
 <widget class="QLabel" name="label_6">
  <property name="text">
   <string>Назад</string>
  </property>
 </widget>
</item>
<item row="7" column="0">
 <widget class="QLabel" name="label_7">
  <property name="text">
   <string>За год. стрілкою</string>
  </property>
 </widget>
</item>
<item row="8" column="0">
 <widget class="QLabel" name="label_8">
  <property name="text">
   <string>Проти год. стрілки</string>
  </property>
 </widget>
</item>
<item row="9" column="0">
 <widget class="QLabel" name="label_9">
  <property name="text">
   <string>Помахати</string>
  </property>
 </widget>
</item>
<item row="1" column="1">
 <widget class="QKeySequenceEdit" name="keySequenceEditRight">
  <property name="enabled">
   <bool>false</bool>
  </property>
 </widget>
</item>
```

```
<item row=“2” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditLeft”>
  <property name=“enabled”>
   <bool>false</bool>
  </property>
 </widget>
</item>
<item row=“3” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditUp”>
  <property name=“enabled”>
   <bool>false</bool>
  </property>
 </widget>
</item>
<item row=“4” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditDown”>
  <property name=“enabled”>
   <bool>false</bool>
  </property>
 </widget>
</item>
<item row=“5” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditForward”>
  <property name=“enabled”>
   <bool>false</bool>
  </property>
 </widget>
</item>
<item row=“6” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditBackward”>
  <property name=“enabled”>
   <bool>false</bool>
  </property>
 </widget>
</item>
<item row=“7” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditRotateRight”>
  <property name=“enabled”>
   <bool>false</bool>
  </property>
 </widget>
</item>
<item row=“8” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditRotateLeft”>
  <property name=“enabled”>
   <bool>false</bool>
  </property>
 </widget>
</item>
<item row=“9” column=“1”>
 <widget class=“QKeySequenceEdit” name=“keySequenceEditWave”>
  <property name=“enabled”>
   <bool>false</bool>
```

```
    </property>
   </widget>
  </item>
  <item row="0" column="0">
   <widget class="QLabel" name="label_10">
    <property name="font">
     <font>
      <pointsize>12</pointsize>
      <bold>true</bold>
     </font>
    </property>
    <property name="text">
     <string>Жести</string>
    </property>
   </widget>
  </item>
  <item row="0" column="1">
   <widget class="QLabel" name="label_11">
    <property name="font">
     <font>
      <pointsize>12</pointsize>
      <bold>true</bold>
     </font>
    </property>
    <property name="text">
     <string>Відповідні комбінації</string>
    </property>
   </widget>
  </item>
 </layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_5">
  <property name="spacing">
   <number>6</number>
  </property>
  <item>
   <widget class="QPushButton" name="pushButtonEdit">
    <property name="text">
     <string>Змінити</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QPushButton" name="pushButtonSave">
    <property name="enabled">
     <bool>false</bool>
    </property>
    <property name="text">
     <string>Зберегти</string>
    </property>
   </widget>
  </item>
```

```
      <item>
       <widget class="QPushButton" name="pushButtonDefault">
        <property name="text">
         <string>Скинути до значень за замовчуванням</string>
        </property>
       </widget>
      </item>
     </layout>
    </item>
    <item>
     <widget class="QLabel" name="statusLabel">
      <property name="text">
       <string/>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

Лістинг А.2 – Заголовочний файл програми

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void readSerialData();

    void on_pushButtonEdit_clicked();

    void on_pushButtonSave_clicked();

    void on_pushButtonDefault_clicked();
```

```
private:
    void initSerial();
    void loadGesturesConfig();
    void setEditingEnabled(bool);
    void simulateKeySequence(const QKeySequence &sequence);
#if defined(Q_OS_WIN)
    bool nativeEvent(const QByteArray &eventType, void *message, qintptr *result)
override;
#endif

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

А.3 – Код десктопної програми

```
void MainWindow::initSerial()
{
    qDebug() << “Доступні COM-порти:”;

    foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts()) {
        qDebug() << “Порт:” << info.portName();
        qDebug() << “Опис:” << info.description();
        qDebug() << “Виробник:” << info.manufacturer();
        qDebug() << “Серійний номер:” << info.serialNumber();
        qDebug() << “Системний шлях:” << info.systemLocation();
        qDebug() << “------------------------------------------------”;
    }

    // Подальша логіка пошуку Arduino або USB-плати
    foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts()) {
        if (info.manufacturer().contains(“Arduino”) || info.description().contains(“USB”))
{
            serial = new QSerialPort(info, this);
            serial->setBaudRate(QSerialPort::Baud115200);
            serial->setDataBits(QSerialPort::Data8);
            serial->setParity(QSerialPort::NoParity);
            serial->setStopBits(QSerialPort::OneStop);
            serial->setFlowControl(QSerialPort::NoFlowControl);
            if (serial->open(QIODevice::ReadOnly)) {
                connect(serial, &QSerialPort::readyRead, this,
&MainWindow::readSerialData);
                ui->statusLabel->setText(“З'єднано: “ + info.portName());
                return;
            }
        }
    }

    ui->statusLabel->setText(“Плата не знайдена”);
}
```

```
void MainWindow::loadGesturesConfig()
{
    QFile file(“gestures.json”);
    if (!file.exists()) {
        file.setFileName(“:/gestures.json”);
    }

    if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        qDebug() << “Не вдалося відкрити gestures.json”;
        return;
    }

    QByteArray jsonData = file.readAll();
    file.close();

    QJsonDocument doc = QJsonDocument::fromJson(jsonData);
    if (doc.isNull() || !doc.isObject()) {
        qDebug() << “Невірний формат JSON”;
        return;
    }

    QJsonObject obj = doc.object();

    if (obj.contains(“Left”)) {
        ui->keySequenceEditLeft-
>setKeySequence(QKeySequence(obj.value(“Left”).toString()));
    }
    if (obj.contains(“Right”)) {
        ui->keySequenceEditRight-
>setKeySequence(QKeySequence(obj.value(“Right”).toString()));
    }
    if (obj.contains(“Up”)) {
        ui->keySequenceEditUp->setKeySequence(QKeySequence(obj.value(“Up”).toString()));
    }
    if (obj.contains(“Down”)) {
        ui->keySequenceEditDown-
>setKeySequence(QKeySequence(obj.value(“Down”).toString()));
    }
    if (obj.contains(“Forward”)) {
        ui->keySequenceEditForward-
>setKeySequence(QKeySequence(obj.value(“Forward”).toString()));
    }
    if (obj.contains(“Backward”)) {
        ui->keySequenceEditBackward-
>setKeySequence(QKeySequence(obj.value(“Backward”).toString()));
    }
    if (obj.contains(“RotateRight”)) {
        ui->keySequenceEditRotateRight-
>setKeySequence(QKeySequence(obj.value(“RotateRight”).toString()));
    }
    if (obj.contains(“RotateLeft”)) {
```

```
        ui->keySequenceEditRotateLeft-
>setKeySequence(QKeySequence(obj.value(“RotateLeft”).toString()));
    }
    if (obj.contains(“Wave”)) {
        ui->keySequenceEditWave-
>setKeySequence(QKeySequence(obj.value(“Wave”).toString()));
    }
}


void MainWindow::on_pushButtonEdit_clicked()
{
    setEditingEnabled(true);
}

void MainWindow::setEditingEnabled(bool enabled)
{
    ui->keySequenceEditLeft->setEnabled(enabled);
    ui->keySequenceEditRight->setEnabled(enabled);
    ui->keySequenceEditUp->setEnabled(enabled);
    ui->keySequenceEditDown->setEnabled(enabled);
    ui->keySequenceEditForward->setEnabled(enabled);
    ui->keySequenceEditBackward->setEnabled(enabled);
    ui->keySequenceEditRotateRight->setEnabled(enabled);
    ui->keySequenceEditRotateLeft->setEnabled(enabled);
    ui->keySequenceEditWave->setEnabled(enabled);

    ui->pushButtonSave->setEnabled(enabled);
    ui->pushButtonEdit->setEnabled(!enabled);

    QTimer::singleShot(50, this, [this]() {
        ui->pushButtonSave->setDown(false);
        ui->pushButtonEdit->setDown(false);
        ui->pushButtonSave->update();
        ui->pushButtonEdit->update();
    });
}

void MainWindow::on_pushButtonSave_clicked()
{
    QJsonObject obj;

    obj[“Left”] = ui->keySequenceEditLeft->keySequence().toString();
    obj[“Right”] = ui->keySequenceEditRight->keySequence().toString();
    obj[“Up”] = ui->keySequenceEditUp->keySequence().toString();
    obj[“Down”] = ui->keySequenceEditDown->keySequence().toString();
    obj[“Forward”] = ui->keySequenceEditForward->keySequence().toString();
    obj[“Backward”] = ui->keySequenceEditBackward->keySequence().toString();
    obj[“RotateRight”] = ui->keySequenceEditRotateRight->keySequence().toString();
    obj[“RotateLeft”] = ui->keySequenceEditRotateLeft->keySequence().toString();
    obj[“Wave”] = ui->keySequenceEditWave->keySequence().toString();

    QJsonDocument doc(obj);
```

```
    QFile file(“gestures.json”);
    if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        qDebug() << “Не вдалося відкрити gestures_extend.json для запису”;
        return;
    }

    file.write(doc.toJson(QJsonDocument::Indented));
    file.close();

    setEditingEnabled(false);
}

void MainWindow::on_pushButtonDefault_clicked()
{
    QFile resourceFile(“:/gestures.json”);
    if (!resourceFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
        qDebug() << “Не вдалося відкрити ресурсний gestures.json”;
        return;
    }

    QByteArray defaultData = resourceFile.readAll();
    resourceFile.close();

    QFile localFile(“gestures.json”);
    if (!localFile.open(QIODevice::WriteOnly | QIODevice::Text)) {
        qDebug() << “Не вдалося створити gestures.json у файловій системі”;
        return;
    }

    localFile.write(defaultData);
    localFile.close();

    loadGesturesConfig();
    qDebug() << “Повернуто налаштування жестів за замовчуванням”;
}

void MainWindow::readSerialData()
{
    QByteArray data = serial->readAll();
    QString str = QString::fromUtf8(data).trimmed();
    qDebug() << “Отримано жест:” << str;

    QMap<QString, QKeySequenceEdit*> gestureMap = {
        {“Left”, ui->keySequenceEditLeft},
        {“Right”, ui->keySequenceEditRight},
        {“Up”, ui->keySequenceEditUp},
        {“Down”, ui->keySequenceEditDown},
        {“Forward”, ui->keySequenceEditForward},
        {“Backward”, ui->keySequenceEditBackward},
        {“RotateRight”, ui->keySequenceEditRotateRight},
        {“RotateLeft”, ui->keySequenceEditRotateLeft},
        {“Wave”, ui->keySequenceEditWave}
```

```
    };

    if (gestureMap.contains(str)) {
#ifdef Q_OS_WIN
        QKeySequence seq = gestureMap[str]->keySequence();
        simulateKeySequence(seq);
#endif
    }
}

static void sendKey(WORD vk, bool keyDown, QList<INPUT> &inputs) {
    INPUT input = {};
    input.type = INPUT_KEYBOARD;
    input.ki.wVk = vk;
    input.ki.dwFlags = keyDown ? 0 : KEYEVENTF_KEYUP;
    inputs.append(input);
}

static WORD qtKeyToVK(Qt::Key key) {
    switch (key) {
    case Qt::Key_Control: return VK_CONTROL;
    case Qt::Key_Shift: return VK_SHIFT;
    case Qt::Key_Alt: return VK_MENU;
    case Qt::Key_Meta: return VK_LWIN;
    case Qt::Key_Enter: return VK_RETURN;
    case Qt::Key_Return: return VK_RETURN;
    case Qt::Key_Tab: return VK_TAB;
    case Qt::Key_Escape: return VK_ESCAPE;
    case Qt::Key_Space: return VK_SPACE;
    case Qt::Key_F4: return VK_F4;
    default:
        return static_cast<WORD>(key);
    }
}

#ifdef Q_OS_WIN
bool MainWindow::nativeEvent(const QByteArray &eventType, void *message, qintptr *result)
{
    MSG* msg = static_cast<MSG*>(message);

    if (msg->message == WM_SYSKEYDOWN && msg->wParam == VK_F4) {
        qDebug() << “Перехоплено Alt+F4”;

        QWidget *focused = QApplication::focusWidget();
        if (focused) {
            QKeySequenceEdit *keySeqEdit = qobject_cast<QKeySequenceEdit*>(focused);
            if (keySeqEdit) {
                keySeqEdit->setKeySequence(QKeySequence(Qt::ALT + Qt::Key_F4));
                qDebug() << “Встановлено Alt+F4 у QKeySequenceEdit”;
            } else {
                qDebug() << “Фокус не на QKeySequenceEdit”;
            }
        } else {
```

```
            qDebug() << “Немає фокусу”;
        }

        *result = 0;
        return true;
    }

    return false;
}
#endif

#ifdef Q_OS_WIN
void MainWindow::simulateKeySequence(const QKeySequence &sequence)
{
    if (sequence.count() == 0)
        return;

    QKeyCombination combination = sequence[0];
    QList<INPUT> inputs;

    Qt::KeyboardModifiers mods = combination.keyboardModifiers();
    Qt::Key key = static_cast<Qt::Key>(combination.key());

    if (mods & Qt::ControlModifier)
        sendKey(VK_CONTROL, true, inputs);
    if (mods & Qt::AltModifier)
        sendKey(VK_MENU, true, inputs);
    if (mods & Qt::ShiftModifier)
        sendKey(VK_SHIFT, true, inputs);
    if (mods & Qt::MetaModifier)
        sendKey(VK_LWIN, true, inputs);

    sendKey(qtKeyToVK(key), true, inputs);
    sendKey(qtKeyToVK(key), false, inputs);

    if (mods & Qt::MetaModifier)
        sendKey(VK_LWIN, false, inputs);
    if (mods & Qt::ShiftModifier)
        sendKey(VK_SHIFT, false, inputs);
    if (mods & Qt::AltModifier)
        sendKey(VK_MENU, false, inputs);
    if (mods & Qt::ControlModifier)
        sendKey(VK_CONTROL, false, inputs);

    SendInput(inputs.count(), inputs.data(), sizeof(INPUT));
}
#endif
```

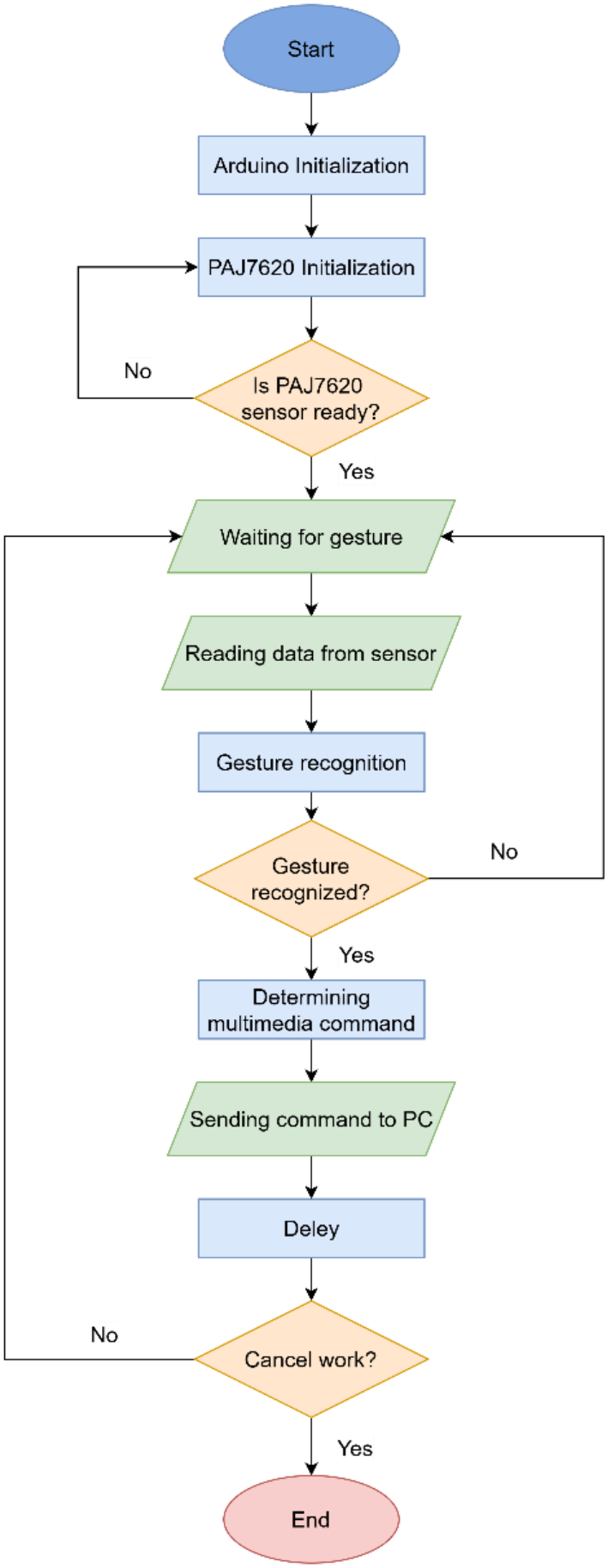

Рисунок А.1 – Блок-схема роботи алгоритму клавіатури

ДОДАТОК Б

Матеріали апробації роботи

DOI:

Станіслав Гануліч,
Борис Салтовський

РОЗРОБКА МУЛЬТИМЕДІЙНОЇ КЛАВІАТУРИ НА БАЗІ ДАТЧИКА PAJ7620

Пропонується безконтактний засіб керування комп'ютером за допомогою HID-пристрою на базі датчика PAJ7620. Цей датчик дозволяє розпізнавати жести які потім передаються в комп'ютер як скан-коди клавіш. Завдяки універсальності HID протоколу пристрій може працювати під будь-якими операційними системами, в тому числі і мобільними.

Ключові слова: *клавіатура, датчики, PAJ7620, USB, HID, Arduino, безконтактні пристрої.*

У сучасну епоху цифрової трансформації питання зручності, швидкості та інтуїтивності взаємодії людини з комп’ютером стає одним із ключових у розробці нових інтерфейсів. Особливої актуальності набувають безконтактні способи керування, які дозволяють мінімізувати фізичний контакт з пристроями та звільнити користувача від залежності від традиційних методів введення. У цій роботі розглядається проєкт створення мультимедійної клавіатури, яка використовує датчик жестів PAJ7620 і мікроконтролер Arduino Leonardo, пропонуючи новий підхід до управління медіаконтентом через природні рухи рук.

Мультимедійні пристрої стали невід'ємною частиною як побутового, так і професійного середовища. Водночас потреба в швидкому доступі до функцій управління — таких як регулювання гучності, перемикання треків, пауза, відтворення, заглушення звуку — зростає пропорційно до кількості використовуваних платформ, сервісів і застосунків. Звичні клавіатури та миші не завжди відповідають цим вимогам, особливо в умовах обмеженого простору або при необхідності частих перемикань між задачами. Це відкриває шлях для альтернативних рішень, зокрема жестових інтерфейсів.

У світі вбудованих систем, де кожен міліметр простору та мікроват енергії має значення, датчик жестів PAJ7620 (рис. 1) представляє собою унікальне поєднання компактності, енергоефективності та функціональності. Цей мініатюрний оптичний сенсор, розроблений компанією PixArt Imaging Inc., став справжнім проривом у демократизації технологій жестового керування, дозволивши навіть простим мікроконтролерним системам розпізнавати жести людини без необхідності у складних обчислювальних алгоритмах та потужних процесорах.

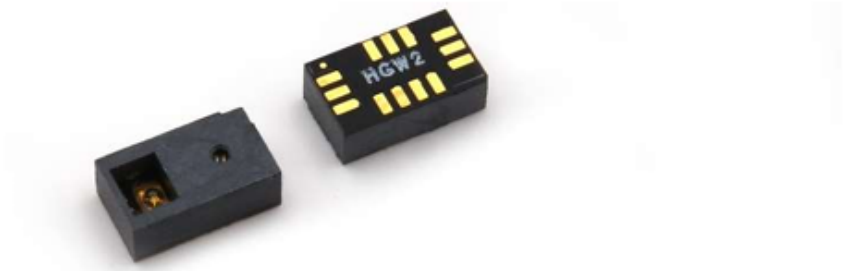

Рисунок 1 — Датчик жестів PAJ7620

PAJ7620 базується на технології інфрачервоного (ІЧ) оптичного розпізнавання. Це система, що інтегрує ІЧ-випромінювачі, масив фотодіодів та спеціалізовану систему обробки сигналів. Саме ця інтеграція дозволяє датчику виконувати розпізнавання жестів автономно, передаючи на мікроконтролер вже готові результати, а не первинні дані, які потребують подальшої обробки.

Ключовою особливістю PAJ7620 (рис.2) є його здатність аналізувати послідовні кадри таких зображень для виявлення руху. Вбудований процесор датчика обчислює вектори руху між послідовними кадрами, визначаючи напрямок, швидкість та характер переміщення об'єкта. Ця інформація обробляється за допомогою вбудованих алгоритмів розпізнавання образів, які порівнюють отримані дані з заздалегідь запрограмованими шаблонами жестів.

Рисунок 2 — Модуль з датчиком PAJ7620U2

Стандартна конфігурація PAJ7620 дозволяє розпізнавати дев'ять базових жестів: рух вгору, вниз, вліво, вправо, вперед (наближення до сенсора), назад (віддалення від сенсора), за годинниковою стрілкою, проти годинникової стрілки та коливальний рух. Цього набору жестів достатньо для реалізації базового управління мультимедійними функціями, такими як регулювання гучності, перемикання треків чи каналів, паузи та відновлення відтворення.

Для обробки інформації з датчика використовується плата Arduino Pro Micro (рис. 3). Вона відрізняється від інших плат Arduino насамперед своєю архітектурою. Мікроконтролер ATmega32u4, який лежить в основі плати, інтегрує USB-контролер безпосередньо в чіп, що усуває необхідність у додатковому перетворювачі USB-UART. Ця особливість має кардинальне значення для проєктів, що потребують безпосередньої взаємодії з комп'ютером, оскільки дозволяє платі емулювати різноманітні периферійні пристрої введення, такі як клавіатура або миша.

Рисунок 3 — Плата Arduino Pro Micro

Найбільш значущою перевагою Arduino Pro Micro для реалізації мультимедійної клавіатури на основі жестів є здатність емулювати HID-пристрої. Завдяки вбудованому USB-контролеру, Pro Micro може функціонувати як стандартна комп'ютерна клавіатура, миша або джойстик без необхідності в додатковому апаратному забезпеченні. Ця можливість відкриває широкі перспективи для створення альтернативних інтерфейсів введення.

Бібліотека Keyboard.h, що входить до стандартного набору Arduino IDE, дозволяє легко реалізувати функції емуляції клавіатури. Це особливо корисно для системи керування жестами, оскільки дозволяє асоціювати певні жести з відповідними командами клавіатури, що виконуються в різних програмах.

Наприклад, жест «змахування вправо» може бути інтерпретований як команда «наступний трек» у медіаплеєрі, що технічно реалізується як натискання клавіші «Next Track» або комбінації «Ctrl+Right». Аналогічно, жести можуть відповідати командам регулювання гучності, паузи/відтворення, перемотування та інших мультимедійних функцій.

Плати між собою з’єднуються за допомогою протоколу I²C (рис. 4).

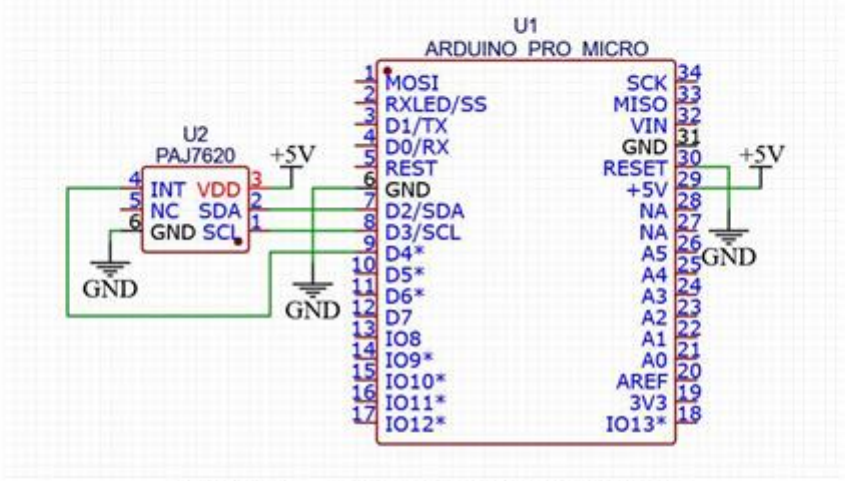

Рисунок 4 — Принципова схема пристрою

На макетній платі пристрій має наступний вигляд (рис. 5).

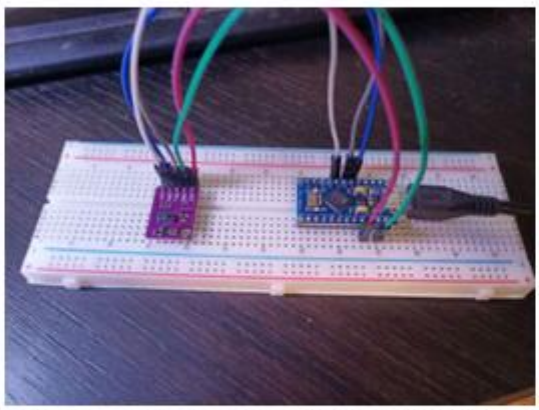

Рисунок 5 — Загальний вигляд прототипу пристрою

Таким чином, алгоритми розпізнавання комбінацій жестів та послідовностей відкривають ще ширші можливості для взаємодії з мультимедійними системами. Подібно до того, як комбінації клавіш розширюють функціональність клавіатури, послідовності та комбінації жестів дозволяють реалізувати більш складні команди без необхідності вводити додаткові базові жести. Наприклад, жест "змахування вправо" з подальшим утриманням руки

в полі зору датчика може інтерпретуватися як команда перемотування вперед, де тривалість утримання визначає швидкість перемотування. Такі комплексні інтерпретації вимагають більш складних алгоритмів, які враховують не лише тип жесту, але й часові характеристики, послідовність та взаємозв'язок різних рухів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Pro Micro & Fio V3 Hookup Guide: https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide/all (Last accessed: 18.05.2025).
2. Pro Micro: https://github.com/sparkfun/Pro_Micro (Last accessed: 18.05.2025).
3. PAJ7620U2 Gesture Sensor URL: https://www.waveshare.com/wiki/PAJ7620U2_Gesture_Sensor (Last accessed: 18.05.2025).
4. PAJ7620U2: Integrated Gesture Recognition Sensor URL: https://files.seeedstudio.com/wiki/Grove_Gesture_V_1.0/res/PAJ7620U2_DS_v1.5_05012022_Confidential.pdf (Last accessed: 18.05.2025).

DEVELOPMENT OF A MULTIMEDIA KEYBOARD BASED ON THE PAJ7620 SENSOR

A contactless computer control device based on the PAJ7620 sensor is proposed. This sensor enables gesture recognition, which is then transmitted to the computer as key scan codes. Thanks to the versatility of the HID protocol, the device can operate on any operating system, including mobile platforms.

Key words: *keyboard, sensors, PAJ7620, USB, HID, Arduino, contactless devices.*

Відомості про авторів / Information about the Authors

Станіслав Гануліч, бакалаврант кафедри комп’ютерної інженерії Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: stanis26060@gmail.com, orcid: https://orcid.org/0009-0004-6948-7862.

Stanislav Hanulich, Bachelor’s student of the Computer Engineering Department at Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. Email: stanis26060@gmail.com, orcid: https://orcid.org/0009-0004-6948-7862.

Борис Салтовський, старший викладач кафедри комп’ютерної інженерії Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: borys.saltovskyi@chmnu.edu.ua, orcid: https://orcid.org/0009-0006-8296-201X.

Borys Saltovskyi, Senior Lecturer at the Department of Computer Engineering, Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. E-mail: borys.saltovskyi@chmnu.edu.ua, orcid: https://orcid.org/0009-0006-8296-201X.