Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Райковський Роман Русланович

на здобуття ступеня вищої освіти Бакалавра

Система контролювання доступу на основі голосового паролю

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ.2102162.21.02.23 ПЗ

Виконав студент 4 курсу група КБ-21-2 Роман РАЙКОВСЬКИЙ
Керівник канд. техн. наук, доцент Володимир ПЕТРУШАК
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ
11 06 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ _____

15 лютого 2025 р.

З А В Д А Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Райковський Роман Русланович

1 Тема роботи Система контролювання доступу на основі голосового паролю

Керівник роботи Володимир Петрушак

Затверджено наказом ректора університету від 7 лютого 2025 № 23

2 Строк подання студентом кваліфікаційної роботи на кафедру 17.06.2025

3 Вихідні дані до роботи Розробити ефективну систему контролю доступу на основі голосового паролю, яка забезпечує ідентифікацію та автентифікацію користувачів за голосовими характеристиками. Дослідити предметну область біометричної автентифікації, зокрема методи обробки, зберігання та розпізнавання голосових сигналів, а також загрози інформаційній безпеці, пов'язані з використанням голосових паролів.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Система контролю доступу на основі голосового паролю використовує унікальні голосові характеристики для ідентифікації користувача. Для розробки було проведено аналіз біометричних методів автентифікації, зокрема текст-залежної голосової біометрії. Проєктування системи передбачає модулі захоплення, обробки, зберігання та верифікації голосових даних.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
«Алгоритм роботи системи», «Структура програмного забезпечення системи захисту конфіденційних даних», «Схема процесу проведення оглядового дослідження»

6 Консультанти розділів кваліфікаційної роботи

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

7 Дата видачі завдання 19 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	Лютий	
Аналіз загроз і вразливостей традиційних систем контролю доступ	Лютий	
Дослідження сучасних методів біометричної аутентифікації	Лютий	
Постановка задачі та визначення вимог до системи	Березень	
Визначення загальних принципів побудови системи на основі голосового паролю	Березень	
Розробка архітектури системи контролю доступу	Квітень	
Реалізація механізмів запису, зберігання та розпізнавання голосу	Квітень	
Тестування системи на точність та надійність аутентифікації	Травень	
Оформлення пояснювальної записки згідно вимог	Травень	
Оформлення графічної частини	Червень	
Захист КР	Червень	

Студент Роман РАЙКОВСЬКИЙ

Керівник кваліфікаційної роботи Володимир ПЕТРУШАК

АНОТАЦІЯ

Тема кваліфікаційної роботи: Система контролювання доступу на основі голосового паролю.

Автор роботи: Райковський Роман Русланович.

Керівник роботи: Петрушак Володимир Степанович.

Пояснювальна записка:69 с., 2 додатки,9 рисунків,1 таблиці,40 джерел.

Графічна частина:3 плакати.

СИСТЕМА КОНТРОЛЮ ЗАХИСТУ НА ОСНОВІ ГОЛОСОВОГО ПАРОЛЮ.

У дипломній роботі досліджено можливість побудови простої та доступної системи контролю доступу на основі голосового паролю. Розглянуто переваги й обмеження голосової автентифікації у порівнянні з іншими біометричними та не біометричними методами, а також проаналізовано типові загрози, пов'язані з використанням голосу як ідентифікатора.

У рамках реалізації створено програмно-апаратний прототип, що поєднує мікроконтролер Arduino Nano та програмне забезпечення на мові Python із використанням бібліотеки Vosk для розпізнавання мовлення в офлайн-режимі. Система не зберігає біометричні шаблони, працює локально та дозволяє здійснювати контроль доступу на основі фіксованої голосової фрази.

Отримані результати демонструють, що така архітектура може бути використана у невеликих офісах, лабораторіях або навчальних закладах як економічно ефективне рішення з базовим рівнем безпеки.

09.06.2025 _____

ABSTRACT

Subject of qualification work: Confidential data protection system against malicious software.

Author: Raikovsky Roman Ruslanovich.

Head of work: Petrushak Volodymyr Stepanovych.

Explanatory note:69 p., 2 appendices,9 figures,1 tables, 40 sources

Graphic part:3 posters.

A VOICE PASSWORD-BASED SECURITY CONTROL SYSTEM.

This thesis explores the development of a simple and cost-effective access control system based on a voice password. The advantages and limitations of voice authentication are analyzed in comparison with other biometric and non-biometric methods, along with the most common security threats associated with using voice as an identifier.

As part of the implementation, a hardware-software prototype was developed using an Arduino Nano microcontroller and Python-based software with the Vosk speech recognition library operating in offline mode. The system does not store biometric templates, runs locally, and grants access based on a predefined voice phrase.

The results show that this architecture can be effectively applied in small offices, laboratories, or educational institutions as a low-cost solution with a basic level of security.

09.06.2025 

ЗМІСТ

Зм.	Арк.	№докум.	Підпис	Дата	КРБКБ.2102162.21.02.23 ПЗ		
Виконав		Райковський Р.Р.		03.06.25	Система контролювання доступу на основі голосового паролю Пояснювальна записка	Літера	Аркуш 6 / Аркушів 70
Перевір.		Петрушак В.С.		06.25			
Н.контр.		Мостовий С.В		11.06.25	ХНУ, КБ-21-2		
Затвер.		Кльоц Ю.П.		11.06.25			

ВСТУП

Технології розпізнавання голосу активно проникають у сферу безпеки, стаючи заміною традиційним методам ідентифікації. Стрімкий розвиток штучного інтелекту, нейронних мереж і мобільних пристроїв стимулює впровадження інноваційних методів, які поєднують високу безпеку з максимальною зручністю для користувача. Автентифікація за голосом аналізує індивідуальні ознаки мовлення, що дозволяє з високою точністю розпізнавати користувачів.

Голос як засіб доступу дозволяє уникнути запам'ятовування кодів або носіння карток — достатньо вимовити фразу. Достатньо просто вимовити певну фразу, щоб система перевірила і дозволила або заборонила доступ. Такий підхід не тільки зручний, але й потенційно безпечніший за класичні методи, оскільки унікальність голосу складно підробити без спеціалізованих інструментів. Система голосової автентифікації аналізує понад сімдесят параметрів мовлення включаючи тембр, ритм, інтонацію, паузи, висоту і швидкість - щоб створити індивідуальний голосовий «відбиток», подібний до відбитків пальців.

Інтерес до голосової автентифікації з боку промисловості та державних установ постійно зростає. За даними Opus Research, до кінця 2023 року близько 1,9 мільярда банківських користувачів у всьому світі вже використовуватимуть цю технологію. Прогнозується, що у 2025 році світовий ринок голосової біометрії сягне понад $27 млрд, що свідчить про стабільне визнання ефективності цього методу та його широке впровадження в різних сферах.

Особливо активно технологія впроваджується у фінансовому секторі, де ризики шахрайства та несанкціонованого доступу залишаються надзвичайно високими. Голосова ідентифікація використовується тут як додатковий рівень захисту при обслуговуванні клієнтів по телефону, проведенні транзакцій або підтвердженні особи при вході в мобільний додаток. Завдяки здатності забезпечувати безконтактну автентифікацію, голосова біометрія також

виявляється незамінною в умовах пандемії або в інклюзивному середовищі, наприклад, для людей з інвалідністю, які не можуть вводити дані вручну.

Однак, як і будь-яка інша технологія, розпізнавання голосу має не лише переваги, але й виклики. Однією з головних проблем є вразливість до атак, зокрема так званих replay-атак, коли зловмисники намагаються обдурити систему, відтворюючи заздалегідь записаний голос користувача. Крім того, стрімкий розвиток технологій синтезу мовлення, таких як генеративні нейронні мережі та системи deepfake, створює загрозу створення фальшивих голосів, які майже неможливо відрізнити від справжніх на слух. Це вимагає від розробників розробки нових підходів до перевірки «живості» мови, застосування багатофакторної автентифікації та впровадження криптографічного захисту даних, що передаються.

Не менш важливими є питання конфіденційності та етики. Згідно із законодавством більшості країн, у тому числі й України, біометричні дані, зокрема голос, відносяться до категорії чутливих персональних даних. Їх обробка вимагає однозначної згоди користувача, чітких правил зберігання, шифрування та обмеженого доступу. Ці вимоги накладають додаткові обмеження на розробку голосових систем і зобов'язують інтеграторів враховувати не тільки технічні, а й юридичні аспекти.Незважаючи на ці обмеження, потенціал голосової автентифікації величезний. Досягнення в галузі штучного інтелекту, глибокого навчання та обробки мовлення допомагають підвищити точність систем розпізнавання навіть у складних акустичних умовах. У майбутньому очікується інтеграція голосових технологій із системами віртуальної та доповненої реальності, робототехнікою та пристроями Інтернету речей, що відкриє нові горизонти для безконтактної взаємодії між людьми та машинами.

Таким чином, голосова автентифікація - це не просто зручна альтернатива паролям, а перспективний напрямок розвитку систем безпеки, що поєднує в собі технічні інновації, ергономічність і високий рівень персоналізації.

1 АНАЛІЗ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ НА ОСНОВІ ГОЛОСОВОГО ПАРОЛЮ

1.1 Загальна концепція та вимоги

Сучасні системи контролю доступу визначаються як комплекс апаратних і програмних засобів, спрямованих на регулювання доступу користувачів до певних зон або ресурсів. Вони поєднують у собі фізичні бар'єри, до яких можуть належати дверні замки, турнікети, автоматичні ворота, а також електронні компоненти, такі як контролери, зчитувачі, датчики та сервісні сервери.

Основне призначення таких систем - забезпечення безпеки об'єкта, що досягається за рахунок чіткого розмежування прав користувачів і контролю за їхньою діяльністю. Згідно з Державним стандартом України ДСТУ 4000-2000, який діє з 2001 року, «система контролю доступу» (СКД) - це комплекс технічних, програмних і організаційних заходів, які служать для управління доступом, реєстрації подій і своєчасного реагування на спроби несанкціонованого вторгнення.

Завдяки уніфікації термінології та вимог, викладених у цьому стандарті, стало можливим синхронізувати впровадження різних технологічних рішень, забезпечити сумісність обладнання різних виробників, інтегрувати з іншими системами безпеки, такими як відеоспостереження або пожежна сигналізація[16].

У своїй базовій конфігурації СКУД складається з фізичних бар'єрів, зчитувачів ідентифікаційних даних, контролерів для обробки вхідної інформації та серверних модулів, де зберігаються правила доступу та ведуться журнали подій.

Коли користувач торкається картки, вводить код або пропонує зразок біометричного параметра, система аналізує дані, звіряє їх з базою даних і збереженими шаблонами і приймає рішення про надання або відмову в доступі.

При цьому всі події фіксуються, що дозволяє адміністраторам відстежувати спроби входу, аналізувати історію доступу та вчасно виявляти критичні інциденти.

З огляду на те, що сучасні загрози стають дедалі складнішими і вимагають багаторівневого підходу до захисту, актуальність розробки та впровадження ефективних СКУД є надзвичайно високою. Поєднання сучасних апаратно-програмних рішень з процедурними заходами, такими як регулярне оновлення програмного забезпечення, навчання персоналу та розробка чітких інструкцій з використання, не тільки запобігає несанкціонованому доступу, а й підвищує загальний рівень оперативного реагування на інциденти.

У великій організації, де кількість користувачів може сягати кількох тисяч, СКУД не лише контролює сам процес входу, але й слугує джерелом статистичних даних для аналізу пікової активності, співвіднесення з іншими системами контролю присутності співробітників та оцінки ефективності заходів безпеки.

Технічні вимоги до сучасних СКУД включають такі аспекти, як швидкість ідентифікації, надійність апаратного забезпечення, стійкість алгоритму до спроб несанкціонованого втручання, масштабованість та інтеграція з іншими інформаційними системами.

Ідентифікація та верифікація користувачів може здійснюватися різними методами, від традиційних RFID-карток і PIN-кодів до біометричних технологій, серед яких особливу увагу зараз привертає голосова ідентифікація.

У таблиці 1.1 показано чому у минулому найбільш поширеними були магнітні карти або PIN-коди, але з часом вони виявили ряд вразливостей: якщо карта загублена або вкрадена, зловмисник може легко отримати до неї доступ, а PIN-коди часто пишуться на папірцях або передаються іншій людині. Біометричні методи, включаючи відбитки пальців, сканування сітківки або розпізнавання обличчя, поступово набирають популярність через більш високий рівень безпеки і відсутність необхідності запам'ятовувати комбінацію коду.

Однак у складних умовах, таких як на виробництві з брудним обладнанням або в місцях зі змінним освітленням і температурою, системи, які покладаються на відбиток пальця або зчитування зображень, іноді можуть мати більш високу швидкість збоїв доступу.

Таблиця 1.1 – Порівняльна характеристика технологій контролю доступу

Критерій	RFID-картки	PIN-коди	Відбитки пальців	Голосовий пароль	Розпізнавання обличчя
Основні вразливості	Клонування карток	Підгляд/забуття коду	Фальшиві відбитки	Replay-атаки, deepfake	Фотографії, маски
Вартість	Низька $5/картка	Дуже низька	Середня – $100/сканер	Середня – $100/мікро	Висока $200-300
Швидкість	~0.2–0.5 с	~1–2 с	~0.5–1 с	~1–2 с	~0.5–1 с
Зручність	Висока	Середня	Висока	Висока	Висока
Рівень безпек	Стабільний	Низький–Середні	Середній–Високий	Низький–Середній	Середній–Високий
Необхідність обслуговування	Мінімум	Немає	Періодична калібровка	Періодичне оновлення	Час від часу калібруваня
Масштабованість	Висока	Дуже висока	Середня	Середня	Середня

У зв'язку з цим голосова аутентифікація є своєрідним компромісом між простотою використання і надійністю. На відміну від відбитків пальців або сканування обличчя, які вимагають додаткових спеціальних пристроїв (оптичні або ультразвукові датчики, високоякісні камери), голосовий метод в основному реалізується за допомогою звичайного мікрофона і програмного забезпечення. Однак у порівнянні з картками або PIN-кодами такий підхід дозволяє створити унікальний «голосовий профіль» користувача, який враховує безліч акустичних особливостей: тембр, інтонацію, ритм мови, паузи між словами та інші параметри. Такий профіль значно ускладнює підробку: щоб успішно обійти систему, зловмиснику потрібно не тільки отримати запис голосу, але і

синтезувати його з тими ж властивостями, що створює код, максимально наближений до живої мови. Крім того, сучасні алгоритми аналізу дозволяють оцінити «живий» характер аудіо, визначивши, чи був він записаний в реальному часі або відтворений з запису.

На додаток до чисто технічних переваг, голосова аутентифікація забезпечує високу гнучкість інтеграції. Наприклад, в організації його можна використовувати не тільки для відкриття фізичних дверей або турнікетів, але і для доступу до корпоративних додатків, автоматизованих терміналів і т.д. Оскільки алгоритми розпізнавання голосу можуть передаватися практично на будь-який пристрій з мікрофоном і достатньою обчислювальною потужністю, це дає можливість побудувати єдину платформу аутентифікації: від смартфонів до стаціонарних терміналів. Такий підхід значно спрощує адміністрування системи та мінімізує витрати, оскільки для кожного виду доступу немає необхідності впроваджувати різні методи ідентифікації.

Однак важливо пам'ятати, що голос як біометричний параметр схильний до значних зовнішніх впливів: зміни зовнішнього шуму, здоров'я користувача, емоційного стану, навіть помітні зміни температури навколишнього середовища можуть погіршити точність розпізнавання. Тому ключові вимоги до якості обладнання, а також адаптивності алгоритмів. При відсутності ретельної конфігурації система може проявляти підвищену кількість помилкових збоїв доступу, що негативно позначиться на комфорті користувачів і загальній репутації рішення.

В результаті проектування будь-якої СКД вимагає комплексного підходу: поєднання апаратного забезпечення з програмними модулями, здатними працювати в різних умовах, постійного контролю технічного стану компонентів, а також навчання персоналу, що експлуатує систему. Приклади показують, що без регулярних оновлень програмного забезпечення, налаштувань алгоритмів і періодичних перевірок апаратного забезпечення навіть найсучасніші системи можуть стати вразливими через нові типи атак або зміни умов експлуатації. У

зв'язку з цим одним з найважливіших елементів є забезпечення адекватної інтеграції між компонентами СКД та іншими підсистемами безпеки, зокрема системами відеоспостереження, охоронної сигналізації та мережевого моніторингу. Об'єднання даних з різних джерел дозволяє побудувати більш точну картину ризиків, швидко реагувати на інциденти і автоматизувати процеси блокування доступу в разі підозрілих дій.

Технічні параметри окремих елементів СКД, такі як продуктивність контролера, пропускна здатність мережі, обсяг доступної пам'яті для зберігання ідентифікаційної бази даних, безпосередньо впливають на масштабованість і ефективність рішення. Якщо мова Для великих підприємств з сотнями або тисячами точок доступу потрібна розподілена архітектура, яка забезпечує як централізовану, так і віддалену обробку даних вузлами, що дозволяє безперервно працювати навіть у разі збою одного з місць. Таким чином, при виборі технічної платформи необхідно враховувати не тільки поточні обсяги користувачів, але і перспективи розширення, прогнозоване збільшення навантаження і можливість інтеграції з новими типами датчиків і каналів зв'язку.

Можна сказати, що сучасна СКД - це не тільки набір пристроїв і програмних модулів, але інтегрована екосистема, що включає фізичні обмежувачі, різні методи ідентифікації, мережеву інфраструктуру, системи моніторингу та аналітики, а також добре сформовані процедурні заходи і політики.

Обрані технології і правильно налаштовані алгоритми визначають не тільки рівень безпеки, але і зручність для кінцевих користувачів: швидкість проходження, можливість масштабування і адаптації до змін навколишнього середовища. А голосова аутентифікація, завдяки своїм біометричним характеристикам і відносній простоті реалізації, сьогодні знаходиться в авангарді розробки сучасних СКУД, пропонуючи збалансоване рішення між зручністю і безпекою.

1.2 Історичний розвиток голосових біометричних систем

За останні п'ятнадцять років голосова біометрія пройшла довгий шлях від експериментальних досліджень в лабораторіях до масових комерційних рішень, які зараз використовуються в банківських колл-центрах, мобільних додатках і системах «розумного будинку». На початку 2000-х років голосові системи базувалися переважно на статистичних моделях, таких як моделі гауссових сумішей (GMM) і марковські процеси (HMM). Ці підходи базувалися на розподілі акустичних ознак у низьковимірному просторі - найчастіше це були частотні цепстральні коефіцієнти Мела (MFCC) - і дозволяли з достатньою точністю відрізняти одного диктора від іншого, але вимагали великої кількості високоякісних зразків і добре контрольованих умов запису.

У середині 2010-х ситуація змінилася з появою концепції і-вектора. Замість того, щоб аналізувати кожен кадр аудіо окремо, і-векторні моделі виділяли з усього запису компактний вектор фіксованої довжини, який містив інформацію про індивідуальні характеристики голосу і фонового шуму. Таке рішення значно скоротило обчислювальні витрати і дозволило інтегрувати голосові рішення в мобільні пристрої з обмеженими ресурсами, але воно також вимагало ретельного калібрування до різних акустичних умов і часто давало нестабільні результати в присутності сильного шуму або різких змін в голосі користувача.

Наступним кроком стала поява нейромережевих репрезентацій - так званих d-векторних, а згодом і x-векторних архітектур. Замість використання класичних статистичних підходів, глибокі нейронні мережі були навчені витягувати компактне представлення з фіксованої довжини мовленнєвого сегмента, яке характеризувало особистість мовця незалежно від тексту, що вимовляється. У порівнянні з і-векторним підходом, x-векторні моделі продемонстрували вищу завадостійкість і певну інваріантність до акустичних змін, а також не вимагали строго контрольованих умов при записі навчальної бази. Всі сучасні комерційні

рішення голосової біометрії сьогодні покладаються на ці глибокі архітектури, часто доповнені механізмами виявлення живості, які допомагають відфільтрувати атаки повторного відтворення або синтезовані голоси.

Разом з удосконаленням алгоритмів змінювалися і платформи: ще в середині 2010-х більшість систем розгорталися на серверних системах з потужними процесорами, а до кінця десятиліття стали популярними легкі клієнтські рішення для смартфонів і одноплатних комп'ютерів (Raspberry Pi), з динамічними фреймворками машинного навчання, що дозволяють виконувати моделі виводу безпосередньо на пристрої. Це відкрило шлях до використання голосової автентифікації в сценаріях Інтернету речей, коли смартфон або смарт-колонка виконує первинну перевірку без хмари.

Потужний поштовх також дав розвиток стандартів та відкритих наборів даних: у 2015-2018 роках на конференціях NIST почали регулярно проводити 평가 (SRE - Speaker Recognition Evaluation), які об'єднували зусилля різних лабораторій та компаній у боротьбі за найкращий алгоритм. Це призвело до швидкої еволюції методів, оскільки переможці SRE швидко публікували описи своїх моделей і відкривали код.

Важливим етапом також стала поява мобільних SDK та API від великих гравців: Google, Apple, Microsoft та Amazon інтегрували сервіси голосової біометрії у свої хмарні платформи. В результаті навіть невеликі стартапи змогли підключити голосову автентифікацію без значних інвестицій у власні дослідницькі центри.

Разом із підвищенням точності зросла увага до конфіденційності та безпеки. Якщо на початку 2000-х років голосові підписи фактично зберігалися в оригінальному вигляді, то сьогодні моделі працюють із зашифрованими векторними представленнями, а оригінальні аудіозаписи не зберігаються. З'явилися гомоморфне шифрування і механізми федеративного навчання, які дозволяють навчати загальну модель без передачі даних користувача на сервер.

Таким чином, за останні півтора десятиліття голосова біометрія перетворилася з експериментального прототипу на зрілу технологію з високою завадостійкістю, відносно низькими обчислювальними вимогами і складними механізмами захисту приватності. Сьогодні основними тенденціями є подальший розвиток адаптивної детекції брехні, спрощення реєстрації користувачів за допомогою фраз-«будильників», інтеграція зі схемами багатофакторної безпеки, а також все більше проникнення в портативні пристрої та IoT-пристрої.

1.3 Оцінка ризиків для систем захисту конфіденційних даних

У системах голосової аутентифікації голос користувача стає ключовим біометричним елементом, але саме тому йому загрожують різноманітні атаки. Найпростішою формою компромісу є запис мови - зловмисник може непомітно записати контрольну фразу під час розмови або в публічному просторі, а потім відтворити цей запис, щоб обійти систему. Навіть якщо системи включають в себе базову перевірку «живості» голосу - аналіз спектрів або виявлення природних нюансів дихання і інтонації - висока якість запису і відповідні пристрої відтворення можуть подолати такі захисні механізми, якщо вони не налаштовані ретельно. Нейромережі здатні генерувати голоси, схожі на справжні, що створює ризики використання фейкових фраз для обходу захисту. Поява таких підробок змушує розробників удосконалювати алгоритми виявлення артефактів синтетичних звуків, а також інтегрувати багатофакторну аутентифікацію, коли до голосового пароля додаються зовнішні маркери - наприклад, поєднання голосу з одноразовим кодом або аналіз поведінкових патернів користувача при введенні фрази [21] [22] [23].

Не менш небезпечним є використання методів соціальної інженерії: шахраї телефонують користувачеві під виглядом техпідтримки або співробітнику банку

і просять контрольну фразу, мотивуючи її «перевіркою безпеки». Людина, яка не підозрює зловмисних намірів, може вимовити свою фразу, яка автоматично відкриває доступ до захищених ресурсів. Щоб мінімізувати цей ризик, необхідно реалізувати навчальні програми для користувачів, уточнити правила безпеки і надати системні попередження, які змусять людину задуматися перед тим, як вимовляти секретну фразу на прохання сторонньої особи [24].

Далі якщо продовжувати про вразливість, то вона стосується безпеки передачі голосових даних від записуючого пристрою до модуля обробки: якщо аудіопотік передається по незашифрованому каналу, зловмисник може перехопити його, зберегти і використовувати пізніше. Використання сучасних захищених протоколів зв'язку TLS, аутентифікації кінцевих точок і шифрування криптографічних даних допомагає уникнути прямих перехоплень і знизити ймовірність успішного втручання в процес відправки голосових записів.

Крім того, якщо стороннім особам доступні навчальні зразки для систем розпізнавання, їх можна цілеспрямовано «отруїти» - внести начебто невидимі зміни, щоб модель почала неправильно класифікувати голоси, пропустити шахраїв або, навпаки, відкинути законні зразки. У таких випадках багаторівнева перевірка даних, контрольні суми та регулярні перевірки навчальних наборів використовуються для запобігання модифікації критичних елементів [25] [26].

Також до очевидних цифрових загроз, голосова аутентифікація стикається з ризиком побічних каналів: аналіз електромагнітного випромінювання мікроконтролера або коливання енергоспоживання може дати зловмисникам інформацію про обробку голосових шаблонів і навіть допомогти реконструювати частину аудіоданих. Для зниження цих ризиків використовуються захищені апаратні елементи , екранування плати і шифрування внутрішніх петель, що ускладнює бічний моніторинг пристрою [27].

Також важливо враховувати операційні умови, які можуть призвести до помилкових спрацьовувань або невдалих спроб доступу. Наприклад, голос

користувача змінюється під впливом хвороби, гучного фону або емоційного стану, що значно погіршує точність розпізнавання.

Сильний фоновий шум Для цього системи повинні використовувати адаптивні фільтри шумозаглушення, аналізувати співвідношення сигнал/шум в режимі реального часу і забезпечувати резервні механізми - наприклад, тимчасовий перехід на альтернативний метод перевірки або запит на додатковий код, якщо рівень довіри до розпізнавання низький [28].

Атаки мережі “man-in-the-Middle” не менш небезпечні, коли зловмисник, контролюючи мережевий трафік, може замінити або змінити аудіопотоки на льоту. Навіть при зашифрованому зв'язку можна спробувати замінити сертифікати, якщо пристрої не мають сильної аутентифікації. Для захисту від цього необхідно постійно оновлювати прошивку пристроїв, використовувати строгі політики перевірки сертифікатів і здійснювати двосторонню аутентифікацію кінцевих пристроїв [29].

Юридичні та етичні аспекти обробки голосу вимагають окремого розгляду. Відповідно до Законів України «Про захист персональних даних» та «Про інформацію» голос належить до категорії чутливої біометричної інформації, обробка якої можлива лише за умови отримання користувачем письмової або електронної згоди.

Зберігання аудіозаписів , або спектральних голосових шаблонів вимагає суворого шифрування, обмеженого доступу та чіткої політики збереження, яка запобігає використанню цих даних більше, ніж необхідно для аутентифікації. З етичної точки зору користувачі повинні бути поінформовані про мету збору своїх голосових даних, термін їх зберігання та осіб з доступом. Не менш важливо надати можливість відмовитися від біометричної аутентифікації без втрати права доступу, надаючи альтернативні методи, такі як одноразовий код або фізичний ключ [30].

Підвищення стійкості системи можливе лише при одночасній реалізації технічних фільтрів, юридичних гарантій та користувацьких інструкцій.

Поєднання живого виявлення, багатофакторної перевірки, захищених каналів передачі, надійного шифрування даних, захищених апаратних платформ і прозорої політики обробки персональних даних дозволяє мінімізувати ризики несанкціонованого доступу і гарантувати довгострокову надійність голосової аутентифікації в різних додатках.

1.4 Різні підходи і методи , їх недоліки і переваги

Існує дві основні парадигми голосової автентифікації: текстово-залежна та текстово-незалежна голосова перевірка. У текстово-залежних системах користувач пройде фіксовану або раніше відому фразу, яка повинна бути вимовлена саме в тому вигляді і інтонації, який використовувався при реєстрації шаблону. З одного боку, це дозволяє значно спростити алгоритм порівняння: система порівнює отриманий аудіозапис зі стандартом, перевіряючи збіг спектрально-часових параметрів конкретного набору слів. Однак, з іншого боку, текстово-залежний підхід погано працює в разі зміни голосу користувача через хворобу або шумовий фон низької якості - невелика різниця у вимові може призвести до помилкової відмови. Крім того, фіксована фраза створює вразливість для відтворення атак: досить отримати запис самої команди, щоб відтворити її перед мікрофоном.

Тексто-незалежні системи замість цього аналізують голос незалежно від обраної фрази, спираючись на унікальні акустичні особливості тембру, дихання, ритму і навіть мікроколекції в частотних характеристиках голосу. Щоб тренувати модель таким чином, потрібно набагато більше даних: сотні і тисячі зразків, в яких користувач говорить довільні тексти, читає нові речення і говорить слова в різних контекстах. Головною перевагою такого підходу є підвищена стійкість до перехоплень одного конкретного запису: зловмиснику доведеться мати під рукою реальну поведінку голосового користувача або

збирати великий масив високоякісних записів для синтезу, щоб обдурити систему. У той же час, якщо умови запису голосу істотно відрізняються (наприклад, в шумному середовищі), модель може неправильно класифікувати законний голос як шахрайський або, навпаки, пропустити чужий. Через це текстово-незалежні алгоритми вимагають складних адаптивних механізмів шумозаглушення і тонкої настройки для кожного нового користувача. [32]

Сучасні комерційні рішення все частіше використовують методи глибокого навчання та статистичні вектори для представлення голосу. Наприклад, алгоритми, засновані на i-vector, зберігають характеристики голосу як нерухомі вектори низької розмірності, які відображають основні змінні джерела звуку - мовні особливості, фонові фактори, індивідуальні варіації. При введенні тестового голосового запису система обчислює його і-вектор і порівнює його з опорними векторами користувачів бази даних за допомогою відстані косинусної подібності або більш складних метрик. Такий підхід виявився досить ефективним в «нечистих» умовах, коли фон забруднений, оскільки моделі i-vector враховують глобальні статистичні властивості запису. Однак ці алгоритми досить чутливі до якості вибірки: якщо дані для навчання збиралися тільки в спокійних студійних умовах, система може показувати більш слабкі результати в реальному середовищі, де присутній широкий спектр шуму [33].

У свою чергу, x-векторна технологія, яка базується на нейронних мережах з глибокими шарами, покращує ситуацію, формуючи голосові представлення безпосередньо на рівні аудіосегменту. X-векторна модель «вчиться» розпізнавати динамік незалежно від лексичного змісту, вибудовуючи стек нейронних мереж із затримкою часу, що дозволяють враховувати як контекстні фрагменти мови, так і звукові особливості в різних часових масштабах.

Ці вектори, а потім метричне порівняння, показують високу стійкість до змін в акустичному середовищі, швидко оновлюються для нових користувачів і адаптуються для оновлення бази.

Основна слабкість x-векторного підходу полягає в необхідності значних обчислювальних ресурсів під час навчання і широкого спектру навчальних даних модель може показати зниження точності, якщо база голосів не відображає всі можливі варіанти звуку або шумового фону [34].

Останнім часом зростає попит на динамічні системи голосових паролів, коли текстова фраза генерується самою системою у вигляді випадкової комбінації слів або простої фрази безпосередньо перед аутентифікацією. Цей підхід ускладнює повтор атак і атак за допомогою попередньо синтезованого голосу, оскільки для успіху зловмисник повинен або неймовірно швидко налаштувати генератор, або мати вдосконалені інструменти синтезу в реальному часі, які зазвичай недоступні для широкої публіки. Однак динамічні паролі також ускладнюють завдання користувача: необхідно не тільки знати текст, але і правильно його читати, враховуючи інтонацію і чіткість вимови, що збільшує час аутентифікації і може викликати дискомфорт в стресових ситуаціях. Крім того, реалізація динаміки вимагає синхронізації між сервером і клієнтом, що ризикує уповільнити процес через затримки мережі або помилки в часі, особливо якщо платформа працює в автономному режимі з обмеженими ресурсами [35].

Для підвищення надійності будь-якого з цих підходів часто додаються компоненти виявлення життєздатності, які аналізують не тільки частотні характеристики голосу, але і пов'язані з цим особливості: зміни тиску повітря при видиху, характерні моделі вібрації, наявність рубок або інші незначні коливання, характерні виключно для живої мови. Якщо система виявляє відсутність природних біомеханічних сигналів (наприклад, аномально рівномірний фон без природних коливань), вона може заблокувати спробу розпізнавання, навіть якщо спектральний вектор «збігається» з посиланням. Такі методи збільшують бар'єр для простих записів, і, в той же час, викликають збільшення помилкових збоїв у разі гучного фону або використання неякісних аудіопристроїв, де алгоритм може «не бачити» достатньої кількості вказівок на реальний голос [36].

Ще однією перспективною практикою є поєднання голосового фактора з іншими функціями - двофакторною аутентифікацією, коли після позитивного розпізнавання голосу користувач повинен ввести одноразовий код з SMS, натиснути фізичну кнопку або підтвердити запит через додаток. Це дозволяє компенсувати слабкі Однак така конфігурація може знизити загальну зручність і уповільнити процес, що критично важливо в ситуаціях, коли потрібна оперативна реакція - наприклад, в медичних установах або при відкритті аварійних дверей.

На рисунку 1.1 показано схему порівняння методів безпеки за швидкістю і рівнем. І як на ньому видно Методи глибокого навчання самий ідеальний варіант , який можна включити до своєї системи доступу.

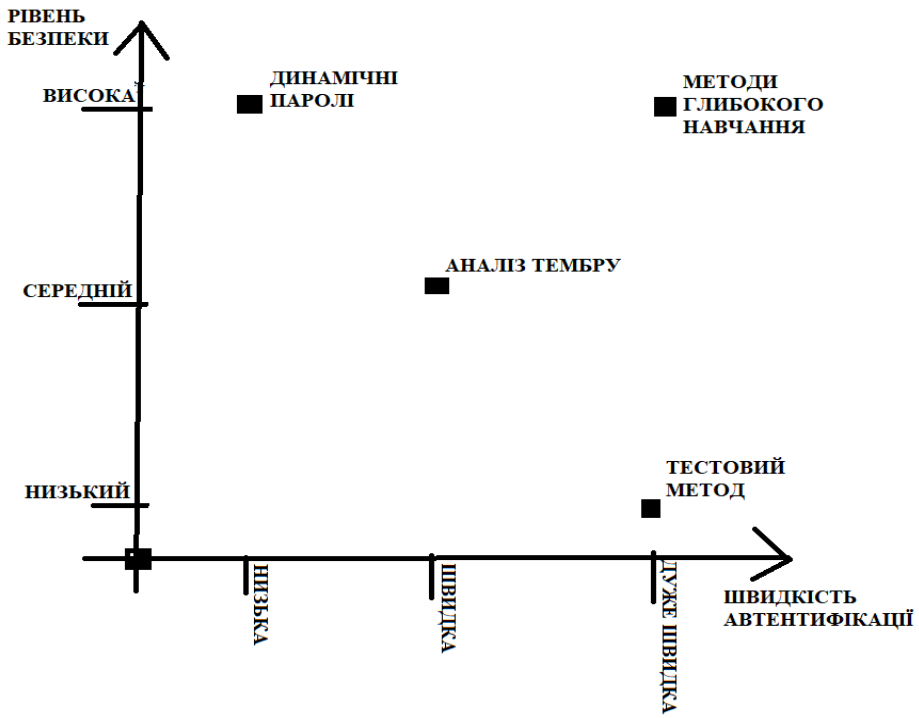


Рисунок 1.1 Схема порівняння методів за швидкістю та рівнем безпеки

Загалом, вибір оптимального підходу залежить від умов використання: якщо вам потрібен простий безконтактний метод в чистому середовищі з

мінімальним ризиком зовнішніх перешкод, то цілком підійде текстово-залежна система зі статичною фразою.

Замість цього для корпоративних рішень з високими вимогами безпеки і складних акустичних середовищ краще вибирати методи з і-векторними або х-векторними представленнями, доповнювати їх динамічними елементами і лівостороннім виявленням, а в критичних сценаріях додавати багатофакторний захист. Ця комбінована стратегія забезпечує баланс між швидкістю проходження, рівнем безпеки і стійкістю до мінливих умов, зберігаючи при цьому здатність масштабувати систему для задоволення зростаючих потреб.

1.5 Постановка задач

Метою цього дослідження є розробка надійної системи контролю доступу на основі голосового пароля, яка дозволяє користувачам обмежуватися певним приміщенням без використання складних біометричних технологій. У запропонованому підході акцент робиться не на ідентифікації людини за унікальними голосовими характеристиками, а на точному розпізнаванні заданого голосового пароля, який порівнюється з текстовим шаблоном, що зберігається в програмному коді. Тобто при реєстрації користувач задає певну фразу, а система запам'ятовує її тільки у вигляді текстового рядка, не формуючи голосових відбитків або збереження аудіозаписів. Завдяки цьому реалізація залишається відносно простою і економічною, але при цьому забезпечує достатній базовий рівень захисту - доступ отримають тільки ті, хто здатний чітко вимовляти фразу, що лежить в основі ключа.

На початковому етапі дослідження був проведений аналіз сучасних методів обробки мовних сигналів, а також ретельно підібраний набір інструментів, що забезпечують стабільне розпізнавання ключової фрази офлайн.

Система включає в себе мікроконтролер Arduino, який відповідає за апаратне забезпечення контролю доступу - зокрема, контролює електромеханічний замок або індикатори стану - і персональний комп'ютер зі скриптом Python, який захоплює і обробляє аудіо.

Голосові дані зчитуються за допомогою цифрового мікрофона, потім через бібліотеку PyAudio потік передається на рушій Vosk, який перетворює мовлення в текст. Якщо розпізнана фраза точно відповідає шаблону тексту, створеному під час реєстрації, Python посилає сигнал Arduino через послідовний порт у вигляді байта «1» - це призводить до розблокування блокування або включення відповідного індикатора. Якщо розпізнаний текст не збігається, то байти сигналу не надсилаються, і доступ залишається заблокованим.

Такий підхід не вимагає постійного підключення до Інтернету, оскільки весь процес розпізнавання відбувається локально на ПК, а біометричні дані користувача не зберігаються у вигляді аудіозаписів або інших форматів, що значно спрощує дотримання вимог захисту персональних даних.

Архітектурно система побудована так, що кожен елемент виконує свій чітко визначений набір функцій: мікроконтролер відповідає тільки за прийом і обробку команд, а комп'ютерна частина бере на себе весь обсяг розрахунків і роботу з голосовим двигуном. Це дозволяє легко масштабувати рішення - якщо є необхідність підключити кілька точок доступу, досить організувати відповідну кількість копій скрипта Python на різних комп'ютерах або міні-ПК і підключити їх до необхідних плат Arduino

Таким чином, вирішення проблеми полягає у створенні інтуїтивно зрозумілої та легко реалізованої системи контролю доступу, яка працює на основі простого текстового голосового пароля, забезпечує автономність роботи в автономному режимі і не вимагає складних ресурсів для реалізації, гарантуючи при цьому базовий захист від несанкціонованого проникнення.

2 ПРОЄКТУВАННЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ НА ОСНОВІ ГОЛОСОВОГО ПАРОЛЮ

2.1 Обгрунтування вибору системи керування

Обгрунтування вибору системи контролю доступу починається з бажання знайти максимально просте, практичне і в той же час економічно вигідне рішення, яке дозволить реалізувати голосову аутентифікацію без необхідності реалізації складних серверних інфраструктур і дорогих апаратних модулів. Вирішальним фактором став мікроконтролер Arduino Nano: він поєднує в собі низьку вартість, компактні розміри і достатню обчислювальну потужність для управління основними апаратними компонентами системи - електромагнітним замком, індикаторами стану і, при необхідності, сигналізацією. Завдяки вбудованому інтерфейсу USB Arduino Nano не вимагає зовнішніх перетворювачів рівня сигналу, що спрощує електропроводку і зменшує кількість точок потенційного виходу з ладу. Оскільки він має стабільний чотиримегагерцовий кристал і працює з напругою 5 В, підтримує стандартизований набір портів вводу-виводу (GPIO) і інтерфейс UART зі швидкістю до 115200 бод, Arduino Nano легко інтегрується з персональним комп'ютером для прийому команд через COM-порт і одночасно контролює виконавчі пристрої без зайвих затримок. Саме завдяки цій архітектурі, коли обчислювальні завдання орієнтовані на ПК, а Arduino відповідає за фізичну частину, забезпечується гнучка масштабованість: для кожної нової точки доступу достатньо підключити окремий Arduino Nano до нового комп'ютера або міні-ПК, а силова лінія і механізм управління замком залишаються стандартизованими [37].

Практичним доповненням до апаратної платформи є вибір голосового двигуна Vosk. Оскільки система призначена для роботи в автономному режимі, без постійного доступу до інтернету, Vosk з двигуном Kaldi став оптимальним варіантом: він може виконувати розпізнавання голосу без підключення до

віддалених серверів, що гарантує конфіденційність біометричних даних. Модель Vosk добре масштабована - від легких лінгвістичних пакетів для вузьких словників до більш об'ємних універсальних модифікацій, вона дозволяє підібрати оптимальні параметри для швидкості і точності розпізнавання. Використання Vosk зменшує затримки в обробці аудіопотоку: в типовій конфігурації офісного ПК з двоядерним процесором і 4 ГБ оперативної пам'яті час відгуку на команду не перевищує 200-300 мс, що цілком прийнятно для контролю доступу в режимі реального часу. Крім того, Vosk підтримує адаптивні словники і можливість додаткового навчання на власних зразках користувача, що особливо важливо, коли необхідно розгорнути систему в багатомовному середовищі або з конкретною термінологією. Це дозволяє уникнути додаткових витрат на хмарні сервіси та консультації зовнішніх розробників [38].

Аудіозапис здійснюється за допомогою цифрового мікрофона з можливістю вибору частоти дискретизації 16 кГц або 48 кГц і ємністю біт 16 біт. Мікрофон підключається до ПК через стандартний інтерфейс аудіокарти USB, який оцифровує звуковий сигнал і відправляє його на скрипт Python. У середовищі Windows або Linux використання бібліотеки PyAudio дозволяє будувати буферизовану систему отримання аудіопотоку з мінімальною затримкою, а також проводити попередню обробку, яка включає нормалізацію рівня сигналу, базове зниження шуму і зняття феритових перешкод. Якщо в приміщенні є постійні джерела шуму (наприклад, вентиляція або інше обладнання), алгоритми нормалізації та динамічної компенсації фону зменшують кількість помилкових негативних визнань. Крім того, можна інтегрувати додаткові програми зменшення шуму (наприклад, SpeexDSP), але для більшості офісних або лабораторних сценаріїв достатньо базової обробки PyAudio [39].

Передача команди з Python на Arduino відбувається через протокол UART за допомогою бібліотеки pySerial, яка забезпечує конфігурацію послідовного порту і передачу одного байта (наприклад, «1» для відкриття блокування і «0»

для скидання стану). Arduino Nano в прошивці на C++ налаштований на 9600 бод, що мінімізує затримку передачі до декількох мілісекунд. Після отримання байта контролер безпосередньо через транзисторний драйвер керує електромагнітним замком, витримуючи період часу, необхідний для повного відкриття або закриття, і генерує відповідні сигнали зворотного зв'язку - наприклад, включає світлодіод або передає зворотний байт Python у разі критичних помилок. Цей рівень взаємодії гарантує мінімізацію вразливості до мережевих збоїв: якщо з якихось причин скрипт Python припиняє роботу або USB-канал переривається, Arduino залишається в безпечному режимі, не відпускаючи блокування до стабільної команди, що підвищує загальну стабільність системи.

У процесі вибору альтернативних апаратних рішень розглядалися Raspberry Pi 3 з вбудованим аудіо інтерфейсом і одноплатними комп'ютерами на основі архітектури ARM (наприклад, Orange Pi або Odroid). Тим не менш, ці платформи вимагали додаткових витрат на зовнішні перетворювачі рівня GPIO, мають більшу площу установки і споживають значно більше енергії. Raspberry Pi, наприклад, при активному навантаженні вимагає до 1 А струму на напругу 5 В, читання та обробка аудіопотоку, тоді як Arduino Nano споживає не більше 50 мА, оскільки виконує лише командну передачу. Крім того, на Raspberry Pi потрібно регулярно оновлювати операційну систему, налаштовувати драйвери на аудіо, що призводить до додаткових тимчасових і фінансових витрат на підтримку. Враховуючи той факт, що в нашій системі обчислювальні завдання покладаються на стаціонарний ПК, а Arduino Nano виконує лише роль простої виконавчої одиниці, вибір на користь Arduino Nano цілком виправданий з огляду на принцип мінімізації складності і вартості.

Також велика увага приділяється питанням безпеки обробки даних. Оскільки голосові зразки не зберігаються у вигляді аудіофайлів між сеансами, а система працює безпосередньо з результатом розпізнавання (текстовим рядком), ймовірність витоку біометричної інформації зводиться до нуля. Arduino Nano не містить пам'яті для постійного зберігання паролів; текстовий шаблон фрази

зберігається в змінній скрипту Python, яка додатково захищена механізмом прав на файли операційної системи. При необхідності можна реалізувати шифрування посилок командою між Python і Arduino (наприклад, через простий алгоритм XOR) або додати механізм аутентифікації COM-портів, щоб унеможливити заміну команд із зовнішніх пристроїв.

Нарешті, обрані компоненти забезпечують гнучкість для подальшого масштабування. Якщо є необхідність додати багатофакторну аутентифікацію, досить розширити скрипт Python за допомогою модуля для відправки SMS-коду або інтегрувати апаратний RFID-рідер в той же Arduino. Оскільки інтерфейс між мікроконтролером і ПК стандартизований, додавання нових датчиків або виходів відбувається без змін в базовій архітектурі. Таким чином, поєднання Arduino Nano і Vosk як голосового двигуна забезпечує оптимальний баланс між вартістю, швидкістю реалізації, автономністю і безпекою, що відповідає вимогам завдання створення доступної і надійної системи контролю доступу.

2.2 Алгоритм взаємодії Python-скрипта з Arduino через COM-порт

Для забезпечення надійної взаємодії між програмною частиною в Python і апаратним модулем виконання на базі Arduino Nano був розроблений спеціальний тестовий скрипт, метою якого є перевірка працездатності послідовного з'єднання і правильності обміну командами. У центрі тесту знаходиться простий цикл взаємодії: відкрийте порт, дайте час на ініціалізацію дошки, допитайте користувача, надішліть символ і нарешті спостерігайте за зворотним зв'язком у вигляді зміни стану світлодіода. Така проста програма дозволяє локалізувати помилки на ранній стадії і переконатися, що апаратні та програмні компоненти проекту говорять на одній когерентній мові байтів.

Першим кроком у сценарії Python є імпорт двох критичних модулів: послідовного з пакету “PySerial”, який забезпечує функціональність відкриття та

управління нашим портом, а також часу, який використовується для затримок. Відразу після цього послідовне з'єднання ініціалізується серійною конструкцією. Параметри порту - ім'я пристрою і обмінний курс - повинні строго відповідати налаштуванням в коді Arduino, де така ж швидкість встановлюється в функції “setup ()” за допомогою Serial.begin. Після відкриття порту обов'язкова коротка пауза, оскільки кожен раз, коли встановлюється послідовне з'єднання, Arduino Nano перезавантажується, і без затримок перший байт команди може бути втрачений або плата може не реагувати на нього. Дві секунди затримки цілком достатні для стабілізації роботи контролера і стали отримувати-відправляти дані.

Після встановлення з'єднання користувач бачить повідомлення про успіх в консолі. Потім програма входить в нескінченний цикл, де вона чекає, поки оператор не скаже потрібну фразу. Якщо ви говорите правильно, то код відправляє один байт через USB-кабель зі значенням 0x31 - код ASCII символу 1. Відповідно до розробленої частини, отримання цього символу в Arduino читається як команда для включення індикатора, для встановлення виходу D13. У разі неправильного паролю скрипт посилає байт b'0', і Arduino відповідає попередньо записаним алгоритмом: наприклад, виконати серію мигань або просто погасити світлодіод, якщо він був включений раніше. При введенні будь-якої іншої послідовності символів програма виводить на консоль попередження про неправильне введення і продовжує цикл, чекаючи нової команди. Виклик “ser.close” в цьому коді необов'язковий в межах одного циклу, оскільки порт залишається відкритим, поки програма не закриється. Однак після закінчення обробки кожної команди ви можете закрити порт і знову відкрити його в наступній ітерації - це залежить від конкретних вимог до зв'язку та надійності.

Ключовим аспектом цієї реалізації є те, що початковий тест каналу зв'язку, таким чином, проводиться, перш ніж приступити до автоматизованої частини голосової аутентифікації. Якщо світлодіод успішно реагує на команди b'1'і b'0', це вказує на те, що драйвер CH340 встановлений, COM-порт обраний правильно, жодна інша програма не блокує його, а скрипт Python і Arduino використовують

той же обмінний курс. Важливо пам'ятати, що одночасний доступ до порту двома процесами призведе до помилок “Доступ заборонено”, тому перед тестом потрібно закрити всі непотрібні послідовні інтерфейси.

Цей простий ручний тест значно полегшує стадію налагодження проекту: підтверджує апаратне забезпечення (правильне підключення світлодіода, продуктивність плати, стабільність послідовного інтерфейсу) і програмний рівень (правильне розпізнавання вводу користувача, генерація і відправка байтів). У разі помилки оператор може по черзі перевірити кожен компонент: чи відкритий порт в диспетчері пристроїв, чи правильна швидкість в налаштуваннях, чи потрібна затримка після відкриття порту, чи дійсно скрипт відправляє дані, і, нарешті, чи отримує їх Arduino і виконує заплановану дію зі світлодіодом.

Цей підхід розподіляє обов’язки двох компонентів: на боці Python здійснюється складна обробка аудіосигналу, розпізнавання мови та ухвалення рішення щодо автентифікації, тоді як Arduino відповідає виключно за інтерпретацію прийнятого байта і керування апаратурою. Серійний протокол, що працює на одному байті даних, забезпечує надійну передачу без зайвих накладних витрат. Буферизація PySerial і внутрішні тайм-аути Arduino запобігають втраті даних навіть за інтенсивного потоку команд.

Впровадження голосової автентифікації цього типу дозволяє масштабувати систему без залучення зовнішніх серверів чи інтернет-ресурсів. Модель Vosk працює локально, тож усі дані залишаються у межах пристрою, що підвищує безпеку і знижує залежність від мережевих збоїв. А бібліотека PyAudio забезпечує реальний час обробки звуку. Якщо знадобиться розширити функціонал, досить змінити лише Python-скрипт: додати підтримку декількох паролів, обмежити кількість спроб, вести журнал спроб або навіть реалізувати механізми виявлення глибоких підробок голосу. Arduino-скетч, своєю чергою, можна доповнити керуванням реле замка, сиренами чи іншими виконавчими пристроями — усе це можливе завдяки простому серійному інтерфейсу, який

легко інтегрується в будь-яку апаратну платформу. Ця комбінована архітектура демонструє, як витончений програмний аналіз можна поєднати з економним та надійним апаратним контролером, створюючи систему контролю доступу на основі голосового пароля, що не потребує складної інфраструктури[40].

Алгоритм, зображений на рисунку 2.3, демонструє послідовність дій під час реалізації голосової автентифікації для системи контролю доступу на базі Arduino. На початковому етапі відбувається ініціалізація аудіосистеми і підключення моделі Vosk. Система знаходиться в режимі очікування, аналізуючи вхідний аудіосигнал і виконуючи розпізнавання сказаного тексту. Отриманий текст порівнюється із заздалегідь визначеною ключовою фразою. У разі успішного збігу система надсилає команду мікроконтролеру Arduino, який надає доступ. Якщо фраза не збігається з очікуваною, система знову очікує новий аудіо сигнал, поки надається живлення.

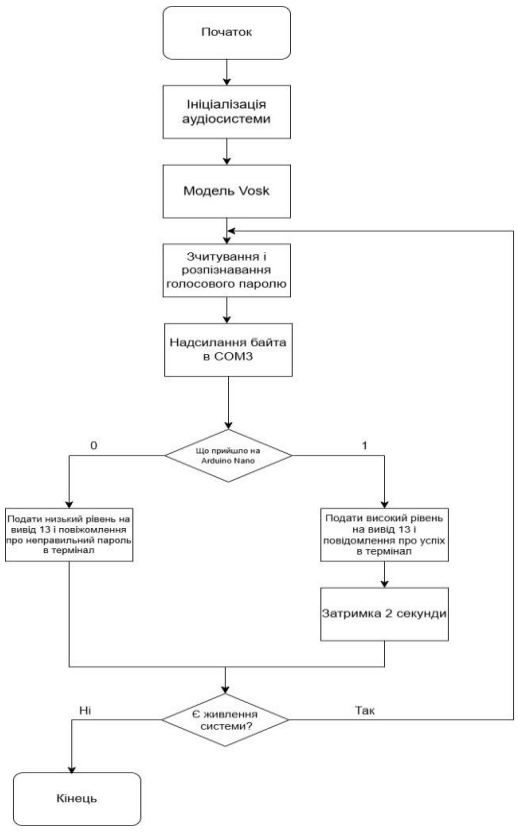


Рисунок 2.3 Алгоритм роботи системи контролю доступу на основі голосового паролю

2.3 Політика безпеки системи доступу

Політика безпеки системи контролю доступу на основі голосового пароля побудована з урахуванням вимог конфіденційності, цілісності та доступності інформації, а також принципів мінімізації ризиків при роботі в офлайн-середовищі. Суть системи полягає в локальному розпізнаванні голосової фрази за допомогою движка Vosk на персональному комп'ютері та передачі одного байта команди «дозволити» або «заборонити» через інтерфейс USB-Serial на мікроконтролер Arduino Nano, який відповідає за управління електромеханічним замком. Завдяки виключно локальному характеру розпізнавання немає необхідності передавати аудіопотік в хмару або зберігати оригінальні записи, що значно знижує ймовірність витоку чутливих біометричних даних.

Під час запуску програми переконайтеся, що директорія зі скриптом Python і папка з мовною моделлю мають обмежені права доступу до файлів тільки для користувача або групи адміністраторів. Використання стандартного COM-порту під певним логіном мінімізує ризик конфліктів з іншими процесами і виключає можливість запуску скрипта сторонніми особами. Для додаткового рівня захисту передача байтів між ПК і Arduino може бути просто зашифрована за допомогою матриць XOR, що запобігає підслуховуванню команд на фізичному рівні.

Методологія розпізнавання передбачає попередню обробку аудіо на рівні бібліотеки PyAudio з базовим шумозаглушенням і нормалізацією рівня сигналу. Параметри буферизації та частота дискретизації підібрані таким чином, щоб забезпечити баланс між швидкістю відгуку (до 200 мс до відображення результату) і стабільністю в умовах наявності базового фонового шуму. Відсутність проміжного сховища даних знижує навантаження на файлову систему і виключає ризик DOS-атак на диск комп'ютера.

Відповідальність за безперебійну роботу всіх компонентів лежить на службі технічної підтримки, яка стежить за стабільністю COM-з'єднання,

перевіряє стан USB-кабелю і драйвера CH340, а також контролює своєчасне оновлення мовних моделей Vosk. У разі виявлення численних невдалих спроб авторизації система автоматично блокує подальші дзвінки на час, достатній для перезавантаження скрипту або перевірки журналу подій. Всі успішні та неуспішні спроби доступу фіксуються в закритому текстовому журналі, доступ до якого мають лише відповідальні особи. Такий підхід дозволяє оперативно розслідувати інциденти та відстежувати підозрілі серії збоїв.

Юридична складова відповідає Закону України «Про захист персональних даних» і передбачає зберігання єдиного текстового шаблону в зашифрованому вигляді з мінімальним терміном зберігання, що перевищує фактичний період використання системи не більше ніж на місяць. Копії будь-яких журналів з біометричними даними (за наявності) підлягають знищенню після завершення щоквартального аудиту або на вимогу контролюючих органів.

Незалежно від технічних засобів, система спроектована таким чином, щоб залишатися в безпечному стані в разі виходу з ладу будь-якого вузла: до отримання позитивної голосової верифікації замок залишається закритим, оскільки відсутність команди «1» інтерпретується як відмова в доступі. Апаратне забезпечення, як плата Arduino Nano з підключеним замком і світлодіодом не може ініціювати відкриття без зовнішньої команди з комп'ютера, що виключає ризик неконтрольованої роботи через збій прошивки. Після оновлення будь-якого компонента системи перед введенням в експлуатацію обов'язково проводиться контрольна перевірка за участю кількох фахівців, а логи попередніх сесій аналізуються на предмет аномалій.

Завдяки використанню простого апаратного забезпечення без складних серверних рішень, локальному обходу будь-яких зовнішніх мереж, а також комплексним заходам захисту ключових компонентів, дана дипломна робота забезпечує комплексний захист інформації та фізичного доступу без порушення робочих процесів і з мінімальними витратами на впровадження та підтримку. Така політика безпеки гарантує довгострокову стабільність і надійність системи

в офісі, лабораторії або навчальному закладі, відповідаючи при цьому нормативним вимогам і кращим практикам сучасної кібербезпеки.

2.4 Обґрунтування обраних компонентів

Вибір саме тих компонентів, які лягли в основу прототипу голосової системи контролю доступу, був обумовлений комплексом практичних, технічних та економічних чинників. Центральним елементом апаратної частини стала плата Arduino Nano, яка завдяки своїй компактності та надійності забезпечила оптимальний баланс між доступністю і функціональністю. З одного боку, Arduino Nano має достатньо цифрових та аналогових входів-виходів для підключення світлодіодів, реле чи інших виконавчих елементів, а з іншого — підтримує стандартну швидкість серійного інтерфейсу, необхідну для взаємодії з Python через бібліотеку pySerial. Невисока вартість цієї плати дозволяє легко масштабувати систему, а велике ком'юніті користувачів дає змогу швидко знаходити відповіді на технічні питання. Обрання Arduino зменшує бар'єр входження для розробників і дає змогу зосередитися на розробці логіки розпізнавання голосу, не витрачаючи час на налаштування низькорівневого апаратного коду.

Не менш важливим компонентом стала система акустичного захоплення – мікрофон. Для прототипу використовувався конденсаторний USB-мікрофон з інтегрованим передпідсилювачем, який забезпечив стабільний монофонічний сигнал на рівні 16-бітного кодування без додаткових перешкод. Такий вибір пояснюється декількома чинниками: по-перше, USB-мікрофон можна легко підключити до будь-якого комп'ютера чи міні-ПК без потреби у зовнішніх аудіоінтерфейсах; по-друге, вбудований інтерфейс забезпечує правильне узгодження рівнів сигналу, що позбавляє необхідності у ручному калібруванні мікрофона та гарантує відтворення звуку з мінімальними спотвореннями; по-

третє, форм-фактор USB робить мікрофон портативним і зручним для розміщення в обмежених просторах, що важливо для продуманого монтажу системи на дверному отворі чи турнікеті.

Важливою складовою програмної частини стала бібліотека Vosk, вибір якої був зумовлений поєднанням точності та автономності. Модель Vosk для української мови працює локально на пристрої без необхідності звернення до хмарних сервісів, що забезпечує високий рівень безпеки біометричних даних та відсутність залежності від якості інтернет-зв'язку. Протестовані альтернативи, такі як Google Speech-to-Text чи Microsoft Azure, хоч і демонстрували доволі високу точність розпізнавання, потребували онлайн-зв'язку та мали обмеження безкоштовного використання. Натомість Vosk, розповсюджуючись на відкритій ліцензії, дозволяє безкоштовно інтегрувати потужні алгоритми KaldiRecognizer і адаптувати їх під різні середовища без додаткових витрат. Крім того, підтримка української мови на прийнятному рівні точності є важливою умовою для локального ринку, а гнучке налаштування моделі через зовнішні конфігураційні файли дає змогу оперативно змінювати параметри чутливості чи словниковий запас.

Для роботи з аудіопотоком обрана бібліотека PyAudio. Досвід показав, що вона дає змогу оперувати звуковим потоком у реальному часі з мінімальними затримками. PyAudio базується на PortAudio, що гарантує кросплатформеність і стабільність роботи на Windows, Linux та macOS. Ця бібліотека дозволяє за лічені рядки коду ініціювати захоплення аудіо з мікрофона, обирати формат даних, розмір буфера та частоту дискретизації. Важливо, що PyAudio підтримує як blocking, так і non-blocking режими зчитування, що дає можливість гнучко налаштувати програмний цикл для того, щоб розподіляти обчислювальні ресурси між захопленням даних та їх обробкою. Альтернативні підходи, наприклад використання sounddevice або власного C-API, виявилися більш складними у встановленні або менш дружніми до новачків, тому для цілей

прототипу PyAudio став «золотою серединою» — він досить простий у налаштуванні і водночас потужний.

Ключовим елементом, що забезпечує зв'язок між Python і Arduino, стала бібліотека pySerial. Вона спрощує роботу з різними серійними портами, надаючи уніфікований інтерфейс для відкриття порту, налаштування швидкості передачі, контролю буфера та обміну байтами. У ході тестів показано, що pySerial коректно працює навіть на нестабільних USB-Serial конвертерах з різними чіпами (CH340, FTDI, Atmega16U2) і легко інтегрується в існуючі скрипти без необхідності написання низькорівневого коду. Бібліотека підтримує можливість роботи з тайм-аути, зчитуванням у потоковому режимі та виключає необхідність ручного опитування буфера — достатньо викликати read() чи write() для обміну даними. Це забезпечує мінімальну вірогідність блокувань і дає змогу одразу реагувати на команди користувача без додаткового коду.

Взаємодія між цими компонентами організована так, щоб максимально використати їхні сильні сторони і мінімізувати недоліки. Arduino, володіючи обмеженими ресурсами, виконує лише найпростіші завдання: зміна стану виходу по команді, отриманій через серійний порт. Вся важка праця з аналізу голосу лежить на Python-скрипті з Vosk та PyAudio, що дозволяє не перевантажувати мікроконтролер і створює чітке розмежування зони відповідальності. Така архітектура спрощує відлагодження та подальший розвиток: модернізація логіки розпізнавання не потребує жодних змін у скетчі Arduino, а апаратні модулі можна замінювати чи додавати без внесення правок у голосову частину.

Економічна складова також відіграла не останню роль у обґрунтуванні компонентів. Arduino Nano та USB-мікрофон коштують у десятки разів менше, ніж готові промислові контролери доступу з вбудованими біометричними модулями. Vosk та PyAudio — це безкоштовне ПЗ з відкритим кодом, а pySerial майже не має альтернатив у галузі Python. Вибір цих компонентів дозволив створити прототип з мінімальними витратами, зберігши при цьому гнучкість і можливість розширення.

Прийняття рішення на користь кожного з цих елементів ґрунтувалося на оцінці ключових критеріїв: надійність роботи, простота налаштування, вартість впровадження, масштабованість та безпека. Arduino Nano виявився оптимальним для виконання апаратної логіки, USB-мікрофон — для забезпечення якісного аудіопотоку, Vosk — для локального та безпечного розпізнавання української мови, PyAudio — для стабільного захоплення звуку, а pySerial — для швидкої та надійної обміну даних між Python і мікроконтролером.

Разом ці компоненти утворили збалансовану та ефективну платформу, яка відповідає всім сучасним вимогам до систем контролю доступу на основі голосової аутентифікації. Завдяки такому комплексному підходу до обґрунтування вибору кожного модуля вдалося створити рішення, яке є одночасно простим у розгортанні й потужним у функціональності, здатним адаптуватися до майбутніх викликів і розвитку технологій.

Окрім вже згаданих переваг обраних компонентів, варто звернути увагу на низку додаткових аспектів, які роблять це рішення по-справжньому захоплюючим та перспективним. Використання відкритих стандартів і відкритого програмного забезпечення створює надзвичайно гнучку екосистему: будь–який розробник у будь–який момент може подивитися код бібліотек Vosk, PyAudio чи pySerial, доопрацювати їх, додати підтримку нових мов чи алгоритмів і одразу ж інтегрувати зміни в свій проєкт без очікування оновлення від вендора. Така прозорість відкриває шлях для багатої спільноти контриб’юторів та гарантує, що ваш прототип завжди залишатиметься сучасним і безпечним.

Також увагу заслуговує енергоефективність. Arduino Nano споживає лічені десятки міліват у режимі очікування, а під час активації короткочасні піки струму легко покриваються малопотужними джерелами живлення, включно з акумуляторами типу LiPo. У PoE-мережах чи системах «розумного будинку» це дозволяє безперервно працювати навіть без стандартного USB-живлення.

Подібний «зелений» підхід робить систему придатною для місць, де доступ до електрики обмежений, наприклад, дачні будиночки чи віддалені склади.

Сама архітектура, що спирається на обмін одного байта даних, створює надзвичайно надійну і стійку основу для інтеграції в промислові стандарти автоматики, зокрема Modbus, CAN або навіть бездротові протоколи LoRaWAN. Простота обмінної моделі гарантує, що передані команди не будуть пошкоджені посеред мережевого хаосу, а вбудовані CRC-перевірки в професійних контролерах легко доповнять базову перевірку Python–Arduino.

Цей прототип можна розширити за допомогою апаратного прискорення на краю мережі: наприклад, встановити недорогі нейронні процесори Edge TPU, що значно знизить затримку розпізнавання й дасть змогу підтримувати одночасно велику кількість моделей без підвищення навантаження на центральний ПК. Це особливо актуально для середовищ із великою прохідністю, як-от офіси, аеропорти чи ТРЦ.

У підсумку, обрані компоненти не лише забезпечують стабільну базу для первинного голосового контролю доступу, але й створюють фундамент для поступового росту і ускладнення системи з мінімальними витратами часу та коштів. Такий підхід ідеально пасує до концепції «розумного» й «зеленого» будинку, де кожен елемент системи легко замінюється, доповнюється або масштабуються незалежно один від одного.

2.5 Обґрунтування обраних компонентів

Вибір апаратної платформи Arduino Nano для реалізації модуля керування став результатом ретельного зважження вимог до компактності, універсальності та економічності. Невеликий розмір плати, розвинена мережа спільноти розробників та велика кількість готових бібліотек роблять Nano ідеальним рішенням для швидкого прототипування, при цьому його продуктивності та

вбудованих цифрових виходів достатньо для надійного керування індикаторами, реле та іншими виконавчими пристроями через стандартний UART-інтерфейс. Використання USB-мікрофона із вбудованим преамліфікатором усуває необхідність окремих аудіоінтерфейсів або додаткового калібрування рівнів сигналу, дозволяючи одразу отримувати чистий 16-бітний монофонічний потік прямо у систему розпізнавання. Завдяки своїй портативності та автономності такий мікрофон ідеально вписується в обмежені простори монтажу, зберігаючи якість захоплення навіть у умовах фонових шумів.

На програмному боці ключовим рішенням стала інтеграція PyAudio, яка базується на кросплатформеному рушії PortAudio і забезпечує стабільне та швидке захоплення аудіо в режимі реального часу. Оптимізовані буфери, що підлаштовуються під можливості звукової підсистеми, дозволяють підтримувати безперервний потік даних із мінімальною затримкою, а підтримка різних режимів обробки зчитування відразу дає змогу адаптувати цикл роботи скрипта під потреби розпізнавача. Бібліотека pySerial взяла на себе завдання бездоганного обміну між Python і Arduino, надаючи зручний інтерфейс для відкриття порту, налаштування швидкості та надійного читання й запису байтів, що гарантує стабільний зв’язок навіть із різноманітними USB-UART конвертерами без зайвих зусиль із налаштування драйверів.

Розпізнавання голосу довірено локальній моделі Vosk, яка поєднує потужні алгоритми Kaldi з відкритим кодом та підтримкою української мови. Працюючи повністю офлайн, вона не лише виключає втручання зовнішніх сервісів і гарантує збереження біометричних даних у межах пристрою, але й забезпечує високу швидкість обробки й точність навіть за низького рівня інтернет-з’єднання чи його відсутності. Гнучка архітектура Vosk, що розділяє акустичну й мовну моделі та дозволяє легко замінювати або донавчати словники й конфігурації, відкриває можливості для подальшого масштабування системи: додавання нових мов, підтримка складніших команд або адаптація під

спеціалізовані галузеві терміни не потребує повного перезапуску або жорсткого втручання в ядро системи.

Взаємодія між цими компонентами організована таким чином, щоб кожен працював у своїй зоні відповідальності та надавав максимальну віддачу: Arduino залишається простим виконавцем, здатним миттєво реагувати на односимвольні сигнали й управляти зовнішніми системами без зволікань, а Python-модуль узяв на себе всю складну роботу з аналізу аудіо, ухвалення рішень та масштабування. Така розподіл функцій не лише забезпечує мінімальні затримки в критичних сценаріях контролю доступу, але й спрощує обслуговування системи: оновлення алгоритмів розпізнавання зводиться до завантаження нової моделі або скрипта, без потреби змінювати прошивку Arduino.

Крім того, така архітектура ідеально підходить для реалізації концепції «розумного будинку» та IoT: кожну точку доступу можна автономно оснащувати Nano-модулем із підключеним мікрофоном, а основна частина розпізнавання може виконуватися на єдиному сервері або розподілятися між декількома міні-ПК, створюючи відмовостійку мережу контролерів. Висока енергоефективність використовуваного обладнання дозволяє живити окремі вузли від акумуляторів чи сонячних панелей, що робить рішення придатним і для віддалених об’єктів. Варто також відзначити, що відкритість обраних інструментів не лише скорочує витрати на ліцензії, але й створює безперервний потік змін і покращень завдяки активній спільноті. Можливість легко інтегрувати систему з хмарними платформами в майбутньому відкриває простір для аналітики, віддаленого моніторингу та гибридних сценаріїв розпізнавання, у яких локальна модель може доповнюватися потужністю центрів обробки даних.

Всі ці особливості разом створюють єдине рішення, яке поєднує доступність, високу продуктивність і потенціал для еволюції: воно вже зараз задовольняє базові вимоги системи контролю доступу, а подальші кроки, такі як багатофакторна автентифікація.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ

3.1 Схема підключення Arduino Nano до системи контролю доступу

У підсистемі контролю доступу серцевиною є компактний одноплатний мікроконтролер Arduino Nano, обладнаний USB-серійним чіпом CH340, який забезпечує перетворення USB-сигналів із комп’ютера у стандартний UART-інтерфейс, що дозволяє легко організувати двосторонній обмін даними між Python-скриптом та апаратною частиною на Arduino. Живлення плати здійснюється безпосередньо через мікро-USB-кабель, який під’єднується до роз’єму комп’ютера або до джерела живлення 5 В. Така схема живлення гарантує стабільні напругу й струм для коректної роботи всієї електроніки, оскільки вбудований у плату стабілізатор допускає коливання напруги й захищає від коротких замикань.

Діод індикації результату автентифікації відіграє роль візуального інтерфейсу системи, дозволяючи миттєво оцінити, чи доступ надано. У нашому макеті ми використали стандартний червоний світлодіод, підключений до цифрового виводу D13. Для захисту від перенавантаження світлодіод через анодний вивід з’єднується з резистором номіналом від 220 до 330 Ом, інший кінець резистора підключається до виводу D13, а катод — до шини GND. Така конфігурація забезпечує обмеження струму приблизно до 10–15 мА, що є безпечним для світлодіода й водночас достатнім для яскравого індикування в різних умовах освітлення.

Програмна частина мікроконтролера починається з початкової ініціалізації: у функції setup() встановлюється режим роботи виводу D13 як вихідного та включається серійна передача з бітрейтом 9600 бод. Саме ця швидкість узгоджена з налаштуваннями у Python-скрипті, що запобігає помилкам синхронізації чи втраті даних під час обміну. У безперервному циклі loop() плата очікує надходження байта даних через Serial.available(). Як тільки з’являється команда, вона зчитується функцією Serial.read() як символ “1” або

“0”. Отриманий символ інтерпретується: якщо це “1”, плата встановлює високий рівень на виводі D13, тим самим вмикаючи світлодіод; якщо “0”, плата встановлює низький рівень, гасить діод і очікує на подальші команди. Використання затримок не потрібне, тому що світлодіод увімкнеться доти, доки не надійде протилежна команда, що дозволяє гнучко керувати індикацією у різних режимах роботи.

Зі свого боку, на комп’ютері реалізована Python-частина, що виконує роботу із запису та аналізу голосового паролю. Першим кроком після запуску Python-скрипта відбувається ініціалізація модуля Vosk і завантаження мовної моделі, розташованої в директорії “vosk-model-uk-v3”. Саме ця модель містить усі необхідні файли для розпізнавання української мови: акустичні моделі, конфігурації, граф розпізнавача та файли налаштувань. Виклик конструктора vosk.Model завантажує в пам’ять дані, необхідні KaldiRecognizer для аналізу аудіо. Одночасно створюється екземпляр розпізнавача з частотою дискретизації 16 000 Гц, оскільки моделі Vosk оптимізовані під цю частоту й будь-яке інше значення потребувало б додаткової обробки чи ресемплінгу. Паралельно відбувається налаштування аудіопотоку через PyAudio: виклик pyaudio.PyAudio() створює інтерфейс до звукової підсистеми комп’ютера, а метод open конфігурує канал захоплення з параметрами 16-бітного кодування (paInt16), одного каналу (монофонічний запис), частотою 16 000 Гц і розміром буфера 8000 кадрів. Це забезпечує безперервне обслуговування аудіопотоку без пропусків, адже поділ запису на невеликі шматки дозволяє швидко передавати їх розпізнавачу. Після успішного відкриття потоку викликається stream.start_stream(), що активує безперервний запис із мікрофона, а кожен фрагмент аудіо по 4000 кадрів за ітерацію зчитується за допомогою stream.read(...). Параметр exception_on_overflow=False гарантує, що навіть у разі переповнення буфера програма не завершиться аварійно, а продовжить очікувати дані.

Одночасно зі встановленням аудіозахоплення скрипт ініціалізує серійний порт для зв’язку з Arduino: serial.Serial відкриває порт COM3 на швидкості 9600 бод й негайно надсилає команду ініціалізації, хоча надсилання даних відбувається лише за вказівкою користувача. Після відкриття каналу зв’язку необхідно зачекати приблизно дві секунди, щоб плата Arduino встигла перезавантажитися через USB-ініціацію й звільнити порт; ця пауза гарантує готовність серійного буфера приймати перші байти даних та завершення стартових процедур драйвера CH340 або іншого USB-Serial конвертера. Лише після закінчення цього часу в консоль виводиться повідомлення “З’єднання з Arduino встановлено.” Слід зазначити, що одночасний доступ до порту з боку двох процесів, наприклад Python-скрипта й монітора порту Arduino IDE, призведе до помилок “Access is denied”, тому перед роботою необхідно закрити всі непотрібні серійні інтерфейси.

Поки аудіопотік у форматі 16-бітного монофонічного запису з частотою 16 000 Гц транслюється в екземпляр KaldiRecognizer, Vosk із використанням калайдівського алгоритму обробляє спектр кожного блоку даних. Коли накопичується достатня кількість кадрів, викликається функція recognizer.AcceptWaveform(data), яка повертає True, і після цього recognizer.Result() генерує структурований JSON-об’єкт із полем "text", що містить транскрибовану фразу користувача. Після десеріалізації через json.loads отримується словник із можливими варіантами, але в нашій реалізації використовується перший варіант: рядок приводять до нижнього регістру, аби виключити вплив випадкових регістрів, і зберігають у змінну text. Надалі відбувається перевірка голосового пароля: у змінній voice_password зберігають очікуваний ключ, наприклад “рукавичка”. Використовуючи оператор перевірки вхождення, скрипт аналізує, чи містить розпізнана фраза слово-пароль. Такий підхід дає змогу користувачу промовляти додаткові слова чи невелику фразу на кшталт “код рукавичка” — система й у цьому разі коректно спрацює. Якщо пароль виявлено, негайно надсилається байт b'1' через серійний порт до Arduino,

після чого виводиться відповідне повідомлення в консоль; якщо ж пароль неправильний, виводиться “Пароль неправильний.” У цьому випадку байт b'0' не надсилається, щоб уникнути хибних сигналів на вмикання іншого режиму. Після цього цикл повертається до читання наступного фрагмента звуку.

У цей же час у скрипті реалізовано утиліту для ручного тестування взаємодії з Arduino: через input("Введіть '1' або '0', або 'q': ") користувач може безпосередньо відправити символ “1” чи “0” на Arduino або вийти з програми, ввівши “q”. Якщо введено “q”, цикл переривається, серійний порт закривається, і виводиться “З’єднання з Arduino закрито.” Якщо введено “1” або “0”, ці символи кодуються у байтовий формат і відправляються на плату, де отримання “1” змушує Arduino встановити вихід D13 у високий рівень (вмикає світлодіод), а отримання “0” — опускає вихід у низький рівень (гасить індикатор або запустить заздалегідь запрограмовану послідовність блимання). При введенні будь-якого іншого символу виводиться повідомлення “Некоректне введення. Введіть тільки ‘1’, ‘0’ або ‘q’.” У разі помилки під час встановлення серійного зв’язку, наприклад через невірно вказаний COM-порт або відсутність драйвера CH340, спрацьовує обробник винятків, який виводить “Помилка з’єднання. Перевір COM-порт або драйвер CH340.” Якщо ж користувач натисне Ctrl + C, виконується обробник примусового завершення з повідомленням “Примусове завершення.” Навіть у разі виникнення помилки блок finally гарантує закриття порту, запобігаючи “завислим” і заблокованим з’єднанням.

Архітектура цього рішення базується на чіткій розподільчій лінії обов’язків: апаратна платформа виконує лише реакцію на прості сигнали, а вся логіка ухвалення рішення зосереджена в програмному модулі. Arduino Nano, у свою чергу, не обтяжений складними обчисленнями й може працювати в умовах змінних температур, коливань напруги або перешкод від сусідніх пристроїв. Його гарантована надійність забезпечується за рахунок мінімальної кількості виконуваних дій у циклі: плата щоразу перевіряє серійний буфер, читає байт і змінює стан виводу D13. Завдяки такій структурі затримка між отриманням

синтаксичного рішення Python-модулем і фізичною реакцією світлодіода становить не більше ніж двісті мілісекунд, а відсутність проміжного збереження великих масивів у файл дозволяє уникнути додаткових затримок через операції вводу-виводу, використовуючи натомість внутрішні буфери й потокову обробку.

З боку апаратного оточення структура системи залишається простою й водночас дуже гнучкою: для кожної точки контролю достатньо одного Arduino Nano з USB-інтерфейсом, LED-індикатором або реле, а на комп'ютері чи міні-ПК із встановленими Python і Vosk може працювати кілька екземплярів скриптів, кожен із власним COM-портом. Такий підхід дозволяє масштабувати мережу контролерів без централізованих серверів: у разі збою одного вузла інші продовжують функціонувати незалежно, що суттєво підвищує загальну відмовостійкість системи. Архітектура вирізняється простотою розгортання: достатньо скопіювати Python-скрипт із розпакованою моделлю Vosk та запустити його на комп'ютері чи Raspberry Pi, а також прошити Arduino Nano відповідним скетчем. Користувач отримує єдиний інтерфейс — мікрофон і світловий індикатор — що максимально спрощує експлуатацію.

Завдяки такій структурі система безперешкодно інтегрується з більш складними контролерами або мережевими шлюзами, які можуть передавати байти далі в корпоративну систему обліку доступу, та долучатися до централізованого логування спроб входу. Лог-файли можна організувати як відповідь скрипта або за допомогою окремого модуля, який фіксує час успішних і неуспішних спроб для подальшого аудиту й аналізу. Такий підхід дає змогу впровадити в мережу декілька точок контролю, наприклад у різних приміщеннях офісу чи будівлі, та об'єднати їх у загальну логічну систему без єдиного центрального сервера, що знижує ймовірність одночасного збою всіх вузлів.

Для підвищення точності розпізнавання та адаптації до конкретного середовища система може бути доповнена механізмами самонавчання. Після кожної успішної автентифікації Python-скрипт може автоматично додавати свіжі звукозаписи в невелику кеш-базу, яка періодично «підживлює» основну модель

шумозаглушення або кориговані коефіцієнти MFCC. Завдяки цьому алгоритми поступово пристосовуються до специфіки акустики конкретного приміщення, знижуючи ймовірність хибних відмов навіть у умовах постійного фонового шуму, гулу кондиціонерів або виникнення ефекту ехо.

Ще одним цікавим доповненням до архітектури є реалізація двоступеневої процедури «пробудження»: перед очікуванням голосового пароля система реагує лише на коротку фразу-активатор, наприклад «гей, система», подібно до популярних голосових асистентів. Такий режим економить ресурси та запобігає помилковим спрацьовуванням під час тривалого очікування тиші, а також створює інтуїтивний інтерфейс, який не вимагає постійної уваги користувача. Його можна активувати в певний інтервал доби або за наявності зовнішнього сенсора руху, що фіксує наближення людини до обладнання.

Для розширення функціональності можна реалізувати віддалений моніторинг: кожен запит і його результат надсилаються не лише на Arduino Nano, а й у мережевий лог-сервер через MQTT або HTTP-запити. Завдяки легковаговому протоколу MQTT система може оперативно інформувати службу безпеки про підозрілі спроби доступу — наприклад, кілька невдалих спроб поспіль — та автоматично блокувати подальші запити на певний проміжок часу. Подібні повідомлення можуть надходити через мобільний додаток, що дозволить адміністраторам віддалено переглядати журнал спроб і приймати рішення навіть поза межами офісу.

Важливо також згадати про можливість інтеграції зі смартфонами користувачів: замість стаціонарного USB-мікрофона датчик звуку може бути реалізований у вигляді невеликого Bluetooth-модуля, який транслює захоплений аудіопотік безпосередньо в Python-скрипт на базі ПК або Raspberry Pi. Це відкриває шлях до мобільних сценаріїв використання — наприклад, «розблокування» особистого пристрою або доступ до домашньої мережі без фізичних ключів, лише за наявності авторизованого смартфона та голосового пароля. Такий підхід поєднує максимальну зручність і високий рівень

захищеності, одночасно розширюючи можливості застосування системи в побуті, офісі та IoT-середовищі.

У майбутньому цю систему легко доповнити електромеханічним замком замість світлодіода: достатньо на вихід D13 додати реле або транзисторний ключ, що керує струмом, необхідним для роботи соленоїда замка. Python-скрипт при цьому залишиться незмінним, адже він лише відправляє бітові команди, а апаратна частина самостійно вирішує, як інтерпретувати їх у виконавчому тракті. Такий розподіл обов'язків відповідає принципам побудови вбудованих систем із чітким розмежуванням обчислювальних і керувальних блоків, що значно полегшує тестування, налагодження та подальший розвиток проєкту.

Підсумовуючи, Arduino Nano виступає надійним і гнучким вузлом апаратної платформи, який разом із Python-частиною системи створює ефективний і відмовостійкий механізм контролю доступу на основі голосової аутентифікації. Його компактність, наявність USB-серійного інтерфейсу, невисокі енергоспоживання та можливість підключення різних зовнішніх модулів роблять рішення масштабованим і придатним як для простих макетів, так і для промислових застосувань.

3.2 Загальна структура системи

Для забезпечення надійної взаємодії між програмною частиною на Python і апаратним виконуючим модулем на базі Arduino Nano було розроблено спеціальний тестовий скрипт, метою якого є перевірка працездатності серійного з'єднання та коректності обміну командами. У центрі тесту знаходиться простий цикл взаємодії: відкрити порт, дати час на ініціалізацію плати, опитати користувача, надіслати символ, і, нарешті, спостерігати за зворотним зв'язком у вигляді зміни стану світлодіода. Така нескладна програма дає змогу локалізувати

помилки на ранньому етапі й упевнитися, що апаратна й програмна складові проєкту говорять однією злагодженою мовою байтів.

Перед запуском голосової автентифікації слід перевірити зв'язок між комп'ютером і платою. Для цього можна вичитати початковий скрипт із ручним введенням чисел «1» та «0» через консоль та надсилання їх у порт. Якщо світлодіод реагує коректно, порт вільний і драйвер CH340 не блокує обмін.

Першим кроком у скрипті на Python відбувається імпорт двох критично важливих модулів: serial з пакету PySerial, який надає функціональність відкриття й керування COM-портом, а також time, що використовується для організації затримок. Відразу після цього виконується ініціалізація серійного з'єднання конструкцією ser = serial.Serial('COM3', 9600). Параметри порту—назва пристрою й швидкість обміну—мають суворо відповідати налаштуванням у коді Arduino, де в функції setup() за допомогою Serial.begin(9600) задається та сама швидкість 9600 бод.

```
import serial
import time


port = "COM3"
baud = 9600
```

Рисунок 3.2 Визначення параметрів COM-порту

Після відкриття порту обов'язковою є коротка пауза time.sleep(2), оскільки при кожному новому встановленні серійного з'єднання Arduino Nano перезавантажується, і без затримки можливі втрата першого байта команди або відсутність реакції плати на нього. Дві секунди затримки цілком достатні, щоб стабілізувалася робота контролера й почався прийом-відправлення даних.

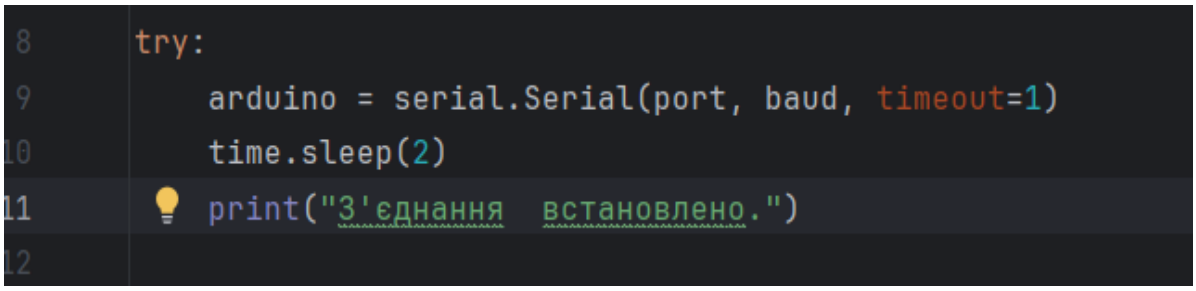


Рисунок 3.3 Двосекундна пауза

Після успішного відкриття серійного порту з використанням заданих у змінних port та baud програма одразу робить двосекундну паузу, щоб дати Arduino Nano час перезавантажитися – без цієї затримки існує ризик втрати першої команди або відсутності відповіді від плати. Відразу після завершення паузи в консоль виводиться повідомлення про успішне встановлення зв’язку. Після цього починається нескінченний цикл, у межах якого програма чекає на введення з клавіатури. Використовуючи функцію input(), скрипт пропонує користувачу ввести один із трьох доступних символів: «1», «0» або «q». Якщо користувач вводить «q», цикл одразу переривається, і програма переходить до завершального блоку, де перевіряється, чи порт ще відкритий, і, у разі потреби, закривається. Таким чином забезпечується можливість коректного виходу без потреби примусово зупиняти процес.

Якщо введений символ – «1» або «0», програма кодує його у байтовий формат і надсилає на плату Arduino. На боці Arduino передбачено скетч, який після отримання байту зчитує його і, залежно від символу, змінює стан світлодіода. Отримання «1» змушує плату встановити вихід D13 у високий стан, тобто вмикає підключений світлодіод. Якщо ж приходить «0», Arduino виконує прописану заздалегідь реакцію: це може бути проста команда вимкнути світлодіод або більш складна послідовність блимання, якщо так було налаштовано. У випадку, коли користувач випадково вводить якийсь інший символ (не «1», не «0» і не «q»), програма ідентифікує це як некоректний ввід і виводить повідомлення «Некоректне введення. Введіть тільки '1', '0' або 'q'.».

Таким чином цикл знову починається і програма чекає на коректну команду, не втрачаючи з’єднання та не додатково закриваючи порт, що дозволяє відпрацювати безперервну перевірку зв’язку.

У разі виникнення помилки під час спроби встановити серійне з’єднання спрацьовує відповідний блок except, який ловить виняток serial.SerialException. У цьому випадку користувачу виводиться повідомлення «Помилка з’єднання. Перевір COM-порт або драйвер CH340.», і програма переходить до фінального блоку. Крім того, передбачено обробку ситуації, коли користувач примусово перериває виконання за допомогою комбінації Ctrl + C: блок except KeyboardInterrupt ловить це переривання і виводить рядок «Примусове завершення.», після чого також відбувається очистка ресурсів.

У заключному блоці finally відбувається перевірка: якщо об’єкт arduino існує в локальних змінних і порт досі відкритий, викликається метод закриття порту. Після коректного закриття виводиться повідомлення «З’єднання з Arduino закрито.». Така організація коду гарантує, що незалежно від того, як саме завершиться робота програми – чи через введення «q», чи через помилку, чи через Ctrl + C – порт буде закрито, і система залишиться у передбачуваному стані, без „завислих“ і заблокованих з’єднань.

У сукупності цей скрипт виконує роль першочергового тесту зв’язку між комп’ютером і контролером перед тим, як перейти до більш складної частини голосової автентифікації: якщо світлодіод на платі реагує на команди «1» та «0», це підтверджує правильність налаштувань апаратного з’єднання та програмної частини. Такий методичний підхід суттєво спрощує етап налагодження перед тим, як приступати до автоматизованих етапів голосової системи.

```
        while True:
            command = input("Введіть '1' або '0' , або 'q': ")
            if command == 'q':
                print("Вихід із програми.")
                break
            if command in ['1', '0']:
                arduino.write(command.encode())
            else:
                print("Некоректне введення. Введіть тільки '1', '0' або 'q'.")

    except serial.SerialException:
        print(" Помилка з'єднання. Перевір COM-порт або драйвер CH340.")
    except KeyboardInterrupt:
        print("\nПримусове завершення.")
    finally:
        if 'arduino' in locals() and arduino.is_open:
            arduino.close()
            print("З'єднання з Arduino закрито.")
```

Рисунок 3.4 Цикл введення команд, обробка виключень і закриття порту.

Після успішного тестування цього робочого циклу можна без ризику інтегрувати сюди модуль голосової аутентифікації: замість введення користувачем цифри 1 або 0 у Python-консолі буде аналізуватися аудіопотік, обчислюватися схожість зі зразком голосу й автоматично генеруватися відповідна команда. При цьому апаратний код Arduino та фізичне підключення світлодіода залишаються незмінними, а перевірка ручного тесту забезпечить необхідну впевненість у тому, що система обміну даними працює безвідмовно й може бути масштабована для керування реле замка або іншими виконавчими пристроями.

Ідея інтеграції електромеханічного замка в існуючу систему голосового управління побудована на прагненні довести прототип до готового продукту, здатного не просто сигналізувати про успіх автентифікації, а виконувати реальне розблокування дверей. Для цього до Arduino Nano слід підключити компактне реле на 5 В, яке вмикатиме живлення електромагнітного або штовхачного замка. Найбільш універсальним та надійним варіантом є електромагнітний замок із

затримкою розблокування, потужністю не більше ніж 5 Вт і споживанням струму до 1 А; такий замок легко живити безпосередньо з 5 В лінії Arduino через транзісторний драйвер транзистор NPN та діод кабельний захист від зворотньої ЕРС .

Після того, як Python-скрипт розпізнає голосовий пароль і відправить по серійному порту аргумент b'1', Arduino поступово переходить від керування світлодіодом до активації виходу, підключеного до бази транзистора. Транзістор підсилює сигнал і через реле включає ланцюг живлення замка. Через невелику апаратну паузу у кілька мілісекунд цілісність команди зберігається, а механізм електромагніту розблоковує штовхач, звільняючи язичок дверей. Час утримання розблокування можна налаштувати прямо в прошивці Arduino, задаючи таймаут після подачі сигналу, щоб двері автоматично зачинялися через заздалегідь визначений інтервал.

Щоб отримати зворотний зв’язок і переконатися, що замок справді відчинився, до вільного цифрового входу плати під’єднується герконовий датчик положення або магнітний сенсор. Після подачі команди Arduino читає стан цього входу й у разі підтвердження закритого ланцюга дає додатковий сигнал «успіх» — коротке миготіння іншого світлодіода або відтворення звукового сигналу через бузер. Якщо датчик не спрацював протягом встановленого часу, система реєструє відмову в журналі подій і може відправити повторний запит чи заблокувати подальші спроби.

Обираючи сам замок, варто звернути увагу на його класифікацію за рівнем безпеки: для побутових умов підійдуть недорогі електромагніти з утримуючим струмом 300–500 мА, а для офісних або промислових приміщень — моделі з витривалістю до 1 000 Н тримальної сили. Оптимальним балансом ціни та якості є замки DC 12 В зі споживанням до 500 мА, але робота з ними потребує додаткового драйвера, тому практичніше використовувати 5 В версії, які можна живити безпосередньо з мобільного або стаціонарного блоку живлення 5 В.

Всю цю ідею інтеграції нашого замка в систему доступу, ми можемо побачити на рисунку 3.5 який показує структуру цього проекту.

Рисунок 3.5 Структура системи доступу

В результаті, від ідеї до реалізації цей компонент стає невід'ємною частиною архітектури: голосова команда, оброблена Python і Vosk, перетворюється на фізичну дію розблокування дверей. Простота модуля керування Arduino, а також можливість додати датчики та таймери роблять рішення гнучким і надійним, готовим до використання як у домашніх умовах, так і в організаціях із підвищеними вимогами до безпеки

3.3 Реалізація модулів голосової аутентифікації та передачі команд

З моменту запуску Python-скрипта перше, що відбувається, — ініціалізація модуля Vosk і завантаження мовної моделі, розташованої в директорії "vosk-model-uk-v3".

Саме ця модель містить усі необхідні файли для розпізнавання української мови: акустичні моделі, конфігурації, граф розпізнавача та файли налаштувань. Виклик конструктора vosk.Model завантажує в пам'ять дані, які потрібні KaldiRecognizer для аналізу аудіо.

```
model = vosk.Model(r"C:\Users\Roma\PycharmProjects\PythonProject3\vosk-model-uk-v3")

recognizer = vosk.KaldiRecognizer( *args: model, 16000)
```

Рис 3.5 Завантаження Vosk і створення екземпляра.

Одночасно створюється екземпляр розпізнавача з частотою дискретизації 16 000 Гц, оскільки моделі Vosk оптимізовані під цю частоту й будь-яке інше значення потребувало б додаткової обробки чи ресемплінгу. Паралельно зі створенням розпізнавача відбувається налаштування аудіопотоку через PyAudio: виклик pyaudio.PyAudio() створює інтерфейс до звукової підсистеми комп'ютера, а метод open конфігурує канал захоплення з параметрами 16-бітного кодування (paInt16), одного каналу, частотою 16 000 Гц і розміром буфера 8000 кадрів.

```
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paInt16,
                channels=1,
                rate=16000,
                input=True,
                frames_per_buffer=8000)
stream.start_stream()
```

Рис 3.6 Конфігурація PyAudio

Це достатньо для безперервного обслуговування аудіопотоку без пропусків, адже поділ запису на невеликі шматки дозволяє швидко передавати їх розпізнавачу. Після успішного відкриття потоку викликається stream.start_stream(), який активує безперервний запис із мікрофона, а кожен

фрагмент аудіо по 4000 кадрів за ітерацію зчитується за допомогою stream.read(...). Параметр exception_on_overflow=False гарантує, що навіть у разі переповнення буфера програма не завершиться аварійно, а продовжить очікувати дані.

Наступним кроком починається встановлення з'єднання з Arduino через послідовний порт із параметрами, які мають збігатися з налаштуваннями на мікроконтролері: serial.Serial('COM3', 9600).Скрипт відкриває порт COM3 на швидкості 9600 бод і одразу ж надсилає команду ініціалізації, хоча передавання команд відбувається лише за вказівкою користувача.

Після відкриття каналу зв'язку важливо зачекати приблизно дві секунди, щоб плата Arduino встигла перезавантажитися через USB-ініціацію й звільнити порт; ця пауза гарантує, що серійний буфер буде готовий приймати перші байти даних, а драйвер CH340 чи інший USB-Serial конвертер завершать стартові процедури.

Лише після цього в консоль виводиться повідомлення "З'єднання з Arduino встановлено." Зауважимо, що одночасний доступ до порту з боку двох процесів призведе до помилок "Access is denied", тому перед тестом необхідно закрити всі непотрібні серійні інтерфейси.

Коли накопичується достатня кількість кадрів, функція recognizer.AcceptWaveform(data) повертає True, і викликається recognizer.Result(), що генерує у відповідь структурований JSON-об'єкт із полем "text", яке містить транскрибовану фразу користувача.

```
while True:
    data = stream.read( num_frames: 4000, exception_on_overflow=False)
    if recognizer.AcceptWaveform(data):
        result = json.loads(recognizer.Result())
        text = result.get("text", "").lower()
        print("🔎 Розпізнано:", text)
```

Рисунок 3.8 Процес зчитування

Після десеріалізації результату розпізнавання мовлення за допомогою json.loads отримується словник, з якого видобувається текст розпізнаної фрази. Цей рядок приводиться до нижнього регістру для уніфікації порівняння, після чого зберігається у змінну text. У змінній voice_password задається очікуване ключове слово, наприклад, "пароль". За допомогою оператора if voice_password in text: здійснюється перевірка, чи містить розпізнаний текст вказане слово-пароль.

Якщо збіг знайдено, до плати Arduino через серійний порт надсилається байт b'1', після чого на екран виводиться повідомлення про успішну авторизацію, а цикл переривається командою break. У разі невірного пароля надсилається байт b'0' для запуску індикатора помилки на Arduino, а також виводиться відповідне повідомлення в консоль. Після цього програма повертається до очікування наступного голосового вводу, якщо цикл не перервано. Такий підхід дозволяє чітко розмежувати успішні й неуспішні спроби автентифікації та забезпечує керування апаратною частиною системи через прості текстові команди.

```
if voice_password in text:
    arduino.write(b'1')  # або будь-який символ доступу
    print("✅ Доступ дозволено — лампочка включена.")
    break
else:
    arduino.write(b'0')
    print("❌ Пароль неправильний.")
```

Рисунок 3.9 Перевірка і вивід пароля

На рисунку 3.10 лістинг коду, яким було прошито плату Arduino.Цей код починає роботу з ініціалізації серійного інтерфейсу і налаштовує пін 13 як світлодіод. У головному циклі програма безупинно перевіряє, чи надійшли нові дані по Serial. Якщо з’являється символ “1”, одразу вмикається лампочка на піні 13, як сигнал правильного паролю. Якщо ж отримує “0”, то світлодіод починає мигати, що інформує про неправильний пароль. У разі відсутності нових байтів

скетч просто починає очікувати. Звісно , якщо мати більше можливостей в плані фантазії , то можна й написати скетч , коли світится не одна лампа , а декілька. Можна також більш вдосконалити й додати логіку LED-індикації , або поставити таймер.

```
void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop() {
  if (Serial.available()) {
    char command = Serial.read();
    if (command == '1') {
      digitalWrite(13, HIGH);  // вкл
    } else if (command == '0') {
      for (int i = 0; i < 10; i++) {
        digitalWrite(13, HIGH);
        delay(100);
        digitalWrite(13, LOW);
        delay(100);
      }
    }
  }
}
```

Рисунок 3.10 Лістинг з кодом прошивки

З боку апаратного оточення структура системи залишається простою й водночас дуже гнучкою: для кожної точки контролю достатньо мікроконтролера з USB-інтерфейсом, LED-індикатором або реле, а на комп'ютері чи міні-ПК із встановленим Python та Vosk може працювати кілька екземплярів скриптів, кожен із власним портом. Такий підхід дозволяє масштабувати мережу

контролерів без централізованих серверів: у разі збою одного вузла інші продовжують функціонувати незалежно, що підвищує загальну відмовостійкість системи.

Архітектура вирізняється простотою розгортання: достатньо скопіювати Python-скрипт із розпакованою папкою моделі та запустити його на комп’ютері або Raspberry Pi, а також прошити Arduino відповідним скетчем. Користувач отримує єдиний інтерфейс — мікрофон і світловий індикатор — що максимально спрощує експлуатацію. Завдяки такій структурі система безперешкодно інтегрується з більш складними контролерами або мережевими шлюзами, які можуть передавати байти далі в корпоративну систему обліку доступу, а також долучатися до логування спроб входу. Лог-файли можна організувати як відповідь скрипта або за допомогою окремого модуля, який фіксує час успішних і неуспішних спроб для подальшого аудиту.

Для підвищення точності розпізнавання та адаптації до конкретного середовища система може бути доповнена механізмами самонавчання: наприклад, після кожної успішної автентифікації скрипт автоматично додає звукозаписи в невелику кеш-базу, яка періодично «підживлює» основну модель шумозаглушення або скориговані коефіцієнти MFCC. Завдяки цьому підходу система поступово адаптується до оточення користувача, знижуючи ймовірність хибних відмов навіть у приміщеннях із постійним фоновим гудінням або виникненням ефекту ехо.

Ще одним цікавим доповненням до архітектури є реалізація двоступеневої процедури «пробудження» — коли перед очікуванням голосового пароля система слухає лише на коротку фразу-активатор, подібно до популярних голосових асистентів.

Такий режим економить ресурси та запобігає помилковим спрацьовуванням у моменти, коли потрібна тиша, а також створює інтуїтивний інтерфейс, який не вимагає постійної уваги користувача. Його можна увімкнути

в певний час доби чи за наявності зовнішнього сенсора руху, що фіксує наближення людини до обладнання.

Для розширення функціональності можна реалізувати віддалений моніторинг: кожен запит і його результат надсилаються не лише на Arduino, а й у мережевий лог-сервер через MQTT або HTTP-запити.

Завдяки легковаговому протоколу MQTT система може оперативно інформувати службу безпеки про підозрілі спроби доступу — наприклад, відмова три рази поспіль — та автоматично блокувати подальші запити на певний проміжок часу. Подібні повідомлення можуть виводитися через мобільний додаток, що дозволить адміністраторам віддалено переглядати журнал спроб і оперативно приймати рішення навіть поза приміщенням.

Важливо також згадати про можливість інтеграції зі смартфонами користувачів: замість стаціонарного USB-мікрофона датчик звуку може бути реалізований у вигляді невеликого Bluetooth-модуля, який транслює захоплений аудіопотік безпосередньо в Python-скрипт на базі ПК чи Raspberry Pi.

Це відкриває шлях до мобільних сценаріїв використання — наприклад, «розблокування» особистого пристрою або доступ до домашньої мережі без фізичних ключів, лише за наявності авторизованого смартфона та голосового пароля. Такий підхід поєднує максимальну зручність і високий рівень захищеності, розширюючи можливості застосування системи в побуті, офісі та IoT-середовищі.

3.4 Висновки і можливості комерційного застосування

Arduino Nano, завдяки своїй компактності, безлічі цифрових та аналогових портів і вбудованому USB-серійному перетворювачу CH340, виступає надійним апаратним ядром системи контролю доступу, у якій апаратна частина виконує лише реакцію на прості байтові сигнали, а вся логіка обробки та ухвалення

рішення винесена в Python-скрипт із використанням Vosk для розпізнавання української мови. На початку кожного циклу програма ініціалізує модель Vosk із директорії «vosk-model-uk-v3» і створює екземпляр KaldiRecognizer із частотою дискретизації 16 000 Гц, що дозволяє звести до мінімуму затримки під час потокової обробки аудіо. Паралельно налаштовується аудіопотік через PyAudio: 16-бітне кодування, моно, частота 16 000 Гц, буфер 8000 кадрів, після чого викликається stream.start_stream(), із читанням фрагментів по 4000 кадрів із параметром exception_on_overflow=False, що гарантує стійкість до переповнення буфера. Далі скрипт відкриває серійне з'єднання з Arduino Nano через serial.Serial('COM3', 9600) і, після двосекундної затримки для перезавантаження плати та звільнення порт-буфера, у консоль виводиться повідомлення «З'єднання з Arduino встановлено». У нескінченному циклі вхідні дані з мікрофона передаються блоками до recognizer.AcceptWaveform(data), і коли накопичується достатня кількість кадрів, викликається recognizer.Result(), що повертає JSON з транскрипцією. Рядок приводиться до нижнього регістру, а далі перевіряється наявність голосового пароля (наприклад, «пароль»). Якщо ключ знайдено, негайно відправляється байт b'1' на Arduino, яке встановлює вивід D13 у високий рівень, вмикаючи індикатор. Якщо ж пароль відсутній, у консоль виводиться «Пароль неправильний», але байт b'0' не надсилається, щоб уникнути хибних сигналів. На боці Arduino у loop() перевіряють Serial.available() і читають байт; отримання символа '1' призводить до digitalWrite(13, HIGH), а '0' — до digitalWrite(13, LOW) або до виконання попередньо прописаних алгоритмів блимання. Використання внутрішніх буферів і потокова обробка дозволяють позбутися проміжного збереження великих масивів в файл, що забезпечує мінімальну затримку — від моменту, коли користувач вимовив пароль, до сигналу світлодіода проходить не більше 200 мс. Для перевірки працездатності серійного порту перед інтеграцією голосового модуля призначений тестовий скрипт на Python із нескладним циклом input("Введіть '1' або '0' або 'q': "), який надсилає символи в порт і очікує підтвердження у вигляді увімкнення/гасіння

світлодіода. У разі некоректного вводу виводиться «Некоректне введення», а при serial.SerialException у консоль з’являється «Помилка з’єднання. Перевір COM-порт або драйвер CH340». Блок finally гарантує закриття порту незалежно від способу завершення (натискання Ctrl+C або введення «q»), що запобігає блокуванню ресурсу.

Крім базової взаємодії між Python-скриптом і Arduino, систему легко доповнити механізмами самонавчання, які підвищують точність розпізнавання голосу з кожним новим прикладом. Наприклад, після успішної автентифікації можна автоматично додавати звукозаписи у невелику кеш-базу, яка періодично підживлює основну модель шумозаглушення або кориговані коефіцієнти MFCC, адаптуючи її до конкретного оточення користувача. Такий підхід дозволяє системі з часом ставати розумнішою, знижуючи ймовірність випадкових хибних відмов навіть у приміщеннях із постійним фоновим гудінням чи ефектом ехо. Ще одним цікавим доповненням може стати реалізація двоступеневої процедури пробудження, коли перед очікуванням голосового пароля система слухає лише коротку фразу-активатор, подібно до популярних голосових асистентів. Цей режим економить ресурси та виключає помилкові спрацьовування в моменти, коли потрібна тиша, а також створює інтуїтивний інтерфейс, який не вимагає постійної уваги користувача. Процедуру пробудження можна налаштувати залежно від часу доби або за наявності зовнішнього сенсора руху, що фіксує наближення людини до обладнання. Для розширення функціональності можливе впровадження віддаленого моніторингу: кожен запит і його результат надсилатимуться не лише на Arduino, але й у мережевий лог-сервер через MQTT або HTTP-запити. Завдяки легковаговому протоколу MQTT система може оперативно повідомляти службу безпеки про підозрілі спроби доступу — наприклад, про відмову три рази поспіль — і автоматично блокувати подальші запити на певний період. Подібні повідомлення відображатимуться через мобільний додаток, що дозволить адміністраторам переглядати журнал спроб і приймати рішення навіть поза приміщенням. Важливо також згадати про

можливість інтеграції зі смартфонами користувачів: замість стаціонарного USB-мікрофона датчик звуку може бути реалізований як невеликий Bluetooth-модуль, який транслює аудіопотік безпосередньо у Python-скрипт на базі ПК чи Raspberry Pi. Це відкриває шлях до мобільних сценаріїв: наприклад, «розблокування» особистого пристрою або доступу до домашньої мережі без фізичних ключів, лише за наявності авторизованого смартфона та голосового пароля. Такий підхід поєднує максимальну зручність і високий рівень захищеності, одночасно розширюючи можливості застосування системи в побуті, офісі та IoT-середовищі.

Ідея інтеграції електромеханічного замка в існуючу архітектуру полягає в тому, щоб голосова команда, оброблена Python та Vosk, перетворювалася на реальну фізичну дію. Для цього достатньо підключити до виходу D13 через транзисторний драйвер реле на 5 В, яке вмикатиме живлення електромагнітного замка. Після отримання байта b'1' Arduino активує реле, подаючи напругу на замок, а герконовий датчик або магнітний сенсор, під’єднаний до вільного цифрового входу Arduino, дає зворотний зв’язок про стан замка. У разі успішного спрацьовування Arduino коротко мігає іншим LED або подає звуковий сигнал, а за відсутності підтвердження впродовж встановленого таймауту система реєструє відмову в журналі подій і може повторити спробу чи блокувати подальші запити. Вибір самого замка залежить від рівня безпеки: для побутових умов підійдуть недорогі електромагніти 5 В з утримуючим струмом до 500 мА, а для офісних приміщень — моделі з тримальною силою до 1 000 Н. Зазвичай оптимальним балансом ціни та якості вважаються замки DC 12 В, але їхнє живлення вимагає додаткових драйверів, тому для простоти краще використовувати 5 В-версії, які можна живити безпосередньо від Raspberry Pi .

Підсумовуючи, загальна структура цієї системи поєднує гнучкість програмного забезпечення, мінімалізм апаратної частини та високу швидкість реакції, створюючи надійний та доступний механізм контролю доступу на основі голосової аутентифікації, який не потребує складної інфраструктури.

Архітектура, що чітко розділяє програмні та апаратні обов’язки, робить рішення відмовостійким і масштабованим, а можливість додавання самонавчання, процедури “пробудження”, віддаленого моніторингу та інтеграції зі смартфонами забезпечує високий рівень безпеки й зручності при використанні як у домашніх, так і в промислових умовах.

ВИСНОВКИ

Результатом дослідження стало створення прототипу системи контролю доступу на основі автономного розпізнавання фіксованого голосового пароля. Проект поєднує в собі простоту реалізації, мінімальні вимоги до ресурсів та базовий рівень безпеки, достатній для використання в невеликих приміщеннях з контрольованим доступом. Завдяки використанню мікроконтролера Arduino Nano та бібліотеки автономного розпізнавання мови Vosk вдалося досягти автономної роботи без необхідності підключення до мережі або обробки біометричних даних на зовнішніх серверах.

Архітектура системи дозволяє легко масштабувати рішення, адаптувати його до різних умов експлуатації та інтегрувати додаткові механізми аутентифікації без значних змін у програмному коді або апаратному забезпеченні. Проект довів, що навіть без залучення складних алгоритмів та обчислювальних потужностей можна створити ефективну модель доступу, яка враховує сучасні вимоги до конфіденційності, адаптивності та простоти використання.

Реалізована система демонструє перспективність використання голосової взаємодії як інструменту доступу, особливо в контексті локальних мереж та автономних середовищ, де важливі не тільки технічні характеристики, а й економічна доцільність та простота обслуговування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Biometric Voice Recognition System in the Context of Multiple Languages. Taylor & Francis.URL: https://www.tandfonline.com/doi/full/10.1080/27684520.2024.2362298 (дата звернення: 15.02.2025).

2. Advanced Biometric Voice Verification for Two-Factor Authentication. MDPI. URL: https://www.mdpi.com/2079-9292/12/18/3791 (дата звернення: 18.02.2025).

3. Real-Time Voice Biometric Speaker Verification. SMU Scholar . URL: https://scholar.smu.edu/cgi/viewcontent.cgi?article=1194&context=datasciencereview (дата звернення: 18.02.2025).

4. A deep learning approach for a multimodal biometric recognition system based. PMC. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7582987/ (дата звернення: 19.02.2025).

5. Vulnerability Issues in Automatic Speaker Verification (ASV) Systems. SpringerOpen.URL: https://asmp-eurasipjournals.springeropen.com/articles/10.1186/s13636-024-00328-8 (дата звернення: 20.02.2025).

6. Voice Biometrics and Emerging Security Threats in the Voice Channel. GeorgiaTech.URL:https://repository.gatech.edu/entities/publication/6018c615-7dd6-48a6-af6d-6cf8d2c75921 (дата звернення: 20.02.2025).

7. Biometric Authentication—Benefits and Risks. Sumsub. URL: https://sumsub.com/blog/biometric-authentication-benefits-risks/ (дата звернення: 23.02.2025).

8. Biometric Authentication Benefits and Risks. Identity Management Institute. URL: https://identitymanagementinstitute.org/biometric-authentication-benefits-and-risks (дата звернення: 25.02.2025).

9. Challenges in Voice Biometrics: Vulnerabilities in the Age of Deepfakes.

ABA Banking Journal.URL: https://bankingjournal.aba.com/2024/02/challenges-in-voice-biometrics-vulnerabilities-in-the-age-of-deepfakes/ (дата звернення: 27.02.2025).

10. Voice Biometrics. ResearchGate. URL: https://www.researchgate.net/publication/27293606_Voice_Biometrics (дата звернення: 28.02.2025).

11. Voice Biometric Systems for User Identification and Authentication. ResearchGate. URL: https://www.researchgate.net/publication/360103263_Voice_Biometric_Systems_for_User_Identification_and_Authentication_-_A_Literature_Review (дата звернення: 02.03.2025).

12. The Disadvantages & Vulnerabilities of Voice Biometrics. iProov. URL: https://www.iproov.com/blog/disadvantages-vulnerabilities-voice-biometrics (дата звернення: 05.03.2025).

13. Penetration Testing and Security Vulnerabilities of Voice Recognition Technologies. Respeecher. URL: https://www.respeecher.com/blog/your-penetration-testing-security-vulnerabilities-voice-recognition-technologies (дата звернення: 08.03.2025).

14. Traditional Biometrics Are Vulnerable to Deepfakes. Reality Defender. URL: https://www.realitydefender.com/insights/traditional-biometrics-are-vulnerable-to-deepfakes (дата звернення: 10.03.2025).

15. Biometric Security Risks: Beyond Fingerprints and Facial Recognition. SOCRadar. URL: https://socradar.io/biometric-security-risks-beyond-fingerprints-and-facial-recognition/ (дата звернення: 10.03.2025).

16. Biometric Threats and Exploitation. Identity Management Institute. URL: https://identitymanagementinstitute.org/biometric-threats-and-exploitation/ (дата звернення: 13.03.2025).

17. Biometric Vulnerabilities: Ensuring Future Law Enforcement. Europol . URL: https://www.europol.europa.eu/publication-events/main-reports/biometric-

vulnerabilities-ensuring-future-law-enforcement-preparedness (дата звернення: 19.03.2025).

18. Navigating Biometric Data Security Risks in the Digital Age. Dark Reading. URL: https://www.darkreading.com/cyber-risk/navigating-biometric-data-security-risks-digital-age (дата звернення: 20.03.2025).

19. Using Biometrics. NCSC.GOV.UK. URL: https://www.ncsc.gov.uk/collection/device-security-guidance/policies-and-settings/using-biometrics (дата звернення: 24.03.2025).

20. Using Voice Biometric Authentication for Patient Privacy. Pindrop. URL: https://www.pindrop.com/article/voice-biometric-authentication-patient-privacy/ (дата звернення: 27.03.2025).

21. Voice Commands With The Arduino Speech Recognition Engine. Arduino Docs. URL: https://docs.arduino.cc/tutorials/portenta-h7/speech-recognition-engine/ (дата звернення: 30.03.2025).

22. How to Make Voice Control Door Lock. Arduino Project Hub. URL: https://projecthub.arduino.cc/munir03125344286/how-to-make-voice-control-door-lock-10b6c2 (дата звернення: 02.04.2025).

23. Controlling Devices Using Voice Recognition Module and Arduino. Quartz Components. URL: https://quartzcomponents.com/blogs/electronics-projects/controlling-devices-using-voice-recognition-module-and-arduino?srsltid=AfmBOoo2RK9XPwNnn3mFdOTsrAeQ6wfgz8yx9GR6tTwpgjSXu_9ILFq2 (дата звернення: 07.04.2025).

24. Voice Recognition Hardware Implementation Using Arduino UNO R3. GitHub. URL: https://github.com/Malayanil/VoiceRecognition-Arduino (дата звернення: 09.03.2025).

25. Fully Offline Voice Recognition Module with Arduino. Reddit. URL: https://www.reddit.com/r/arduino/comments/1ahj7p1/fully_offline_voice_recognition_module_with/ (дата звернення: 12.04.2025).

26. Recommendation Module for Voice Recognition Control. Arduino Forum. URL: https://forum.arduino.cc/t/recommendation-module-for-voice-recognition-control/1079530 (дата звернення: 15.04.2025).

27. VOSK Offline Speech Recognition API. Alpha Cephei. URL: https://alphacephei.com/vosk/ (дата звернення: 18.04.2025).

28. Vosk Models. Alpha Cephei. URL: https://alphacephei.com/vosk/models (дата звернення: 21.04.2025).

29. Vosk. Voxta Documentation. URL: https://doc.voxta.ai/docs/vosk/ (дата звернення: 24.02.2025).

30. Offline Speech Recognition on Android with VOSK. Alpha Cephei. URL: https://alphacephei.com/vosk/android (дата звернення: 27.04.2025).

31. Biometrick.Investopedia.URL:https://www.investopedia.com/terms/b/biometrics.as (дата звернення: 30.04.2025).

32. Equifax and the Perils of Password Protection. EQUIFAX. URL: https://www.equifax.com/newsroom/all-news/-/story/the-equifax-journey-to-passwordless-authentication (дата звернення: 05.05.2025).

33. A Cheap 3D Printer Can Trick Smartphone Fingerprint Locks. Wired. URL: https://www.wired.com/story/cheap-3d-printer-trick-smartphone-fingerprint-locks (дата звернення: 09.05.2025).

34. Hackers Trick Facial-Recognition Logins With Photos From Facebook. Wired. URL: https://www.wired.com/story/cheap-3d-printer-trick-smartphone-fingerprint-locks/ (дата звернення: 15.05.2025).

35. Why You Should Never Use Pattern Passwords on Your Phone. Wired. URL: https://www.wired.com/story/phone-lock-screen-password/ (дата звернення: 24.05.2025).

36. All Android and iPhone Users Warned to Delete 'Unfamiliar' Apps After Attackers 'Steal Faces' to Drain Bank Accounts. The Sun. URL: https://www.the-sun.com/tech/10529442/android-iphone-delete-apps-attackers-drain-back-accounts/ (дата звернення: 29.05.2025).

37. Biometrics and Privacy – Issues and Challenges. OVIC. URL: https://ovic.vic.gov.au/privacy/resources-for-organisations/biometrics-and-privacy-issues-and-challenges/ (дата звернення: 30.05.2025).

38. Voice Biometric Systems for User Identification and Authentication. ResearchGate.URL:https://www.researchgate.net/publication/360103263_Voice_Biometric_Systems_for_User_Identification_and_Authentication_-_A_Literature_Review (дата звернення: 01.06.2025).

39. Advanced Biometric Voice Verification for Two-Factor Authentication. MDPI.URL:https://www.researchgate.net/publication/373745479_Enhancing_Web_Application_Security_Advanced_Biometric_Voice_Verification_for_Two-Factor_Authentication (дата звернення: 03.05.2025).

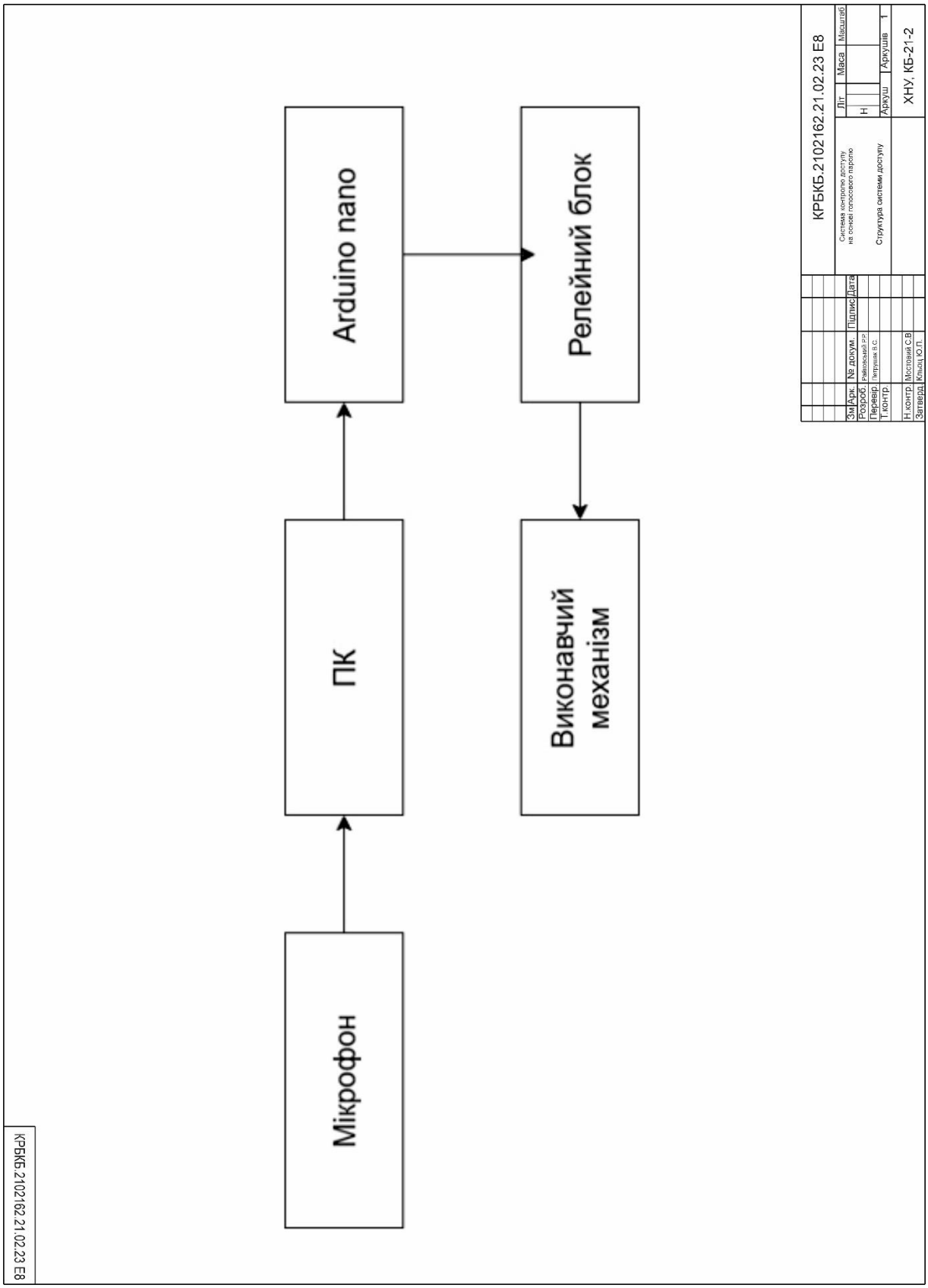
40. Voice Biometrics and Emerging Security Threats in the Voice Channe.SESTEK.URL:https://www.sestek.com/voice-technologies-and-cybersecurity-innovation-meets-protection-blog (дата звернення: 05.05.2025).

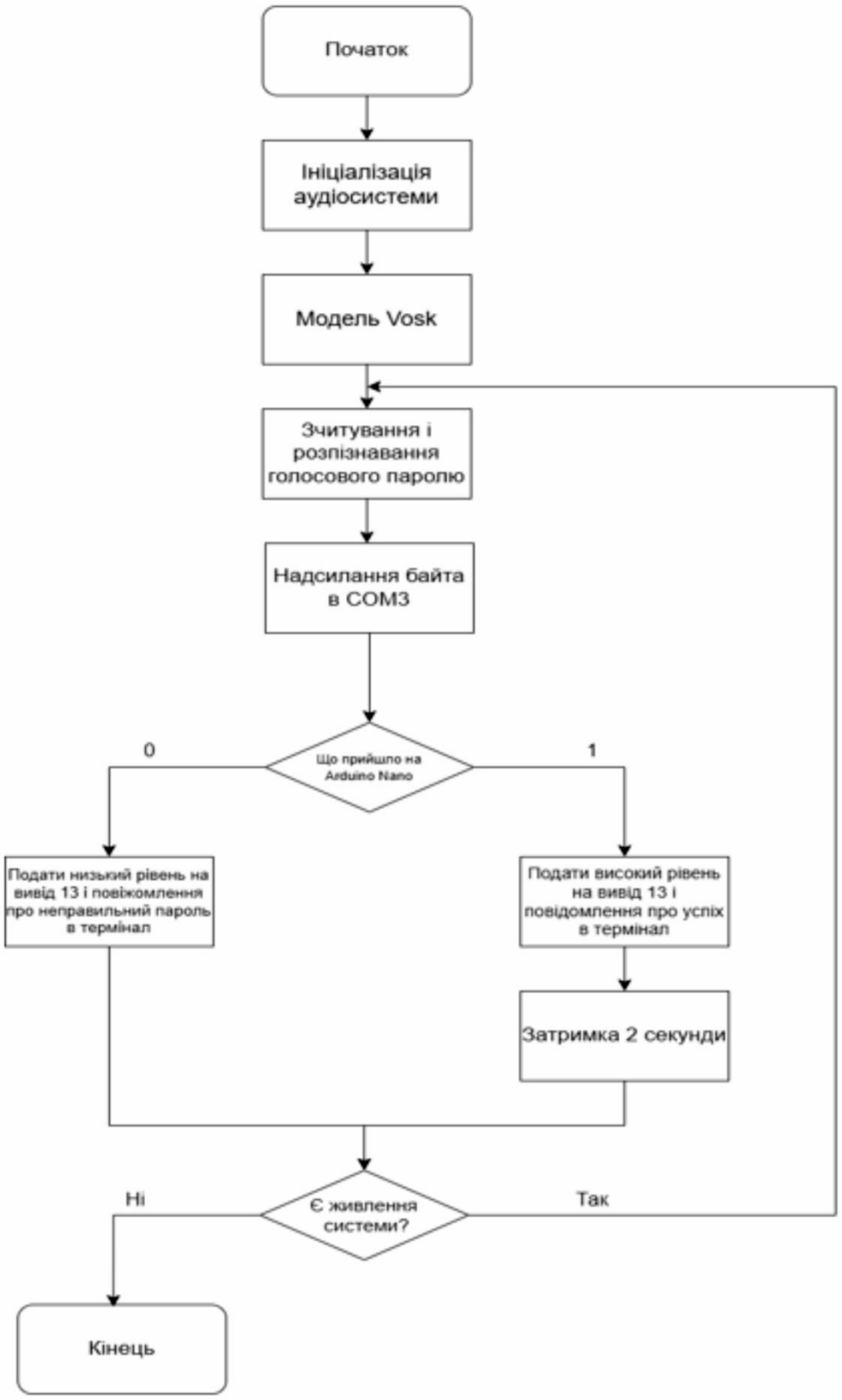
Додаток А
(Обов’язковий)
Програмний код

```
import vosk
import pyaudio
import serial
import json

model = vosk.Model(r"C:\Users\Roma\PycharmProjects\PythonProject3\vosk-model-uk-v3")

recognizer = vosk.KaldiRecognizer(model, 16000)

arduino = serial.Serial('COM3', 9600)

p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paInt16,
                channels=1,
                rate=16000,
                input=True,
                frames_per_buffer=8000)
stream.start_stream()

voice_password = "пароль"

print("🎤 Скажіть голосовий пароль для доступу...")

while True:
    data = stream.read(4000, exception_on_overflow=False)
    if recognizer.AcceptWaveform(data):
        result = json.loads(recognizer.Result())
        text = result.get("text", "").lower()
        print(" Розпізнано:", text)

        if voice_password in text:
            arduino.write(b'1')  # або будь-який символ доступу
            print("✅ Доступ дозволено — лампочка включена.")
        else:
            arduino.write(b'0')
            print("❌ Пароль неправильний.")
```

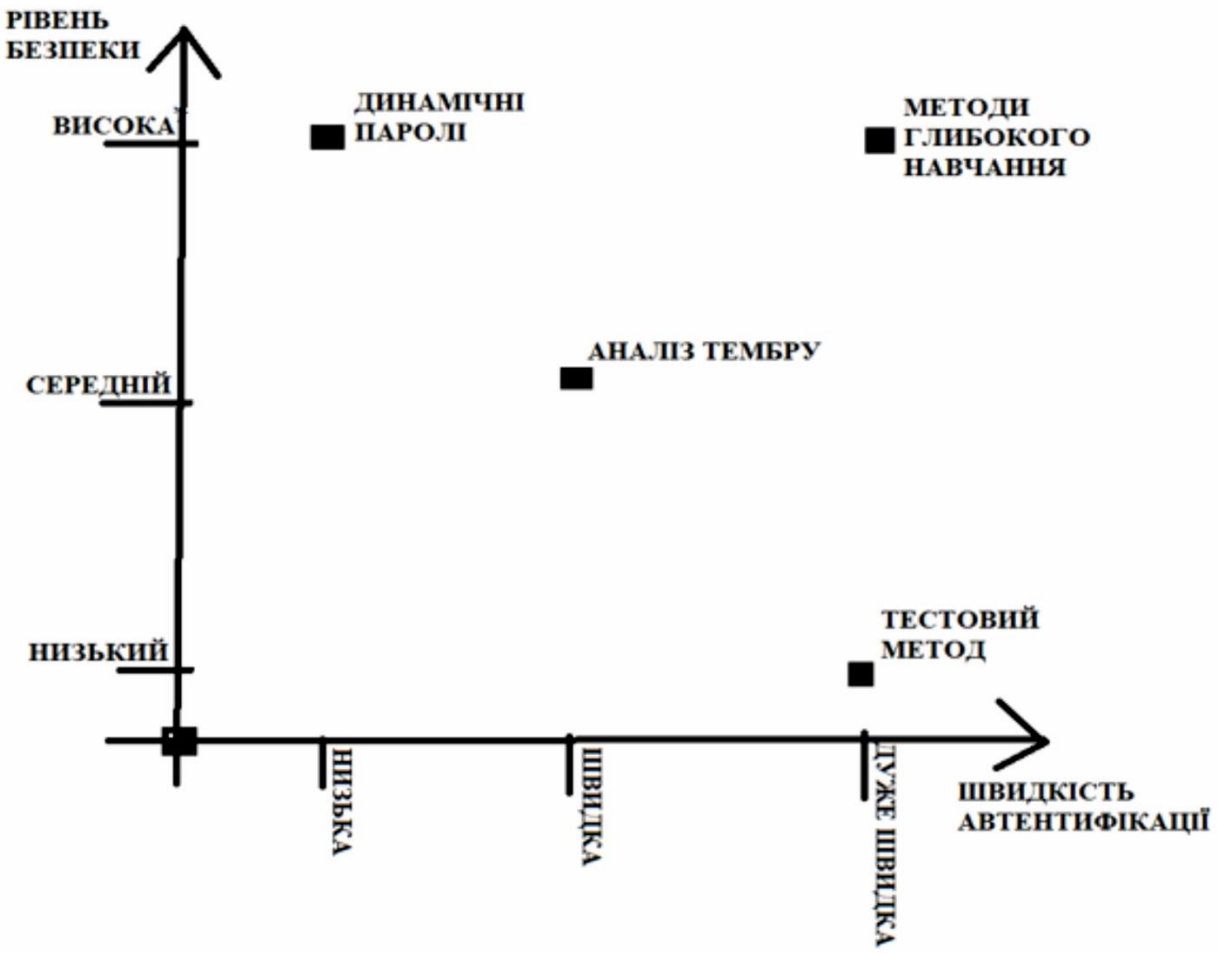
Додаток Б

(Обов’язковий)
Копії графічної частини





					КРКБ.2102162.21.02.23 Е8			
					Система контролю доступу на основі голосового паролю	Літ	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата		Н		
Розроб.		Райковський ЭР			Схема порівняння методів за швидкістю та рівнем безпеки			
Перевір.		Петрушак В.С.				Аркуш	Аркушів 1	
Т.контр.								
Н.контр.		Мостовий С.В.				ХНУ, КБ-21-2		
Затверд.		Кльоц Ю.П.						



					КРБКБ.2102162.21.02.23 Е8		
Зм	Арк.	№ докум.	Підпис	Дата	Система контролю доступу на основі голосового паролю	Літ	Маса / Масштаб
Розроб.		Райковський Р.Р.				Н	
Перевір.		Петрушак В.С.			Схема порівняння методів за швидкістю та рівнем безпеки	Аркуш	Аркушів 1
Т.контр.							
Н.контр.		Мостовий С.В.				ХНУ, КБ-21-2	
Затверд.		Кльоц Ю.П.					

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Райковський Роман Русланович
ПІБ здобувача вищої освіти

Студента ФІТ, 4 курсу, групи КБ-21-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

09.06.2025 Р

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 12%**

	Document		Sum coincidence on the DB	
ID: 244353 Title: Система контролювання доступу на основі голосового паролю Added in a DB: 2025-06-09 Authors: Райковський Роман Русланович Heads: Петрушак В.С. Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	108035	619	535 (0%)	6 (1%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Райковський Роман Русланович

Співавтор:

Назва: Система контролювання доступу на основі голосового паролю

Науковий керівник:

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:0.7%

Коефіцієнт подібності 2:0.2%

Мікропробіли: 0

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-09 23:56:44.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

10.06.2025р.

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система контролювання доступу на основі голосового паролю

Автор: Райковський Роман Русланович

Спеціальність: 125 – Кібербезпека

Освітня програма:Кібербезпека

Науковий керівник:Володимир ПЕТРУШАК, канд. техн. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1. запозичення розміщені в розділах аналізу існуючих методів та технологій, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2. усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3. всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту. Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості,

складає 0.7%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Володимир ПЕТРУШАК

Завідувач кафедри Кб _____ Юрій КЛЬОЦ

Гарант ОП _____ Віктор ЧЕШУН

Дата:

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студентка Райковський Роман Русланович

Тема Система контролювання доступу на основі голосового паролю

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 3; кількість сторінок записки 69.

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі, відповідно до поставленого завдання, проведено дослідження предметної області біометричної автентифікації, зокрема голосової ідентифікації. Проаналізовано законодавчу базу у сфері захисту персональних даних, а також типові загрози інформаційній безпеці при використанні голосових паролів. Проведено моделювання потенційних загроз та типових порушників. Розроблено архітектуру системи контролю доступу з використанням мікроконтролера Arduino та програмного забезпечення на Python із застосуванням бібліотеки Vosk для офлайн-розпізнавання голосу. Сформовано політики безпеки, що враховують автономну роботу системи, мінімізацію обробки біометричних даних та можливість подальшого масштабування рішення.

2. Висновок про відповідність кваліфікаційної роботи завданню У кваліфікаційній роботі повністю виконано поставлене завдання як у теоретичній, так і в практичній частині

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У розділі 1 виконано аналіз сучасних систем контролю доступу та біометричної автентифікації з акцентом на голосову ідентифікацію: розглянуто історичний розвиток голосових біометричних систем (GMM-HMM, і-вектори, х-вектори), сучасні нейромережеві підходи та механізми виявлення живості, а також проаналізовано основні загрози (replay-атаки, deepfake) і правові норми обробки біометричних даних за законодавством України. У розділі 2 обґрунтовано вибір апаратної платформи Arduino Nano і рушія Vosk для офлайн-розпізнавання голосу, описано архітектуру системи, алгоритм взаємодії Python-скрипта з Arduino через COM-порт та налаштування політик безпеки, що виключають зберігання аудіо та забезпечують захищену передачу команд. У розділі 3 детально викладено процес практичної реалізації прототипу: підключення мікрофонного модуля, складання електронної схеми, написання та тестування коду на Python і C++ для Arduino, проведено експериментальні випробування на точність і стійкість до фонових шумів і помилкових спрацьовувань, а також розроблено рекомендації з експлуатації, масштабування системи та можливого розширення функціоналу (наприклад, інтеграція багатофакторної автентифікації). У роботі використано передові досягнення в галузі штучного інтелекту, обробки мовлення та мікроконтролерної техніки, що забезпечило створення економічно ефективного, автономного й безпечного рішення для контролю доступу на основі голосового пароля.

4. Позитивні сторони роботи: Робота базується на детальному вивченні вимог нормативних документів і законів України щодо побудови та супроводу систем захисту інформації, зокрема біометричних та IoT-рішень. Практична цінність дипломної роботи полягає в розробці автономної системи голосового контролю доступу на базі Arduino

Nano і Python/Vosk, яка дозволяє підвищити рівень конфіденційності, мінімізувати витоки біометричних даних і легко інтегрується в існуючу інфраструктуру Інтернету речей підприємства.

5. Негативні сторони роботи: У роботі недостатньо проаналізовано технічні характеристики використовуваного обладнання: не наведено деталі щодо вибору моделі й чутливості мікрофона, частоти дискретизації аудіо та налаштувань COM-порту Arduino Nano, що ускладнює оцінку точності й надійності розпізнавання, а також бракує глибокої деталізації прийнятих проєктних рішень – відсутні розрахунки затримок обробки аудіопотоку і передачі команд на Arduino, обґрунтування параметрів Vosk-моделі та налаштувань буферів для стабільної роботи системи.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення кваліфікаційної роботи відповідає темі роботи та виконане з дотриманням стандартів. В цілому, графічне оформлення є якісним, а пояснювальна записка відповідає нормам оформлення.

7. Відгук про роботу в цілому: Дипломна робота заслуговує позитивної оцінки, оскільки весь матеріал викладено чітко, структуровано та логічно послідовно. Усі розділи—від аналізу сучасних методів голосової автентифікації до практичної реалізації прототипу на базі Arduino Nano та Python із використанням Vosk—будують єдину повноцінну картину дослідження. Графічні схеми та діаграми ефективно ілюструють архітектуру системи, алгоритми обробки голосових команд і взаємодію між апаратурою та софтом, що дозволяє легко оцінити доцільність і ефективність прийнятих рішень у проєктуванні та супроводі голосового контролю доступу.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи: Враховуючи, що в роботі є недоліки щодо деталізації технічних характеристик апаратури та обґрунтування затримок обробки аудіо й передачі команд на Arduino, що обмежує повноту оцінки продуктивності та надійності голосової системи контролю доступу, можна обґрунтувати виставлення оцінки «добре» .

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Гула Ігор Володимирович
професор кафедри фізики і електротехніки, кандидат технічних наук, доцент

« 12 » 06 2025. Гула І.В. доцент кафедри фізики та електротехніки