Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Утюжов А.С.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Марченко К.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Утюжову Андрію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi*

2. Керівник роботи *Марченко Костянтин Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Марченко К.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Утюжов А.С.
(прізвище та ініціали)

АНОТАЦІЯ

Утюжов А.С. Програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

Метою розробки є програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

Результат роботи – програмна реалізація системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, цифрова трансформація, Wi-Fi

ABSTRACT

Utyuzhov A.S. Software for a video technology system for digital transformation with Wi-Fi support. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a video technology system for digital transformation with Wi-Fi support.

The purpose of the development is software for a video technology system for digital transformation with Wi-Fi support.

The result of the work is a software implementation of a video technology system for digital transformation with Wi-Fi support.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Python environment.

Keywords: computer engineering, digital transformation, Wi-Fi

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВКЗ	–	відео-конференц-зв’язок
ПЗ	–	програмне забезпечення
CMP	–	Codian Management Platform
DNS	–	Domain Name System
GMS	–	Global Management System
GRE	–	Generic Routing Encapsulation
ISDN	–	Integrated Services Digital Network
MCU	–	Multipoint Control Unit
MPLS	–	Multiprotocol Label Switching
MXM	–	Media eXchange Manager
SSL	–	Secure Socket Layer
TLS	–	Transport Layer Security
TMS	–	Tandberg Management Suite
VPN	–	Virtual Private Network
USENET	–	User Network

ВСТУП

Актуальність теми. Для реконструкції старого конференц-залу, багатоцільового залу, залу для випуску на місці, тимчасового місця проведення конференцій тощо бездротову конференц-систему Wi-Fi можна швидко та легко налаштувати та видалити після використання.

Для цього розробляємо бездротову конференц-систему 5G, щоб забезпечити належне рішення для вищевказаних випадків.

Особливості:

– Підтримуйте свій стіл чистим і охайним без будь-яких кабелів.

– Бездротова конференц-система та бездротова система презентацій забезпечують зручні локальні рішення для доступу до аудіо та відео.

– Швидкий обмін одним клацанням миші, що дозволяє учасникам приділити більше уваги самій зустрічі.

– Надвелика відстань захоплення звуку 60 см.

– Можливість проведення ефективних зустрічей.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

– Дослідження системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

– Програмна реалізація системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

На основі сенсорної технології та технології зв’язку Wi-Fi компанія успішно запустила бездротову конференц-систему 5G Wi-Fi завдяки незалежним дослідженням і розробкам. Дизайн конференц-блока має патентований зовнішній вигляд і унікальний вигнутий сенсорний екран, який не тільки збільшує кут огляду, але також покращує відчуття текстури, технології.

Загальна затримка системи становить 15 мс, що набагато менше затримки в 24 мс, яку може розпізнати людське вухо. Бездротова конференц-система приймає 48K, 16-бітну нестиснену аудіопередачу, здатну передавати 8 каналів голосу та керуючих сигналів одночасно. На основі технології він пропонує клієнтам бездротове обговорення Wi-Fi, бездротовий вхід Wi-Fi, бездротове голосування Wi-Fi, бездротову одночасну передачу Wi-Fi, бездротове інтелектуальне керування, відображення дати годинника та інші функції.

1.2 Область застосування

Персональні відеоконференції

Для звичайного користувача зараз відбувається інтенсивний розвиток високошвидкісного підключення до мережі Інтернет у його домі, на відпочинку або навіть у шляху, що відкриває широкі можливості використання засобів і технології відеозв'язку для організації точкових відеоконференцій за допомогою веб-камери й персонального комп'ютера. Крім того, зараз значно знизилися ціни й на апаратні пристрої відео-конференц-зв’язку, що робить використання відеоконференцій у звичайному житті набагато більше доступним.

Відеоконференції для бізнесу

При грамотному впровадженні, відеоконференції відразу почнуть приносити відчутні фінансові й адміністративні переваги великим і розподіленим організаціям, а так само тим організаціям, які використовують віддалену робочу силу. Відеоконференції підвищують продуктивність, ефективність діяльності, конкурентні переваги й, в остаточному підсумку, прибуток компанії.

Відеоконференції дозволяють прискорити процес прийняття рішень, збільшити доступ до інформації й фахівців, скоротити час відряджень (і, відповідно, витрати й стрес від поїздок), збільшити продуктивність праці в цілому, підвищити ефективність розподілу ресурсів. Крім того, скорочується час виводу продуктів на ринок і час реагування на його зміни.

Відеоконференції для утворення

Відеоконференція дозволяє набагато спростити процес дистанційного утворення з використанням двосторонньої(точка-точка) відеозв'язку. Більше того, учителі й учні із усього світла можу виявитися в одній аудиторії (багатоточкові відеоконференції, селекторні наради, а також веб-конференції). З використанням технології відео-конференц-зв’язку студенти можуть відвідати практично будь-яку точку на світі не залишаючи аудиторії, спілкуватися з іншими учнями й викладачами. Віртуальні класи дозволяють учням, особливо ізольованим географічно або економічно, одержувати знання доступні тільки в кращих навчальних закладах. Школи можуть використовувати цю технологію, наприклад, для навчання дітей мовам, які недоступні в їхньому регіоні.

Відеоконференції в медицині

Відеоконференції можуть дуже ефективно застосовуватися для віддаленої діагностики, лікування й консультацій. При цьому, найважливішою перевагою використання відеоконференцій у медицині є безперервне утворення й підвищення кваліфікації медичного персоналу. Існує навіть спеціальний термін для цього напрямку використання відеоконференцій – телемедицина.

Для проведення діагностики й дистанційного навчання широко поширена можливість під час відеоконференції передавати зображення від цифрових камер над операційним столом, мікроскопом, відео-ендоскопом і т. п. Так само можна передавати й телеметричну інформацію: частоту пульсу, тиск і т.д.

Застосування відеоконференцій у медицині дозволить значно підвищити здоров'я людей, що багато в чому залежить від доступності і якості медичних послуг. Доступність же медичних послуг підвищується за рахунок збільшення кваліфікації персоналу, а так само за рахунок залучення за допомогою відеоконференцій вузькопрофільних фахівців і інститутів до діагностики й лікування.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Polycom

При розробці нових і розвитку наявних продуктів Polycom концентрується на забезпеченні максимальної простоти й зручності взаємодії із системами ВКЗ. Цьому сприяють різні засоби автоматизації. Наприклад, система EagleEye Producer здатна автоматично знаходити всіх учасників дискусії й масштабувати зображення мовця. Недавно в систему була додана можливість визначити, чи є присутньою людина в приміщенні, і подати команду на включення всіх необхідних для проведення сеансу ВКЗ пристроїв. Таким чином, людині досить увійти в приміщення, а система ВКЗ автоматично все підготує до роботи. У планах Polycom – реалізація функції розпізнавання осіб. У цьому випадку система зможе ідентифікувати користувача й завантажити його персональні налаштування (інформацію про зустрічі та ін.).

Ще одне рішення, що автоматизує проведення сеансів ВКЗ, – EagleEye Director з функцією наведення камери за голосом. У червні 2025 року на ринку з'явиться друге покоління цього продукту. Polycom EagleEye Director II буде оснащуватися новими камерами, що підтримують дозвіл 4K. Крім того, система зможе визначати, скільки чоловік перебуває в кімнаті, – це важливо для одержання статистики й оцінки економічної ефективності використання ВКЗ. Polycom планує реалізувати каскадування пристроїв EagleEye Director, щоб засобами автоматичного наведення за голосом можна було охопити приміщення більшої площі.

Компанія Polycom познайомила з інноваційною системою відео-конференц-зв'язку RealPresence Centro. Ця система оснащена камерою з кутом огляду 360°, чотирма розташованими по колу сенсорними екранами, мікрофонами високої чутливості, а також засобами придушення шуму. На відміну від офіціозу традиційних конференц-залів, система RealPresence Centro створює креативний простір, настільки важливе для мозкового штурму.

Рисунок 2.1 – Інноваційна система відео-конференц-зв'язку Polycom RealPresence Centro

Системи компанії підтримують технологію, що дозволяє придушувати сторонні шуми: ви можете друкувати на клавіатурі або шарудіти папером, але на віддаленій стороні чутно цих звуків не буде. Але коли ви починаєте говорити, мікрофони активуються й ваш голос прекрасно почують віддалені співрозмовники. Інше рішення (Acoustic Fence) оточує учасників сеансу ВКЗ акустичним бар'єром, що дуже важливо для комфортного проведення конференцій в офісах з відкритим плануванням.

Yealink

Китайська Yealink – відносно молодий гравець на ринку ВКЗ. Yealink, будучи одним з лідерів ринку IP-телефонів, цей виробник зміг і в рішеннях ВКЗ втілити такі переваги, як простота використання, доступна ціна при високій якості, а також принцип «усе включено» (платиш один раз і одержуєш повний набір можливостей – ніяких додаткових ліцензій купувати не треба). Сьогодні в арсеналі компанії є програмні ВКЗ-клієнти (у тому числі для мобільних пристроїв) і апаратні термінали. Серед продуктів компанії – рішення «усе в одному» VC110 (кодек інтегрований у камеру), а також термінал MC120 з убудованим MCU на вісім точок. Якщо такої ємності MCU не вистачає, то можна скористатися відомими хмарними сервісами (Bluejeans, Zoom, Vidyo.

Рисунок 2.2 – Рішення для ВКЗ «усе в одному» Yealink VC110 (кодек ВКЗ і камера в одному пристрої)

У найближчих планах компанії – випуск персонального відеотермінала T58V із сенсорним дисплеєм 7″. В II кварталі 2025 року Yealink обіцяє випустити ВКЗ-термінал для більших і середніх переговірних VC800 з убудованим MCU на 25 точок. Крім того, на цей рік запланований випуск терміналів VC500 для малих і середніх переговірних і VC200 – для міні-переговірних. Ще одна очікувана новинка – програмний MCU (Yealink Meeting Server), що споконвічно буде підтримувати функції реєстрації, проходження через NAT і зберігання контактів. Під кінець року обіцяна реалізація підтримки Skype for Business, WebRTC і потокової передачі відео.

Vinteo

Незважаючи на свою молодість, компанія Vinteo має в арсеналі всі основні категорії продуктів ВКЗ. Її флагман – це програмний сервер ВКЗ, що підтримує одночасне підключення до 1000 учасників. Рішення побудоване за класичною схемою, коли мікшування відеопотоків здійснюється на стороні сервера (а не на стороні клієнта), при цьому ПЗ Vinteo Video Core працює на стандартних серверах x86. Завдяки підтримці загальноприйнятих стандартів (H.323, SIP, WebRTC і т.д.), сервер Vinteo практично повністю сполучимо з устаткуванням інших вендорів, включаючи лідерів галузі.

Продуктивність процесорів досягла такого рівня, що обробка відео на звичайному сервері (для організації MCU) або ПК (клієнт ВКЗ) більше не є чимсь фантастичним. Починаючи з 2013 року, усе більше виробників знімають із продажу спеціалізовані пристрої MCU і переводять виконувані ними функції на сервери x86, які можуть розміщатися як на площадці замовника, так і в хмарі».

У реалізованих проектах сервер Vinteo, як правило, використовується для організації взаємодії із ВКЗ-терміналами й серверами інших виробників. Однак компанія недавно представила й власні клієнтські продукти. Це програмний клієнт Vinteo Desktop з підтримкою дозволу 1080p, а також апаратні клієнтські термінали Vinteo T і Vinteo ST. Останній підтримує дозвіл аж до 4K и має убудований сервер MCU, що забезпечує одночасне підключення до 20 учасників.

Рисунок 2.3 – Термінал Vinteo T

GrandStream

ВКЗ-термінали «усе в одному» на базі Android стають усе популярніше. Компанія GrandStream представила термінал GVC3200 з убудованим MCU, що підтримує до чотирьох учасників (1080p). Камера термінала забезпечує 12-кратне збільшення. Пристрій оснащений трьома виходами HDMI і поставляється в комплекті зі спікерфоном. Модель GVC3202 відрізняється від GVC3200 тим, що має убудований MCU тільки на три точки (720p) і два виходи HDMI.

Крім того, GrandStream представила хмарний сервіс для відео-, аудіо- і Web-конференцій IPVideoTalk. Як термінали для роботи через хмару IPVideoTalk можуть використовуватися вже згадані GVC3200 і GVC3202. Крім того, сервіс підтримує технологію WebRTC, а виходить, доступний з будь-якого сумісного з даною технологією браузера.

Система відеоконференцій бізнес-класу LifeSize Icon Flex

Microsoft Lync, Skype, Google Hangouts, Cisco jabber. Досить підключити Flex до ПК або MAC за допомогою USB-кабелю – і ви одержите чудову систему

відеозв'язку, де кожний учасник відмінно чує й бачить співрозмовників. LifeSize Icon Flex – система відеозв'язку бізнес-класу, що ідеально підходить для невеликих переговірних кімнат.

Можливості термінала LifeSize Icon Flex:

– Компактний розмір.

– Призначена для використання разом з додатками Microsoft Lync, Google Hangouts, Cisco Jabber, Skype або LifeSize Cloud.

– Апаратне забезпечення, відзначене багатьох галузевих нагород, забезпечує виняткова якість нарад.

– Створена з урахуванням досвіду обслуговування клієнтів світового рівня й забезпечується кваліфікованою сервісною підтримкою.

– Камера LifeSize забезпечує передачу чудового HD-Відео 1080p30/720p60.

– Малошумний привод нахилу/повороту з 6-кратним зумом: трикратне оптичне збільшення й 2-кратний цифровий зум.

– LifeSize Phone – засіб керування конференц-зв'язком, ефективне й з елегантним дизайном.

Поліпшений LifeSize Cloud Користувачі «хмарного» сервісу LifeSize одержали можливість підключатися до конференцій за допомогою веб-браузерів, що підтримують використання WebRTC, Microsoft Lync, а також через Google Calendar і Microsoft Outlook. У число цих браузерів входять Firefox, Internet Explorer, Google Chrome і Safari. У тому випадку, якщо браузер не підтримує убудований WebRTC, для підключення можна використовувати плагін.

Компактна система відеозв'язку LifeSize Icon 400

Можливості термінала LifeSize Icon 400:

– Підключення до сервісу LifeSize Cloud дозволяє з бездоганною якістю проводити багатоточкові виклики, у тому числі прямо з корпоративних довідників, ефективно обходити міжмережні екрани й NAT, одержувати автоматичні відновлення програмного забезпечення.

– Простота масштабування: щоб додати додаткових учасників у конференцію, досить просто прийняти вхідний виклик або нажати на контакт у каталозі.

– Запис відеоконференцій і передача контенту великому числу учасників виробляється оперативно й інтуїтивно зрозуміло.

– Багатоточкові конференції можна почати з єдиної директорії. Необхідно просто знайти ім'я, кликнути на нього й зробити виклик.

– Спрощена система керування й інтуїтивно зрозумілий користувальницький інтерфейс. Значки екранного меню чітко передають функції й операції. Ви можете оперативно зробити необхідні дії.

– Швидкий старт відеоконференцій можливий завдяки можливості формування груп вибраних контактів.

– LifeSize Icon 400 легко встановлюється й настроюється.

– Якість відео до 1080p30 і камера з 6-кратним збільшенням.

– Потрійний мікрофон з охватом в 360 градусів і технологією цифрового формування діаграми спрямованості, що дозволяє передавати кристально чистий звук.

– Як і всі продукти LifeSize Icon серії, Icon 400 створена з урахуванням всіх вимог по простоті використання, гнучкості й цінності для клієнтів.

Система відеоконференцій LifeSize Icon 600

LifeSize Icon являє собою лінійку високопродуктивних відеосистем, що підтримують функцію інтелектуального відео – відеоконференції, які радикально спрощені й повністю захищені від зривів.

Система LifeSize Icon відмінно підходить для будь-якої переговірної кімнати й настільки проста у використанні, що людина, що ввійшла в кімнату, може підключитися до відеоконференції всього за кілька секунд одним натисканням кнопки. Завдяки таким функціям, як зведені розклади нарад і спливаючі нагадування, ви ніколи не пропустите чергову зустріч. Інтегровані списки нарад дозволяють швидко знаходити наради й підключатися до них за

запитом. А каталоги на основі імен і розширені функції пошуку забезпечують швидкий і простий доступ до людини, з яким потрібно зв'язатися.

Використовуйте надійні відеододатки, наприклад відеоцентр LifeSize UVC Video Center, що дозволяє записувати й передавати презентації, або LifeSize UVC Multipoint для підтримки групових викликів на будь-якому пристрої за допомогою платформи LifeSize UVC Platform. Головна перевага системи LifeSize Icon полягає в тому, що доступ до всіх цих додатків можна одержати простим натисканням кнопки. Тому технологічна сторона конференції вимагає менше уваги, що дозволяє більше часу приділяти рішенню основних завдань. Система дуже проста у використанні, і ваші IT-співробітники неодмінно оцінять зменшення потреби в навчанні й підтримці.

Завдяки функції інтелектуального відео, підтримуваною серією LifeSize Icon, все просто працює – бездоганно, інтуїтивно й з неперевершеними характеристиками. Миттєвий зв'язок із клієнтами, партнерами й співробітниками. Це інтелектуальне відео для нового, більше ефективного й зв'язаного миру.

Простота використання

Системи LifeSize Icon розроблені так, щоб максимально спростити їхнє використання. На екрані в простій і наочній формі представлені всі функції, необхідні для встановлення надійного зв'язку.

Розширюваність і гнучкість

Завдяки закладеній при розробці можливості адаптації до змін потреб компанії ви в будь-який момент можете обновити програмне забезпечення й приналежності. Потрібно чи розширити середовище з декількома дисплеями, підвищити продуктивність або додати підтримку нових функцій, наприклад запису або групових викликів, – серія LifeSize Icon забезпечить захист інвестицій і відповідність майбутнім потребам вашої зростаючої компанії.

Інтеграція із платформою LifeSize UVC Platform

Ознайомтеся з перевагами потужних інтегрованих інфраструктурних додатків на платформі LifeSize UVC Platform. Спрощене налаштування й

підготовка до роботи дозволяють IT-фахівцям включати функції потокової передачі й запису, управляти мережею, безпечно обходити міжмережні екрани й NAT, підтримувати групові відеовиклики й у міру необхідності задіяти нові функціональні можливості передачі відео.

Висока продуктивність і потужність

Оцініте можливості найсучаснішого й високопродуктивного з доступних рішень для відеоконференцій високої чіткості. Підтримка відео у форматі HD 1080p60 забезпечує максимальну чіткість і плавність рухів, а можливості для презентацій у форматі 1080p дозволяють домогтися неперевершеної деталізації для успішної спільної роботи.

Система для Full HD відеоконференцій LifeSize Team 220

На сьогоднішній день різним групам співробітників необхідні регулярні зустрічі й переговори віч-на-віч для досягнення потребуючі напруги сил проектних цілей. Наявність можливостей ефективно взаємодіяти й швидко приймати важливі рішення є першорядно значимим для успіху.

LifeSize Team поєднує невеликі групи й організації, дозволяючи їм ефективно взаємодіяти на відстані. Будь те найменші зміни міміки або положення тіла, LifeSize Team забезпечує об'єднання й розуміння один одного всіма учасниками конференції.

Компанія LifeSize розробила абсолютно просту у використанні систему відео-комунікацій. Зробите виклик віддалених учасників за допомогою інтегрованого спікерфона LifeSize Phone™ і приєднаєте комп'ютери й камери, щоб представити колегам мультимедійні файли або документи, а також все чітко побачити й почути.

Якість високої чіткості. Простота. Значимі переговори між групами. LifeSize Team – це така система відео-комунікацій, яка вона повинна бути насправді.

Переваги LifeSize Team:

– Відео високої чіткості – дозвіл 1280x720, 30 кадрів у секунду й кращу якість при будь-якій розв'язній здатності.

– Аудіо високої чіткості – повністю інтегрований спікерфон із широкою смугою частот забезпечує звук до 16 кГц.

– Простий у використанні – інтерфейс, що розуміється інтуїтивно, і убудоване керівництво дозволяють вступити в конференцію новим учасникам також просто, як зробити виклик по телефоні.

– Заснована на стандартах – H.264 і H.263 для функціональної сумісності із системами інших виробників.

– Двопотоківість – представляйте презентації й спостерігайте одночасно за учасниками; використовується протокол H.239.

– Повністю інтегрована система – самоконфігуруєма система з кодеком високої чіткості, пристроєм «голосної» зв'язку, камерою й дистанційним керуванням.

– Широкополосне з'єднання – підключення до IP-мереж до 2.5 Мбіт/с.

– Централізоване керування – інтеграція програмного забезпечення LifeSize Control для керування, підтримка web-інтерфейсу й протоколу керування мережею SNMP.

Система відеозв'язку формату Full HD LifeSize Room 220

Як і всі продукти компанії LifeSize, LifeSize Room – це встаткування, за допомогою якого створюється повний ефект присутності, а також звук і зображення високої чіткості для створення реалістичного, достовірного сприйняття. За допомогою його ви відчуєте свою присутність у тім місці, з яким ви встановили зв'язок; ви зможете встигати більше, менше подорожуючи.

Ідеальне рішення для більших конференц-залів. LifeSize Room дозволяє встановити відеозв'язок з декількома учасниками за допомогою убудованого сервера багатоточкових HD конференцій. На екрані можна буде бачити декількох учасників одночасно (4-х стороння одночасна присутність) або бачити

автоматично, що переміняються картинки, що виступають учасників в 6-ти сторонньої конференції в режимі активізації за голосом. Все це можна одержати без додаткових зовнішніх пристроїв, складної системи планування або допомоги технічного фахівця.

Система LifeSize Room є дивно гнучкою, оскільки підтримує два монітори й дві камери. За допомогою даного встаткування всі учасники зможуть бачити презентації, одержувати документи й інші мультимедійні файли, що дозволить розподіляти інформацію між колегами й у такий спосіб взаємодіяти більш ефективно.

Компанія LifeSize розробила дуже прості в використанні системи відео-комунікацій. Вдайтеся до допомоги відео-комунікацій для проведення особистих зустрічей у тому випадку, коли електронних листів і телефонних переговорів не досить.

Функції й можливості:

– Відеоконференції високої чіткості (1280 x 720, 30 кадрів у секунду).

– Підтримувана швидкість від 128Kbps до 5 Mbps.

– Підтримка конфігурації з одним або двома пристроями відображення.

– Одна або дві керовані PTZ відеокамери високої чіткості.

– Інтегрований конференцфон високої чіткості.

– Підтримка протоколів H.264, H.263 і H.239.

– Можливість включення багатоточкової відеоконференцій із чотирма учасниками.

– Простий інтерфейс керування.

– Керування з використанням LifeSize Control, включаючи підтримку планування конференцій в Microsoft Outlook.

– Сумісність із ISDN абонентами зі шлюзом LifeSize Networker.

Система керування видеовідео-конференц-зв’язком SMC 2.0..

Забезпечує керування сервісами й пристроями відео-конференц-зв'язку, підтримує встаткування Huawei і пристрої інших виробників. Service Management

Center 2.0 (SMC 2.0) пропонує користувачеві простий інтерфейс для організації й керування конференцією, включаючи конференції, що вимагають проходження через приватні мережі й мережі загального користування. У той же час, SMC надає широкий вибір функцій системного адміністратора для керування пристроями в мережах великого масштабу.

Повна підтримка ресурсів відео-конференц-зв'язку зі зрозумілим веб-інтерфейсом SMC і відмінними технологіями QoS.

Функціональні можливості:

Уніфіковане керування й централізований контроль значно спрощують проведення відеоконференцій.

– Уніфікована система керування ресурсами відео-конференц-зв'язку забезпечує взаємодію й безперебійну роботу пристроїв.

– Ієрархічна автентифікація безлічі користувачів на основі ролей спрощує організацію конференції навіть для більших організацій.

– Компоненти гейткіпера H.323 і SIP сервера дозволяють створити гнучку систему й управляти їй.

– Стандартизовані порти дозволяють легко інтегруватися із системами інших виробників.

– Забезпечення безпеки на рівні додатка, системи, мережі й керування.

Сервер SwitchCenter для взаємодії мереж, розділених міжмережними екранами

Надає безперебійний відеозв'язок між мережами за допомогою реєстрації H.323 і протоколу ініціації сесій (SIP), а також за допомогою декількох протоколів передачі трафіку через міжмережні екрани. Останні включають H.460 на базі H.323 і протокол інтерактивного встановлення з'єднання (ICE) на базі SIP, а також утиліти для транслятора мережних адрес (STUN).

Працюючи на базі сервера Huawei під Linux, ПЗ SwitchCenter підтримує взаємодія між публічною й приватною мережами.

SwitchCenter здатний підбудовуватися під різні вимоги відеозв'язку.

Забезпечте високоякісну видеоконференцсвязь із надійним підключенням:

– Доступ з подвійним протоколом H.323/SIP і підтримка гейткіпера/сервера SIP/проксі SIP пропонують різнобічні послуги.

– Потужні можливості обходу брандмауера дозволяють реалізувати відеозв'язок і взаємодія між різними мережами.

– Зручний веб-інтерфейс системи керування послугами (SMC) відеоконференції Huawei забезпечує простоту конфігурації, моніторингу й техобслуговування декількох серверів SwitchCenter.

Платформа послуг відео-конференц-зв’язку з універсальним транскодуванням серії VP9600

Перший у галузі MCU забезпечує можливість повного кодування й декодування медіапотоків 1080p60. Гнучка й масштабована структура за рахунок конфігуруємих портів забезпечує підтримку від Full HD до CIF, а також підключення ISDN і VoIP.

Передові технології обробки відео в сполученні з динамічним керуванням смугою пропущення й технологіями корекції помилок спрощують проведення відеоконференцій і знижують витрати.

Повна інтеграція з мультимедійними системами відео-конференц-зв’язку Huawei, а також зі стандартними пристроями інших виробників забезпечує безпечне включення третіх сторін у конференцію й оптимальне повернення інвестицій.

Легко конфігуруємий і масштабований сервер багатоточкової конференції серії VP9600 компанії Huawei з універсальним транскодуванням 1080p60:

– Універсальне транскодування й постійна присутність на кожному порту забезпечує керування якістю відтворення відео й використанням ресурсів смуги пропущення для оптимізації роботи кожного учасника.

– Інтелектуальне керування швидкістю передачі (IRC) і механізм придушення помилок (SEC) забезпечують плавне відтворення відео HD з можливістю маскування до 20% втрат пакетів.

– Порти, що налаштовуються легко:

1 x 1080p60 = 2 x 1080p30 = 4 x 720p = 8 x SD

– Устаткування операторського класу, оснащене шістьома механізмами резервування, забезпечує стабільне й надійне виконання операцій; середній час наробітку на відмову більше 100 000 годин; середній час відновлення працездатності менш 30 хвилин.

– Відкрита стандартна структура й інтерфейси для підключення систем уніфікованого зв'язку й взаємодії (UC&C) інших виробників; проста інтеграція з Huawei eSpace, Microsoft Lync 2010™, Microsoft Lync 2013™, IBM SameTime™ і Skype™.

Пристрій запису й потокової передачі відео 1080p60 RSE6500

Високопродуктивний пристрій запису й потокової передачі full-HD підтримує багатоточкову відеотрансляцію й послугу відео за запитом (VoD). Підтримує потокову передачу відеотрансляції 1080p60 для 2000 глядачів, має великий обсяг для зберігання даних і різні механізми резервування.

Добре інтегрується з одно- і 3-екранними рішеннями HD відео-конференц-зв’язку для забезпечення запису й потокової передачі; відкриті API для простої інтеграції з IPTV і системами інших виробників.

Підтримує формати телетрансляції й потокової передачі за запитом, різні варіанти «картинка в картинці» і дружнє індексування, пошук і запит контенту з функцією демонстрації мініатюрної картинки.

Функціональні можливості:

– Гнучкий медіапроцесор запису й потокової передачі HD з відкритими API на базі SOAP для інтеграції зі сторонніми додатками потокової передачі медіаданих, підтримується FTP і TS передача й зберігання.

– Великомасштабна 30-канальний запис 1080p60 і 2 000 одночасних каналів відтворення відео за запитом; також підтримується відео у форматі 1080p30, 720p60, 720p30, 4CIF і CIF.

– Підтримує локальний і мережний запис і трансляцію; зручний доступ на базі веб, підтримує мобільне відтворення без додаткових додатків на основі HTML5.

– Три механізми резервування для забезпечення високої надійності – живлення, жорсткі диски й чипсет.

– Шифрування безпеки: HTTPS, H.235, TLS/SRTP.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Бездротова конференц-система Wi-Fi використовує стандартну технологію зв’язку Wi-Fi і сумісна зі стандартною бездротовою мережею Wi-Fi 2.4/5G, що дозволяє системі вільно перемикатися на 40 дозволених каналах. Повторна передача втрачених пакетів і функція інтелектуального керування каналами, навіть якщо зовнішні канали перекриваються, вона має достатню здатність протидіяти перешкодам, щоб уникнути впливу на передачу аудіо в системі.

Просте налаштування та інсталяція

Після простого підключення бездротової точки до центрального блоку вся система працюватиме. Кожна одиниця має індикатор потужності сигналу, ви можете розмістити одиниці відповідно до позиції кожної людини та стежити за індикатором сигналу.

Ідеальний звук

Кожен пристрій постачається з високоякісним вбудованим динаміком для чистого звучання. Коли в основному використовується зовнішній гучномовець, система забезпечує вбудований процесор DSP з автоматичним придушенням зворотного зв’язку AFC для придушення шуму зворотного зв’язку. Автоматичне регулювання гучності AGC дозволяє контролювати відстань звукосприйняття та використовувати різну довжину MIC відповідно до потреб клієнта. Функція автоматичного контролю шуму ANC зменшує шум мікрофона, щоб вловити навколишнє середовище, а також має вбудовану функцію поділу динаміків тощо, щоб усю систему можна було вільно регулювати відповідно до розміру приміщення, обладнання та акустичного середовища для досягнення ідеальної передачі звуку.

Велика ємність батареї та економія енергії

У конференц-блокі використовується знімна батарея надвисокої ємності 4900 мА. Коли мікрофон або динамік увімкнено, пристрій може безперервно працювати протягом 24 годин і працювати в режимі очікування 60 годин. OLED-дисплей має 5 індикаторів живлення зі значком низького енергоспоживання на екрані OLED, щоб нагадати користувачеві, що адміністратор також може безпосередньо замінити батарею або використовувати адаптер постійного струму для безпосереднього живлення пристрою.

Конференц-система має унікальний режим «енергозбереження», який автоматично переходить в режим очікування, якщо система неактивна протягом двох годин. Це економить енергію, а також уникає непотрібного використання ресурсів бездротового каналу. Коли користувачеві потрібно перезапустити зустріч, його можна легко розбудити через інтерфейс розмовного пристрою. Така конструкція допомагає мінімізувати експлуатаційні витрати та подовжує термін служби системи.

Робота з сучасними вимогами до зустрічей

Кожна бездротова конференц-система забезпечує вхідний аудіоінтерфейс, який забезпечує легкий доступ до аудіо з мобільних телефонів і ноутбуків у локальній конференц-системі для мобільних відеоконференцій і спільного використання аудіо з мобільного пристрою.。

Різні функції залежно від моделей

Бездротова конференц-система використовує 48K, 16-бітну нестиснену аудіопередачу, здатну передавати 8 каналів голосу та контрольних сигналів. одночасно. На основі цієї технології ми надаємо клієнтам різні рішення, включно з бездротовими дискусійними блоками. Бездротова дискусія з голосуванням, синхронний переклад, бездротова дискусія з двома одночасними каналами та інші моделі, що охоплюють різні потреби ринку.

Спільна робота дротових/бездротових пристроїв

Для багатофункціональних конференц-залів, кімнат для тренінгів, залів

для релізів тощо, які часто потрібно змінювати під час використання, блок голови, як правило, є фіксованим дротом. Інші місця для аудиторії тимчасово збільшують або зменшують конференц-блок відповідно до потреб. Наразі дротова та бездротова змішана конференц-система є гнучкою у використанні. Забезпечує велику зручність, бездротовий блок не потребує кабелю, і його можна розмістити безпосередньо за потреби.

Автоматичне відстеження HD-камери

Основний блок системи оснащено функцією відстеження камери, яка може надсилати код на камеру або безшовну матрицю для реалізації автоматичного відстеження та запису камери високої чіткості.

Чи можу я запустити кілька систем в одній будівлі або на одному поверсі?

Завдяки багатодіапазонній бездротовій системі, яка працює в діапазонах 2,4 ГГц і 5 ГГц, у вас є 25 каналів на вибір (у Європі). Таким чином, ви можете налаштувати кожну систему на роботу на різних каналах, щоб одна не заважала іншій. Наше меню швидкого вибору на кожному пристрої дозволяє навіть вільно переміщувати пристрій з однієї кімнати в іншу для використання.

Як я можу гарантувати конфіденційність зустрічі?

Само собою зрозуміло, що безпека є важливою проблемою, і до неї не слід ставитися легковажно. Але подумайте: якщо ви зробите правильний вибір, будь-хто, хто намагатиметься прослухати вашу зустріч, повинен буде подолати три основні перешкоди.

Перший – власний протокол. Якщо підслуховувач не має такого ж типу обладнання, йому доведеться спочатку з’ясувати, як працює протокол зв’язку, перш ніж він зможе почати декодування інформації.

Потім є список доступу – список ідентифікованих і довірених підрозділів, які є частиною вашої конференц-системи. Цей список доступу зберігається в системі та використовується для перевірки, чи дозволено пристрою бути частиною системи. Підрозділ, який намагається підключитися, буде відхилено,

оскільки його унікальний ідентифікатор відсутній у списку надійних пристроїв.

Нарешті, шифрування. Завдяки використанню нашої системи весь зв’язок між модулями делегатів і точкою доступу бездротової конференції буде захищено ключем шифрування. З такими заходами безпеки несанкціоноване прослуховування практично неможливо.

3.2 Розробка структурної схеми

Структурна схема системи відображена на рисунку 3.1. Це рішення працює з усіма основними типами персональних пристроїв (з ОС Windows, OSX, Apple iOS і Android), дозволяючи виводити зображення з них на загальний дисплей. Використовувати можна практично з будь-якими дисплеями, що мають порт HDMI.

Ознайомлення з системою

Бездротовий основний блок 5G Wi-Fi VIS-DCP2000-W

Характеристики

Унікальна технологія цифрової кільцевої мережі AUDIO-LINK забезпечує передачу та обробку повністю цифрового сигналу.

Один кабель CAT5e для передачі до 64 каналів аудіо та інших сигналів.

Високоякісне звучання завдяки технології передачі аудіо без втрат, частоті дискретизації аудіо 48 КБ і частотній характеристикі 20 Гц ~ 20 КГц.

З’єднання «Hand-in-Hand-Loop-Network» забезпечує бездоганну роботу системи незалежно від того, змінюєте дискусійний блок або маєте будь-які несправні блоки.

Підтримка AGC (автоматичне регулювання посилення)/AFC (адаптивне придушення зворотного зв’язку)/ANC (акустичне шумопоглинання) Підтримка виведення звуку за допомогою розділені зони, кожна зона автоматично регулюється відповідно до гучності положення мікрофона, досягайте більших відстаней без свистка.

Конфігурація конференц-залу

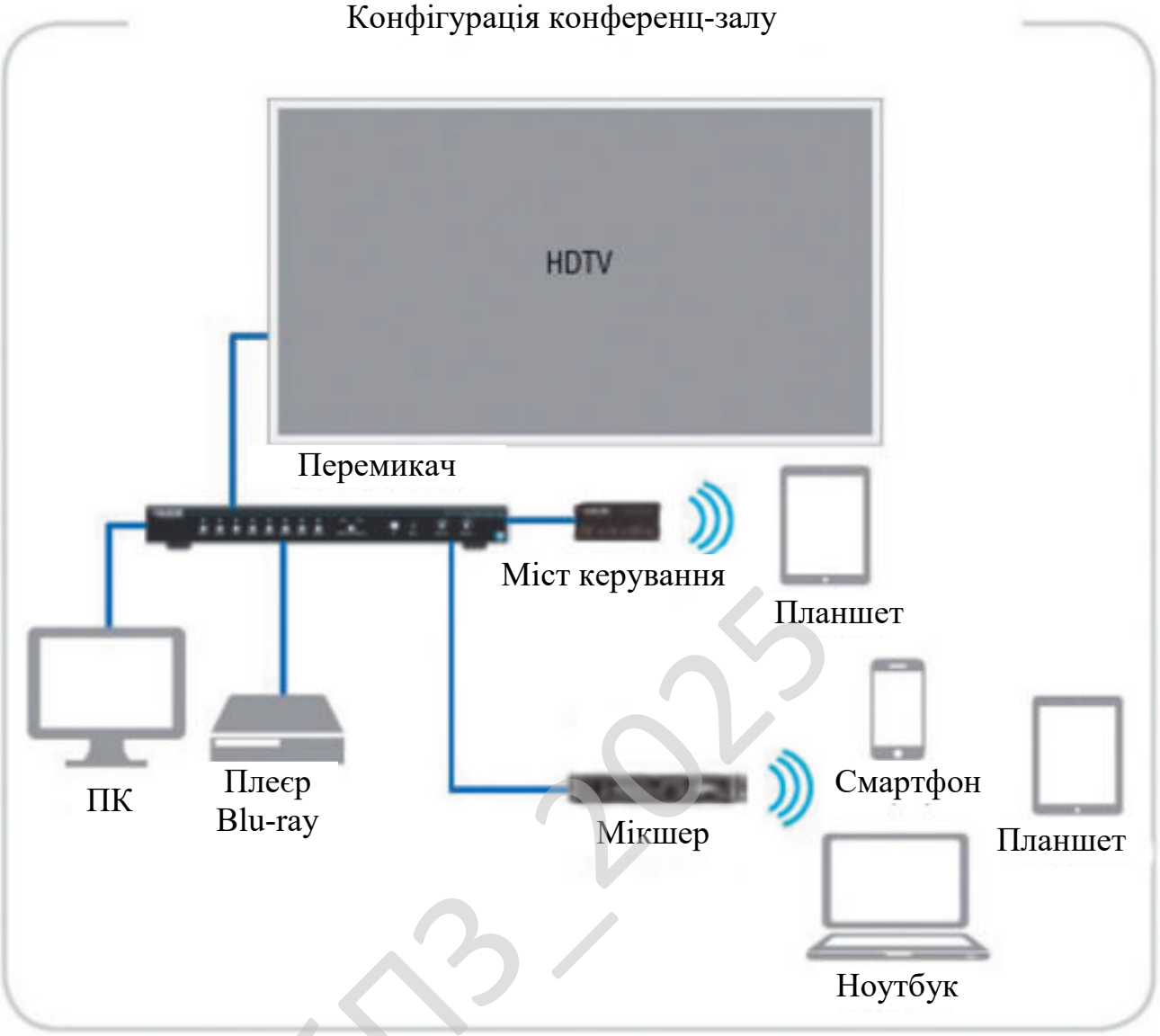

Рисунок 3.1 – Структурна схема системи

Підтримка придушення відлуння, віддаленого двостороннього виклику в режимі реального часу з іншим контролером конференції або іншим стороннім терміналом відеоконференцзв’язку.

Запис USB, розширена технологія шумопоглинання для більш чіткого запису

Підтримка програмного керування через RJ45 та веб-конфігурації

Функції

Відповідність IEC 60914, GBT 15381-94.

Завдяки цифровому блоку живлення VIS-CNB його можна розширити до 5200 дискусійних блоків.

Підтримка З'єднання «Hand-in-Hand-Loop-Network» і система, призначена для кожного блоку, окремий ідентифікатор, щоб уникнути конфлікту повторюваних ідентифікаторів.

Обидва дротові та бездротові дискусійні блоки можуть підключатися до контролера, швидко та легко додаючи делегата нарад.

Бездротовий блок підтримує до 8 активних мікрофонів одночасно, здатність запобігати перешкодам сильніша, ніж будь-коли, відповідність стандарту Wi-Fi IEEE 802.11n як для 2,4 ГГц і 5 ГГц.

2-канальний аудіовхід, тип XLR або RCA, для локального входу або входу для віддаленої відео конференції.

Максимальний 8-канальний аудіовихід, тип XLR, RCA або Phoenix, для підключення системи перекладу мови або виведення в різні зони.

TCP/IP-з'єднання між контролером і ПК.

RS232 для надсилання протоколу PELCO/VISCA для реалізації функції автоматичного відстеження камери.

Гаряча заміна для будь-якого системного блоку, а контролер має функцію автоматичного відновлення.

Широкий діапазон напруги від 110 В ~ 220 В змінного струму

Без використання програмного забезпечення для ПК наш контролер конференції все ще володіє такими налаштуваннями:

– Обмеження на кількість осіб, які говорять: шляхом встановлення кількості активних блоків одночасно (Кількість становить 1/2/4/6/ одиниць).

– Режими обговорення.

Керування та індикатори

РК-дисплей для відображення всіх операцій і результатів.

4-позиційна кнопка для роботи з меню.

Натисніть одну кнопку, щоб почати запис.

Світловий індикатор запису.

Основна ручка регулювання гучності.

Технічні параметри

Потужність 110 В / 220 В змінного струму.

Статичне енергоспоживання 12 Вт.

Максимальне споживання електроенергії 150 Вт.

Частотна характеристика 20 Гц ~ 20 КГц.

Співвідношення шумів (сигнал / шум)> 80 дБ.

Загальні гармонійні спотворення <0,05%.

Перехресні перешкоди каналу> 80 дБ.

Розміри (мм) 483 Д x 260 Вт x 43,6 В.

Темно-сірий колір.

CLEACON VIS-WDC-T/VIS-WDD-T Бездротова цифрова дискусія Голова/група делегатів

Характеристики

Стильний, низькопрофільний дизайн із сенсорним інтерфейсом.

Бездротовий зв’язок, налаштування всієї системи за кілька хвилин, щоб бути готовою до використання, а також її прибирання за короткий час.

Вбудований роз’єм постійного струму для живлення без батареї. Знімну батарею легко замінити та зарядити. Внутрішній гучномовець.

Телеконференція

Повністю цифрова передача та обробка сигналу, повністю уникайте радіочастотних перешкод від мобільного телефону чи подібних пристроїв. Підтримка мікрофона, що підключається, і мікрофона різної довжини з сильною здатністю збору звуку. Відображення часу та дати на OLED-екрані Таймер на дисплеї, щоб встановити певний час для кожного делегата для виступу, нагадує користувачеві або автоматично закриває мікрофон. Усі

підрозділи можна налаштувати як голову чи делегат за допомогою налаштувань програмного забезпечення на основі моделі WDC-T.

Функції

Відповідність IEC 60914, GBT 15381-94.

Стандартна технологія Wi-Fi сумісна з іншими мережами Wi-Fi, підтримує 2,4 ГГц і 5 ГГц на всіх каналах.

Повністю цифрова технологія передачі та обробки нестисненого аудіо, професійний мікрофон для досягнення ідеальної якості звуку на 20 Гц ~ 20 КГц; 128-бітна технологія цифрового шифрування з безпечним з'єднанням WAP2, фільтрацією MAC-адрес, приховуванням SSID для запобігання прослуховування та несанкціонованого доступу з пульта делегатів.

8 блоків делегатів можуть відкриватися одночасно.

Підтримка більшої кількості точок доступу, що працюють разом, щоб збільшити діапазон покриття Wi-Fi для бездротових блоків MIC, які працюють у великій конференц-залі або кількох кімнатах.

З нашою камерою HD PTZ і відеоматричним комутатором за допомогою RS485 або RS232 протокол для надсилання протоколу PELCO/VISCA для реалізації функції автоматичного відстеження камери.

Головка мікрофона з точним напрямком із пінопластовою кришкою для захисту від вітру, із двоколірним світлодіодним індикатором.

Кнопка для ввімкнення/вимкнення мікрофона, пріоритету та застосування. Кнопка відповіді для блоку голови.

Вбудовані динаміки високої точності, він автоматично вимикається, щоб запобігти завиванню, коли натискаєте MIC ON.

Інтерфейс гарнітури з обох сторін із регулюванням гучності.

Підтримка технології AGC (автоматичне регулювання посилення)/AFC (адаптивне придушення зворотного зв'язку)/ANC (активне керування шумом)/міксування (автоматичне змішування).

Вбудовані функції для внутрішнього зв'язку.

Без використання програмного забезпечення для ПК наш контролер конференції все ще володіє такими налаштуваннями:

– Обмеження на кількість осіб, які говорять: шляхом встановлення кількості активних пристроїв одночасно (кількість становить 1/2/4/6/ одиниць).

– Обговорення режими: «OPEN» Безкоштовно для всіх, обмежено налаштуваннями активних пристроїв контролером, «OVERRIDE» Першим прийшов, першим вийшов, «БЕЗКОШТОВНО» Увімкніть мікрофон вільно до максимуму. кількість, "ЗАСТОСУВАТИ" Щоб застосувати, а потім говорити

– Блок, що показує реальний час на екрані, і таймер зворотного відліку для розмовляючої особи.

– Знімну батарею легко замінити та зарядити, технологія енергозбереження забезпечує 20 годин безперервного використання та 48 годин у режимі очікування.

Елемент керування та індикатори

Кожен блок із двоколірним світлодіодним індикатором на мікрофоні, червоний для розмови, зелений означає, що потрібен дозвіл від голови, щоб говорити, OLED-дисплей 128x32 з панеллю гучності, годинником, інтервалом часу активації мікрофона, МІКРОФОНОМ УВІМКНЕННЯ/ВИМКНЕННЯ тощо та іншою інформацією.

Індикатор регулювання гучності.

Індикатор увімкнення/вимкнення мікрофона.

Головний блок із кнопкою пріоритету та кнопкою згоди для делегатів Кнопка увімкнення/вимкнення живлення ховається з лівого боку.

Інтерфейси

Під’ємна основа мікрофона.

2x 3,5 мм стереороз’єм для навушників.

1x 3,5 мм стереовхідний роз’єм.

Контактор для акумуляторної батареї.

Роз’єм постійного струму для адаптера живлення.

Технічні параметри

Кнопка: сенсорний інтерфейс, відсутність фізичних кнопок.

Звук Звукознімач: ємність типу серця.

Дисплей: OLED-дисплей 128 × 32.

Чутливість: -46 дБВ / Па.

Максимальне споживання електроенергії: 2,0 Вт.

Спрямованість: 0 ° / 180 °> 20 дБ (1 кГц).

Навантаження на навушники: 16Ω.

Гучність навушників: 10 мВт.

Гніздо для навушників: 3,5 мм стерео.

Вхідний опір: 2kΩ.

SNR: 70 дБ.

Частотна характеристика: 20 ~ 20000 Гц.

Еквівалент шуму: 20 дБА (SPL).

Основний матеріал: ABS.

Робоча температура: від 0 ℃ до + 55 ℃.

Колір: чорний.

Максимальний звуковий тиск: 125 дБ (THD <3%).

Вага: 1,1 кг (з мікрофоном).

Розміри: 185 × 130 × 50 мм (ширина × глибина × висота) (без мікрофона).

CLEACON VIS-AP4C 2,4 ГГц/5 ГГц точка доступу до конференції

Особливості

Підключення за допомогою конференц-процесора DCP2000 або за допомогою комутатора, щоб розширити більше конференц-точок доступу в одній системі.

Типовий діапазон охоплення 30 м.

Встановлення на стелі, стіні або на стійці.

128-бітна технологія цифрового шифрування, із захищеним з'єднанням WAP2, фільтрацією MAC-адрес, приховуванням SSID для запобігання прослуховування та несанкціонованого доступу з пульта делегатів.

Функції

Світлодіодний індикатор сигналу/живлення.

2.4G/5GHz дводіапазонний зв'язок.

Підтримка локальної мережі. Джерело живлення POE.

RJ45 для підключення до локальної мережі.

Технічні параметри

Споживання в робочому стані: 11 Вт.

Підтримується живлення через Ethernet (POE).

Встановлення POE: на стелі, на стіні або в стійці.

Розміри: 17*17*2,8 см (ШхГхВ).

Вага: 316 г.

Колір: білий.

Модель товару для замовлення

VIS-AP4C 2,4 ГГц/5 ГГц Точка доступу для конференцій (біла).

CLEACON VIS-WCH1 Зарядний пристрій для акумулятора

Характеристики

Чотири години, необхідні для повного заряджання.

МАКСИМУМ 8 акумуляторів заряджаються за один раз.

Широкий діапазон вхідної потужності: 100 В ~ 240 В змінного струму.

Корпус пряжки для запобігання поганому контакту або падінню, легка вага для легкого переміщення.

Функції

Контактор за допомогою висувних штифтів.

Корпус пряжки для запобігання поганому контакту або падінню.

Сертифікати

Сертифікація CE.

Технічні параметри

Вхід: 100 ~ 240 В змінного струму 47 ~ 63 Гц.

Вихід: 8,4 В/1000 мА.

Споживана потужність: 20 Вт.

Розмір: 190*86*127 мм (ДхШхВ).

Вага: 1100 г.

3.3 Розробка функціональної схеми

Робочі місця учасників процесу відеоконференцій

Під «робочим місцем» розуміється комплект основного, додаткового й супутнього технологічного встаткування, а також інших засобів, розміщених на спеціально виділених площах і необхідних учасникові відеоконференцій для виконання всіх передбачених технологією функцій і процесів. Робочі місця для різних учасників процесу відрізняються друг від друга по составі встаткування і їхньому оформленню залежно від характеру технологічних процесів, виконуваних кожним з учасників. Робоче місце оснащується технологічним устаткуванням і технологічними меблями, засобами висвітлення, електропостачання й життєзабезпечення (опалення, вентиляції й кондиціювання повітря).

Загальні вимоги містять у собі також вимоги до технологічності, ергономіці, електропостачанню, висвітленню, шуму, безпеці, гігієні праці й профілактиці профзахворювань.

Робоче місце оператора сервера відеоконференцій

В состав устаткування оператора сервера ВК входить система резервування й керування відеоконференціями, професійний персональний комп'ютер з модемом і точка підключення до Інтернету.

Робочі місця користувачів системи відеоконференцій

Види робочих місць. Залежно від видів конференції, учасників і категорії термінального встаткування можуть бути виділені чотири основних види робітників місць.

Індивідуальне робоче місце абонента (користувача) ВК збігається з його постійним робочим місцем в офісі. Групове робоче місце перебуває або в безпосередній близькості від постійного робочого місця одного з учасників ВК, або розміщається в окремо виділеному або пристосованому для цього приміщенні.

Робоче місце оператора ВК територіально розташовується або поблизу групового робочого місця абонента ВК, або на ділянці архіву ВК, залежно від виконуваної роботи. До складу встаткування робочого місця оператора ВК входить комплект устаткування монтажу, кодування відеофонограм і виготовлення архівних копій, а також термінал керування архівом ВК.

Робоче місце оператора сервера ВК розташовується на ділянці зв'язку поблизу сервера ВК і має термінал системи керування відеоконференціями.

Состав устаткування й структурні схеми. Робоче місце користувача системи відеоконференцій включає абонентський термінал ВК індивідуального або групового застосування з відеокамерою, мікрофоном, відеотерміналом, гучномовцями й електронними блоками; додаткове встаткування (документальну відеокамеру, комп'ютер, додаткові відеокамеру, мікрофон і відеомагнітофон) і супутнє встаткування (відеопроектор, системи спецосвітлення й звукопідсилення). Залежно від призначення й цілей застосування робочого місця користувача додаткове й супутнє встаткування використовують у різній комплектації.

Вимоги до розміщення встаткування й людей

Розміщення встаткування й людей – користувачів системи ВК – повинне відповідати ряду суперечливих вимог, частина з яких нормована. По-перше, повинні бути виконані норми МСЕ-Р (сектора Р Міжнародного союзу

електрозв'язку) і ЄСВ (Європейського союзу віщання), пропоновані до умов перегляду відеоматеріалу й прослуховуванню звукового матеріалу, по-друге, – технічні вимоги розміщення кожного з видів устаткування, по-третє, враховані норми СНіП у частині електро- і пожаропожежебезпеки. Параметри приміщення й умови розміщення встаткування й людей підлягають розрахунку з урахуванням наступних норм і документів:

– співвідношення сторін приміщення – відповідно до «благозвучного» співвідношеннями Болта [1, 2];

– обсяг приміщення й час реверберації – відповідно до норм МСЕ-Р (МККР) серії BS [3, 4] і вимогами ЄСВ [5];

– мінімальна відстань гучномовців термінала ВК від задньої стіни й розташування людей щодо гучномовців – відповідно до вимог МЕК [6] і МСЕ-Р [3, 7];

– шумові характеристики приміщення – відповідно до норм МСЕ-Р і ЄСВ [2, 3, 7];

– розташування людей щодо екрана відеомонітора – відповідно до вимог рекомендацій МСЕ-Р серії ВТ [8 -10];

– розміщення людей щодо стін приміщення – відповідно до вимог СНіП [11-13], а також з урахуванням вимог, пропонованих до висвітлення тла при відеозйомці.

Вимоги до розмірів приміщення визначаються з урахуванням акустичних вимог до часу реверберації, частотній характеристиці й раннім відбиттям [5, 7].

Оптимальне планування робочого місця досягається на підставі інженерного розрахунку.

В випадку розміщення учасників відеоконференції в залі використовують додаткове й супутнє встаткування. Основну відеокамеру, установлену звичайно на відеотерміналі, направляють убік доповідача на трибуні, а додаткову – у зал. Відеотерминал розташовують поблизу доповідача й оператора ВК, щоб вони могли стежити за зображенням. Зображення для учасників у залі проектується на

екран. Для них у залі встановлюють додаткові мікрофони й передбачають систему звукопідсилення. Для забезпечення правильної передачі кольору в залах використовують систему спецосвітлення.

Особлива увага варто приділяти акустиці приміщень. Акустична обробка поверхонь стін, стелі й підлоги повинна бути такий, щоб запобігти раннім відбиттям звуку, видаваного учасниками й вихідного від гучномовців. Авторові доводилося зіштовхуватися з випадками, коли через зневагу акустичними вимогами помилка в наведенні відеокамери на робочому місці користувача ВК перевищувала 200. При цьому камера наводилася не на промовця, а на його сусіда.

Для проведення групових відеоконференцій важливо ретельно проробити світлотехнічні рішення. Так, колірна температура штучних джерел світла в залі повинна бути однакової щоб уникнути порушення передачі кольору.

Багатоточечний сервер відеоконференцій (ВК) разом із системою керування відеоконференціями є гнучким модульним багатофункціональним засобом для підтримки багатоточечних ВК. Функціональна схема системи наведена на рисунку 3.2.

Крім модулів каналів зв'язку, сервер ВК містить модулі звукового процесора й відеопроцесора, синхронізатора зображення й звуку, транскодера H.261/H.263, а також набір інтерфейсних модулів, зв'язаних системною шиною. Керуючі модулі керують роботою всієї системи. На «транковом» рівні сервер ВК звичайно підтримує інтерфейси IDNX-PRI (Integrated Services Digital Network Primary Rate Interface) і T1 (E1) BBS (Robbed-Bit Signaling), на лінійному рівні – інтерфейс ISDN-BRI (Integrated Services Digital Network Basic Rate Interface), протокол цифрових з'єднань DCP (Digital Communications Protocol) і аналогові лінійні канали для з'єднання через модеми. Звичайно сервер ВК заснований на стандартах H.320 і T.120 MCE-T (Сектора Т Міжнародного союзу електрозв'язку) і має широкі можливості транскодування, що забезпечує сумісність його з найрізноманітнішими засобами відеоконференцій. В таблиці 3.1

для приклада наведені деякі характеристики одного із серверів ВК (MCU Lucent Technologies), що визначають його сумісність із іншим устаткуванням.

Сервер ВК підтримує відеостандарти, звукові стандарти й стандарти даних, перераховані в таблицях 3.2-3.5.

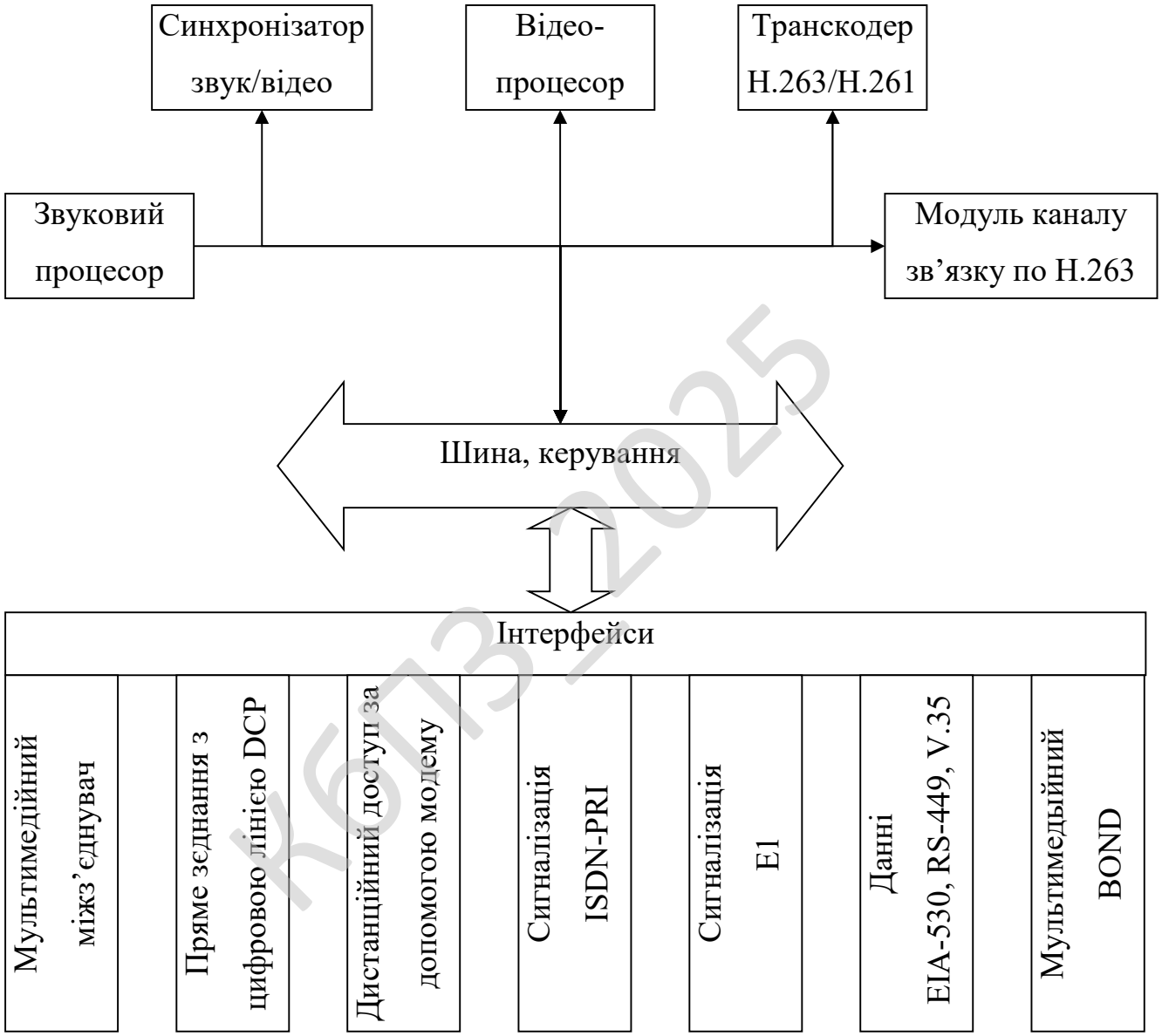

Рисунок 3.2 – Функціональна схема сервера відеоконференцій

Термінали відеоконференцій

У системі відеоконференцій можуть бути використані моделі групових напольних, компактних і настільних відеотерміналів різних виробників. Властивості систем різних фірм у кожній із груп досить близькі.

Ці засоби дозволяють працювати по протоколах H.320 і H.323 і поряд із груповими й компактними терміналами ВК можуть бути включені в корпоративну систему ВК.

Таблиця 3.2 – Можливості транскодування

Категорія	Стандарти й параметри інтерфейсів
Транскодування сигналів зображення	G.711 і G.722
	G.711 і G.728
	G.711 і G.723 (через Gateway)
	Частота кадрів – від 7,5 до 30 кадр/с Розрішення – CIF/QCIF Стиск – H.261/H.263
Доступ до мережі	Цифрова швидкість – від 56 до 768 кбіт/с Режими – BONDing, багатошвидкісний, багатоканальний
Багатоточечні протоколи	H.320 і H.323
Конференція даних (T.120)	Допускається в комбінованих конференціях Змішання H.320/H.323 (через Gateway)

Комунікаційні властивості встаткування системи ВК

Основою технологічного встаткування відеоконференцзв'язку служать термінали ВК різних конфігурацій і сервер ВК із системою керування відеоконференціями. Сервер ВК є головним телекомунікаційним засобом багатоточечної системи ВК і має продуктивні цифрові інтерфейси для високошвидкісної передачі голосових повідомлень, зображень і даних. Він містить інтерфейси цифрових мереж інтегрованих служб (ISDN): для базової цифрової швидкості (BRI) і первинної цифрової швидкості (PRI). Крім того, він

підтримує протокол прямих цифрових з'єднань (DCP). При з'єднанні ISDN-PRI на рівні первинної групи группостворення E1 застосовується формат 30B + D, а при ISDN-BRI, – формат 2B + D, де B (64 кбіт/с) – потік для основної інформації служби, а D (16 кбіт/с) – потік керування й сигналізації для приєднаних каналів B. Протокол цифрових з'єднань DCP застосовується для зв'язку терміналів ВК із системою керування за допомогою високошвидкісних і мультимедійних з'єднань (MML). Сервер ВК може підключатися до кодеків H.320, H.CTX для передачі даних через інтерфейси EIA-530, RS449 або V.35 із застосуванням RS366 для сигналізації (дозвона). У випадку надання смуги на вимогу (BONDing) можна здійснювати проведення відеоконференцій без використання каналів ISDN-PRI і широкополосних каналів типу H0.

Таблиця 3.3 – Відеостандарти, підтримувані сервером ВК

Відеостандарт MCE-T	Найменування, зміст
H.221	Стандарт структури кадрів для відеоконференцзв’язку
H.230	Стандарт кадрової синхронізації для відеоконференцзв’язку
H.231	Стандарт відеоконференцзв’язку, що визначає з'єднання між звуковізуальними терміналами
H.242	Стандарт, що визначає системи для подання з'єднань між звуковізуальними терміналами
H.243	Рекомендації ITU-T: процедура встановлення зв'язку між трьома або більше аудіовізуальними терміналами, що використовують канали зі швидкістю передачі цифрової інформації до 2 Мб/с.
H.261	Стандарт відеокодека для звуковізуальних служб зі швидкістю H.320
H.263	Кодування й декодування зображення для передачі з низькою швидкістю з поліпшеними характеристиками і якістю по каналах H.261

Таблиця 3.4 – Звукові стандарти, підтримувані сервером ВК

Звуковий стандарт МСЕ-Т	Параметри якості	Смуга, бітна швидкість
G.711	3,5 кГц	48/56/64 кбіт/с
G.722	7 кГц	48 кбіт/с на швидкості N*56 кбіт/с
		56 кбіт/с на швидкості N*64 кбіт/с
G.728	3.5 кГц	16 кбіт/с

Таблиця 3.5 – Стандарти даних, підтримувані сервером ВК

Стандарти МСЕ-Т	Найменування, зміст
Т. 122	Багатоточечна служба зв'язку для звукографії й відеоконференцзв’язку
Т.123	Стеки протоколів ISDNProfile-MLP для застосувань звукографії й відеоконференцзв’язку
Т.124	Специфікація GCC (Genetic Conference Protocol)
Т. 125	Протокол MC5
Т.126	Протокол для нерухливих зображень і описів при багатоточечному зв'язку
Т.127	Протокол перетворень багатоточечних бінарних файлів

Устаткування системи ВК

Можливі схеми з'єднання терміналів ВК Термінали відеоконференцій з'єднуються один з одним за допомогою каналів зв'язку й коммунікаторів. Для багатоточечної системи відеоконференцій головним коммунікатором є багатоточечний сервер ВК. Залежно від взаємного розташування терміналів і сервера ВК, режимів роботи пристроїв, а також використовуваних мереж зв'язки можливі різноманітні схеми з'єднання терміналів і серверів ВК. При значній взаємній далекості терміналів ВК зв'язок між ними й сервером, а також між ними

самими (в окремих випадках) здійснюється з використанням каналів магістральних, міжрегіональних і регіональних мереж зв'язку.

У системі відеоконференцій багатоточечний відеоконференцзв’язок може вироблятися по протоколах H.320 і H.323 MCE-T. Зв'язок терміналів ВК, що працюють по протоколі H.323, із сервером ВК може здійснюватися через вузли (Gateway). З'єднання вузла Gateway із сервером ВК виробляється по протоколах BRI і V.35/RS336.

Підключення сервера ВК до магістральних, міжрегіональних і регіональних каналів зв'язку може вироблятися через маршрутизатор, магістральний мультиплексор і цифрову АТМ. З'єднання сервера із цими пристроями може здійснюватися з використанням протоколів PRI/E1, G.703. Підключення терміналів ВК до магістральних, міжрегіональних і регіональних каналів зв'язку може відбуватися через мультиплексори земних станцій (HUB) і абонентських станцій (VSAT) супутникового зв'язку, магістральні мультиплексори (MUX) і АТМ. Залежно від відстані між терміналами й цим устаткуванням з'єднання можуть здійснюватися або безпосередньо з використанням протоколу V.35, або із застосуванням модемів або інверсних мультиплексорів.

Деякі особливості роботи підключення терміналів ВК до каналоутворюючому встаткування

Безпосереднє підключення термінала ВК до каналоутворюючому встаткування допускається в тих випадках, коли довжина сполучних кабелів невелика. У випадках, коли термінали й каналоутворюючое встаткування вилучені друг від друга, для їхнього з'єднання повинні використовуватися модеми або інверсні мультиплексори.

Для входу в супутникову мережу термінали ВК повинні бути підключені до мультиплексору центральної, вузлової або абонентської земної станції (ЗС) з використанням інтерфейсної плати внутрішнього модуля ЗС. Залежно від відстані між терміналом ВК і внутрішнім модулем земної станції підключення

виробляється прямо або із застосуванням мультиплексора (модему). З'єднання рекомендується здійснювати з використанням протоколу V.35. Подібні способи підключення мають на увазі застосування твердої структури з'єднань, слабко пристосованої до подальшого розвитку. Вони доречні для використання в кінцеві (не вузлових) пунктах мережі відеоконференцзв’язку.

У тих випадках, коли об'єкт відеоконференцзв’язку розглядається як вузловий об'єкт, підключення термінала ВК до внутрішнього модуля ЗС супутникового зв'язку варто здійснювати через мультиплексор.

Термінал ВК підключається до інтерфейсному модуля мультиплексора прямо або через модем або інверсний мультиплексор, залежно від відстані між терміналом і мультиплексором.

На вузлових об'єктах з розвитою телефонною мережею доцільно підключати термінали до магістральних мереж через цифрову АТМ. При використанні цифровий АТМ у якості коммунікатора потоків даних відеоконференцзв’язку варто мати на увазі, що ці дані не повинні піддаватися стиску в АТМ, передбаченому для звичайних телефонних сигналів. Інакше кажучи, на період сеансу відеоконференцзв'язку повинна бути як би виділена смуга на вимогу для передачі цифрового потоку з необхідною бітною швидкістю (у загальному випадку 256 кбіт/с) для кожного з напрямків відеоконференцзв’язку.

На центральній ЗС або в центральному вузлі інформатизації з розгалуженими комунікаціями підключення термінала ВК може бути здійснене через магістральний мультиплексор, цифрову телефонну станцію або маршрутизатор.

Підключення терміналів ВК до магістральних, міжрегіональних і регіональних каналів зв'язку через мультиплексори, цифрові АТМ або маршрутизатори забезпечує ряд переваг у порівнянні із прямим підключенням до ЗС. По-перше, добре розвинені комунікаційні функції цих пристроїв дозволяють їх використовувати як вузли відеоконференцзв’язку (хоча з деякими

обмеженнями). По-друге, завдяки широкому діапазону інтерфейсів, властивим цим пристроям, досягається можливість зв'язку терміналів ВК, що діють у різнорідних мережах (наприклад, супутникових і наземних, телефонних і мультимедійних і т.п.).

При цьому забезпечується гнучкість структури системи відеоконференцзв’язку й подальше нарощування користувальницьких послуг, в основному програмними, а не апаратними засобами.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

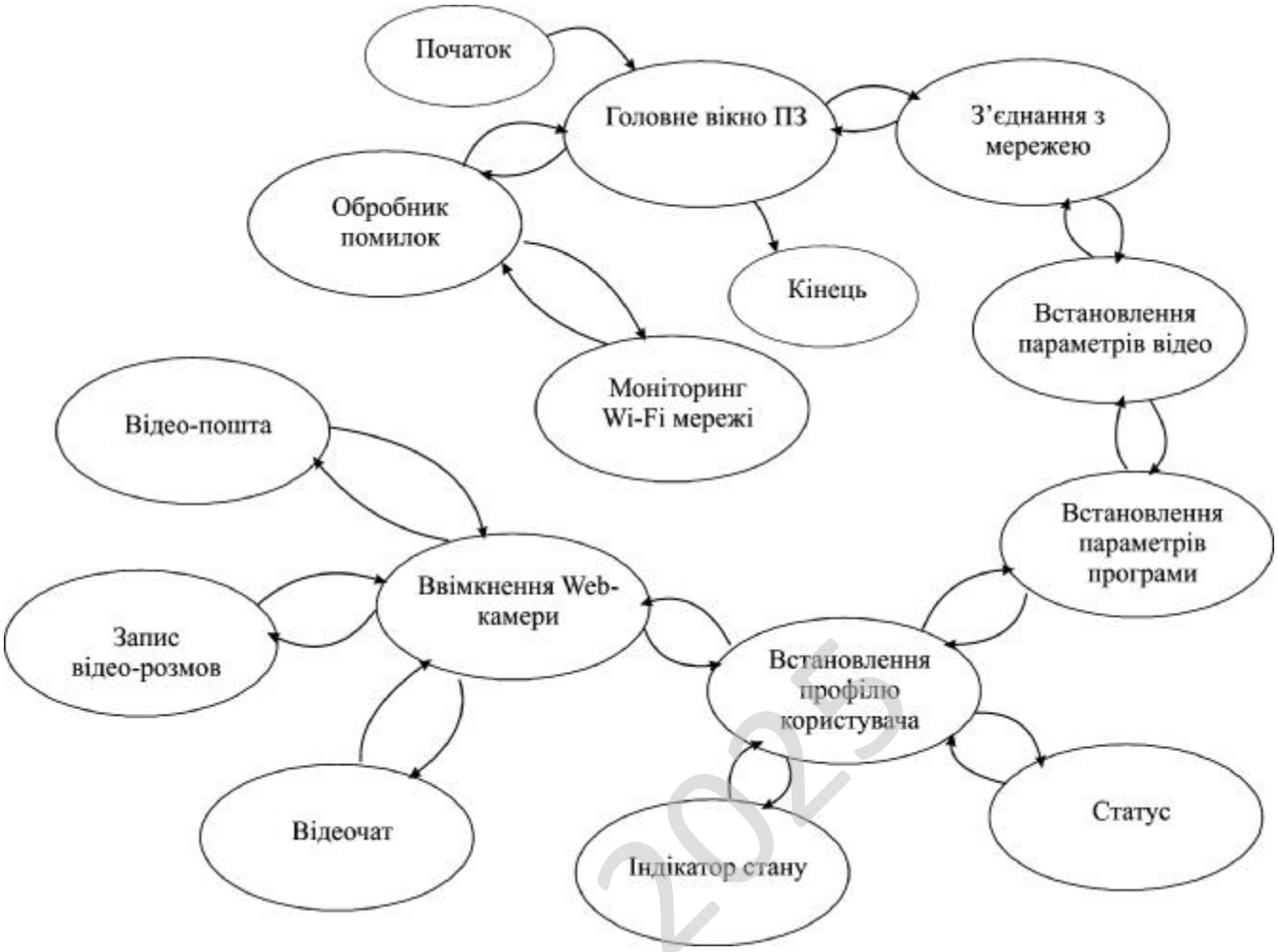

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Опис системи

Система відеотехнологій для цифрової трансформації з підтримкою Wi-Fi розробляється для автоматизації процесу відеоспостереження, аналізу відеопотоків та інтеграції з інтелектуальними алгоритмами обробки даних.

Вона забезпечує стабільну передачу відео високої якості через бездротові мережі, дозволяючи контролювати події в режимі реального часу та здійснювати аналітику отриманих даних.

Опис архітектури системи

Система складається з основних компонентів:

– Камери відеоспостереження передають відеопотік до серверної частини.

– Серверу обробки даних здійснює прийом, збереження та аналіз відео.

– Модулю машинного навчання використовує алгоритми розпізнавання об'єктів та поведінковий аналіз.

– Клієнтського додатку що забезпечує доступ до відео через веб-інтерфейс.

– Мережевий модуль Wi-Fi гарантує бездротовий зв'язок між камерами та сервером.

Обробка відеопотоків відбувається на серверній частині з використанням модулів OpenCV та TensorFlow. Система інтегрує розпізнавання облич, детекцію руху та аналіз підозрілих ситуацій.

Основні функції системи

– Запис та збереження відео у локальному або хмарному сховищі.

– Обробка відеопотоку в реальному часі з використанням нейромереж.

– Доступ до відеоархіву через веб-інтерфейс.

– Інтелектуальне сповіщення при виявленні аномальних подій.

– Адаптивна передача відео з урахуванням швидкості мережевого з'єднання.

Основні модулі програми

Програмна частина написана на Python, основні бібліотеки:

– OpenCV для роботи з відео.

– TensorFlow для глибокого навчання.

– Flask для реалізації веб-інтерфейсу.

– SQLite для збереження логів та налаштувань.

Розрахунок пропускної здатності Wi-Fi

Припустимо, що камера передає відеопотік з роздільною здатністю 1920x1080 пікселів при 30 кадрах за секунду з кодуванням H.264. При середньому бітрейті 4 Мбіт/с на одну камеру, необхідна пропускна здатність для 10 камер становить:

$$4 \times 10 = 40 \text{ Мбіт/с}$$

З урахуванням накладних витрат та бездротових перешкод, рекомендується використовувати Wi-Fi стандарту 802.11ac, що забезпечує швидкість до 867 Мбіт/с, чого достатньо для стабільної роботи системи. Система оптимізує передачу відео з використанням адаптивного бітрейту, що дозволяє зменшити затримки при зміні мережевих умов.

Фрагмент вихідного коду:

```
import cv2
import numpy as np
import tensorflow as tf
import flask
import threading
import sqlite3
from flask import Flask, Response, render_template

# Ініціалізація Flask-додатка
app = Flask(__name__)

# Завантаження моделі для розпізнавання об'єктів
class VideoProcessing:
    def __init__(self, model_path):
        self.model = tf.keras.models.load_model(model_path)
```

```
    def process_frame(self, frame):
        resized_frame = cv2.resize(frame, (224, 224))
        input_data = np.expand_dims(resized_frame, axis=0) / 255.0
        predictions = self.model.predict(input_data)
        return predictions

# Ініціалізація відеопотоку
class VideoStream:
    def __init__(self, source=0):
        self.stream = cv2.VideoCapture(source)
        self.lock = threading.Lock()
        self.frame = None
        self.running = True
        self.thread = threading.Thread(target=self.update, args=())
        self.thread.daemon = True
        self.thread.start()

    def update(self):
        while self.running:
            success, frame = self.stream.read()
            if success:
                with self.lock:
                    self.frame = frame

    def get_frame(self):
        with self.lock:
            return self.frame

    def stop(self):
        self.running = False
        self.thread.join()
        self.stream.release()

# База даних для збереження подій
class Database:
    def __init__(self, db_path="events.db"):
        self.conn = sqlite3.connect(db_path, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.create_table()

    def create_table(self):
        self.cursor.execute("""CREATE TABLE IF NOT EXISTS events (
```

```
                                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                                    timestamp TEXT,
                                    description TEXT)""")
                self.conn.commit()

            def add_event(self, timestamp, description):
                self.cursor.execute("INSERT INTO events (timestamp, description)
VALUES (?, ?)", (timestamp, description))
                self.conn.commit()

        # Обробка відеопотоку та детекція
        video_stream = VideoStream(0)
        video_processor = VideoProcessing("model.h5")
        db = Database()

        def generate_frames():
            while True:
                frame = video_stream.get_frame()
                if frame is None:
                    continue

                # Виконання обробки кадру
                predictions = video_processor.process_frame(frame)
                label = "Object detected" if np.max(predictions) > 0.5 else "No
object"

                # Додавання тексту до кадру
                cv2.putText(frame, label, (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 2)

                # Перетворення кадру у формат JPEG
                ret, buffer = cv2.imencode('.jpg', frame)
                frame_bytes = buffer.tobytes()
                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

        # Маршрутизація Flask
        @app.route('/')
        def index():
            return render_template('index.html')

        @app.route('/video_feed')
```

```
        def video_feed():
            return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')

        # Запуск сервера Flask
        if __name__ == '__main__':
            try:
                app.run(host='0.0.0.0', port=5000, threaded=True)
            finally:
                video_stream.stop()
```

Опис коду:

– Клас VideoProcessing виконує завантаження та використання моделі машинного навчання для аналізу кадрів.

– Клас VideoStream відповідає за отримання відеопотоку з камери у фоновому потоці.

– Клас Database створює базу даних для збереження подій детекції об'єктів.

– Функція generate_frames отримує кадри з камери, обробляє їх та передає у веб-інтерфейс.

– Flask-додаток надає веб-інтерфейс та маршрутизацію для перегляду відеопотоку.

Ця реалізація дозволяє працювати з відеопотоком у реальному часі, інтегрувати нейромережеві моделі та зберігати події у базі даних.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

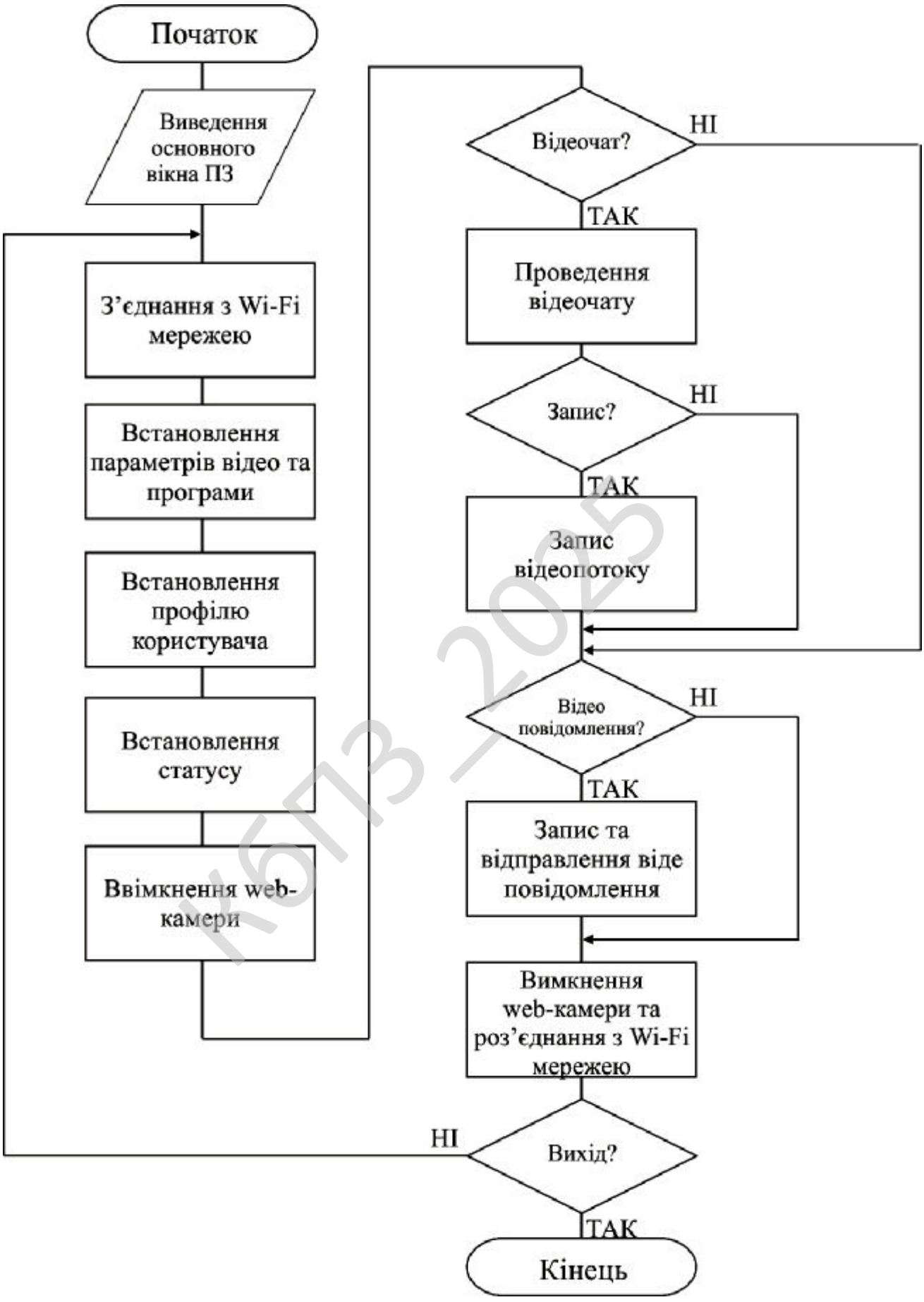

Рисунок 4.1 – Блок-схема основної програми

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

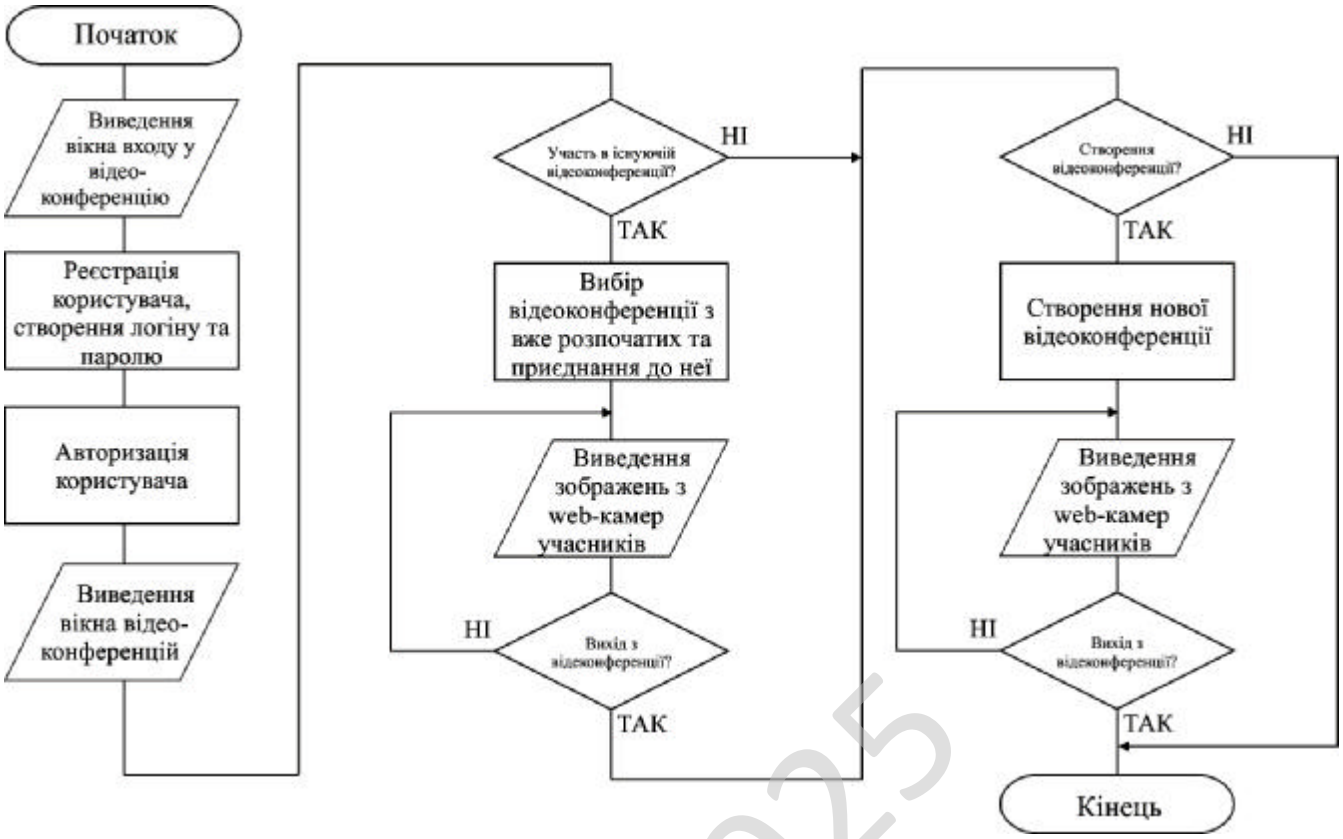

Рисунок 4.2 – Блок-схема роботи підпрограми

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об'єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших

комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а

також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 8845:2019 – алгоритм симетричного потокового перетворення. В основі ДСТУ 8845:2019 лежить класична схема підсумовуючого генератора подібна генератору SNOW 2.0, SNOW 3.0 та SNOW V. В основі ДСТУ 8845:2019 збережені всі базові операції шифрів сімейства SNOW, а раунд шифрування AES замінений на функцію нелінійної підстановки T, що реалізовує перестановку елементів скінченного поля $GF(2^{64})$ за допомогою компонентів національного стандарту симетричного криптоперетворення ДСТУ 7624:2014.

ДСТУ 8845:2019 використовує 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K і забезпечує високий та надвисокий рівень стійкості із врахуванням можливого застосування квантового криптографічного аналізу. При розробці алгоритму ДСТУ 8845:2019 орієнтувалися на сучасні 64-бітні обчислювальні системи, тому розмір слова обрано рівним 8 байт. запису байтів застосовують подання від старшого до молодшого. Генератор ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру схематично приведено у стандарті.

Як випливає з генератора ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру основними компонентами генератору є регістр зсуву з лінійним зворотнім зв'язком та скінчений автомат на базі якого виконується нелінійне перетворення T. Вхідні дані (ключ шифрування K та вектор ініціалізації IV) використовуються для ініціалізації змінної стану $S_i(i \geq 0)$, яка складається із двох компонент до складу яких входять[12]:

– 16 змінних $s^{(i)}$ – комірок регістра зсуву з лінійним зворотнім зв'язком: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, s_{13}^{(i)}, s_{12}^{(i)}, s_{11}^{(i)}, s_{10}^{(i)}, s_9^{(i)}, s_8^{(i)}, s_7^{(i)}, s_6^{(i)}, s_5^{(i)}, s_4^{(i)}, s_3^{(i)}, s_2^{(i)}, s_1^{(i)}, s_0^{(i)})$;

– Два регістри скінченного автомату $r^{(i)}$: $r^{(i)} = (r_2^{(i)}, r_1^{(i)})$. На виході отримуємо ключовий потік (гамма), який формується з 8-байтних слів Z_i.

З рисунку слідує, що відводи регістра зсуву з лінійним оберненим зв'язком побудовані за примітивним над полем GF(2^{64}) поліномом: $f(x) = x^{16} + x^{13} + \alpha^{-1}x^{11} + \alpha$, де α є коренем примітивного над полем GF(2^8) поліному $gz(z) = z^8 + \beta^{170}z^7 + \beta^{166}z^6 + \beta^2 z^5 + \beta^{224} z^4 + \beta^{70}z^3 + \beta^2$. Поле GF($2^8$) як і в ДСТУ 7624:2014 побудовано за примітивним на полем GF(2) поліномом $p(y) = x^8 + x^4 + x^3 + x^2 + 1$, а коефіцієнти $g(z)$ подаються через ступінь примітивного елементу β поля GF(2^8), тобто β корінь поліному $p(y)$. Тобто, у нас є вежа полів: $GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024})$, де:

– поле $GF(2^{1024})$ задається відводами зворотного зв'язку як фактор кільце $GF(2^{64})[x]/(f(x))$;

– поле $GF(2^{16424})$ задається як фактор кільце $GF(2^8)[z]/(g(z))$;

– поле $GF(2^{1024})$ задається як фактор кільце $GF(2)[y]/(p(y))$.

З вищезазначеного, слідує що період вихідної послідовності становить 2^{1024}.

Структурно в алгоритмі симетричного потокового перетворення ДСТУ 8845:2019 виділяють три основні функції:

– функція ініціалізації, яка приймає в якості вхідних даних 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K, і виробляє початкове значення змінної стану $S_0 = (s^{(0)}, r^{(0)})$;

– функція наступного стану Next, яка приймає на вхід зміну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє наступне значення змінної стану $S_{i+1} = (s^{(i+1)}, r^{(i+1)})$;

– функція ключового потоку Strm, що приймає на вході змінну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє на виході 64-бітний ключовий потік Z^i.

Також функція Next може виконуватися в двох режимах, в залежності від способу виконання ітерації, як частина реалізації ініціалізації алгоритму ДСТУ 8845:2019 або як частина функції ключового потоку Strm.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ відеотехнологій для цифрової трансформації з підтримкою Wi-Fi яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Налаштування; Шаблони трансформації; Довідка.

– Розділу виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

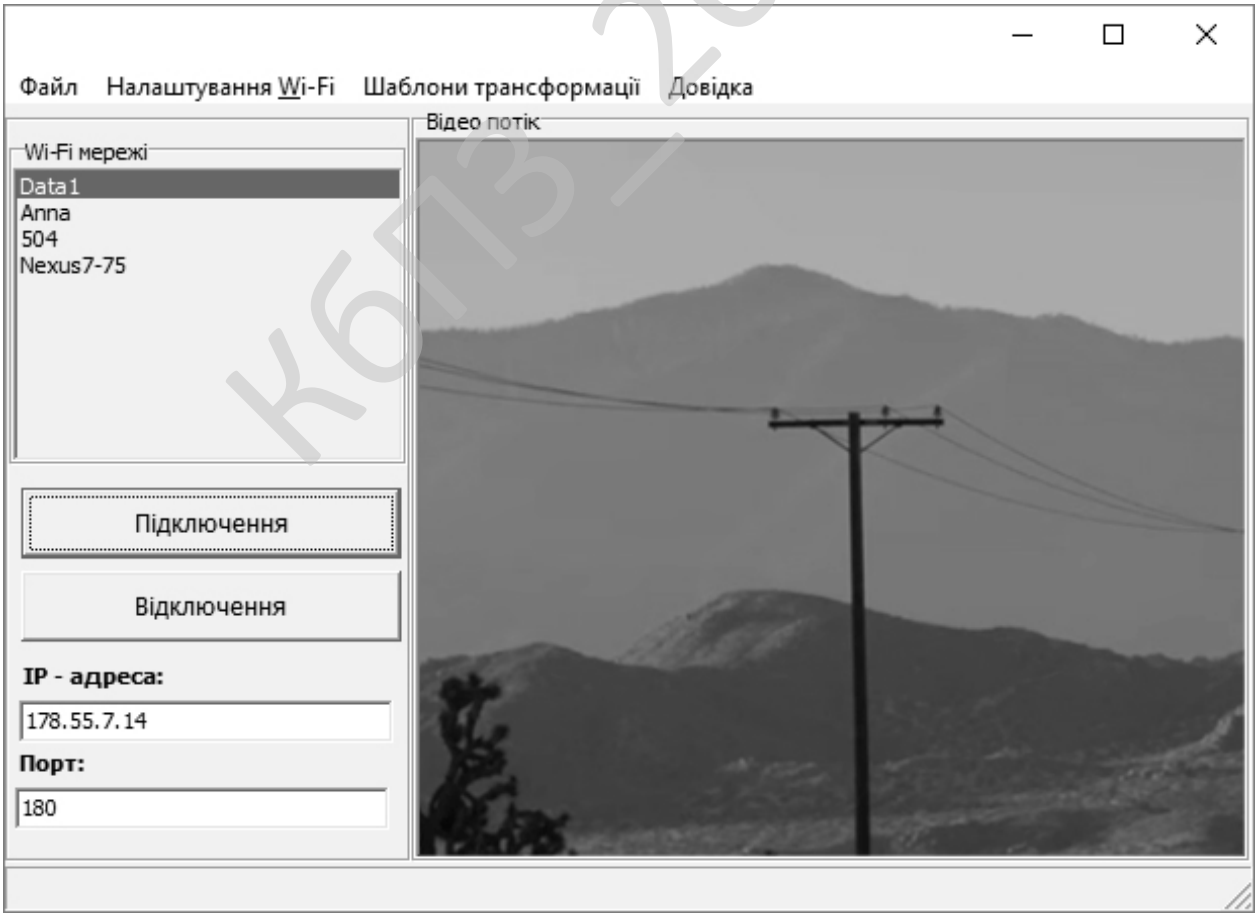

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

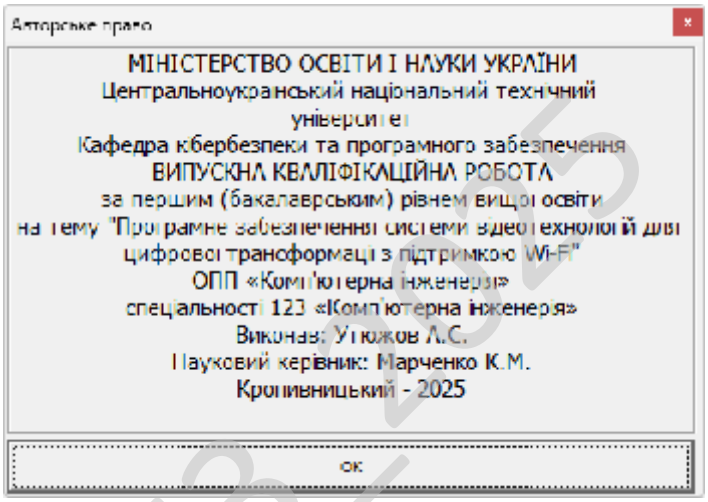

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача.

Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

– Досліджена система відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

– На основі отриманих результатів досліджень створена програмна реалізація системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi. Це

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 8845:2019.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. The MIT Press. 2022 1677 p.

2. Will Grant. 101 UX Principles. Packt Publishing. 2022. 432 p.

3. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

4. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

5. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

6. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

7. Peter Shirley, Steve Marschner. Fundamentals of Computer Graphics. 2009

8. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

9. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

10. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

11. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. - Д.: НГУ, 2016. - 187 с.

12. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

13. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

14. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

15. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447

16. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

17. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

18. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

19. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

20. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

21. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

22. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

23. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

24. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

25. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

26. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

27. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

28. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

29. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

30. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

31. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

32. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

33. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

34. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

35. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

36. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019,

Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

37. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

38. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

39. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

40. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

41. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

42. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

43. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated

Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

44. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

45. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

46. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

47. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

48. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

49. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

50. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 129 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Марченко К.М.

Програмне забезпечення системи відеотехнологій для цифрової трансформації з підтримкою Wi-Fi

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 27

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Імпортуємо необхідні модулі
import cv2
import socket
import threading
import time
import logging
import requests
import sys
import traceback
import struct

#Налаштовуємо базовий логінг для системи
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')

#Клас для управління Wi-Fi з'єднанням та перевіркою доступу до Інтернету
class WiFiManager:
 def __init__(self):
  #Ініціалізуємо змінну статусу з'єднання
  self.connected = False
  #Встановлюємо URL для перевірки підключення до Інтернету
  self.test_url = "http://www.google.com"
  #Ініціалізуємо інтервал перевірки з'єднання
  self.check_interval = 10
  #Ініціалізуємо змінну для зупинки перевірки
  self.stop_check = False

 def check_connection(self):
  #Перевірка з'єднання Wi-Fi шляхом виконання HTTP запиту
  try:
   #Надсилаємо GET запит до тестового URL з таймаутом 5 секунд
   response = requests.get(self.test_url, timeout=5)
   #Якщо отримано успішну відповідь, повертаємо True
   if response.status_code == 200:
    return True
   else:
    return False
  except Exception as e:
   #Логування помилки при невдалому з'єднанні
   logging.error("Error during Wi-Fi connection check: " + str(e))
   return False

 def start_monitoring(self):
  #Створюємо окремий потік для постійного моніторингу з'єднання
  self.monitor_thread = threading.Thread(target=self._monitor_connection)
  #Встановлюємо демон, щоб потік завершувався при завершенні програми
  self.monitor_thread.daemon = True
  #Запускаємо потік моніторингу
  self.monitor_thread.start()

 def _monitor_connection(self):
  #Безкінечний цикл перевірки з'єднання Wi-Fi
  while not self.stop_check:
   #Перевірка з'єднання
   self.connected = self.check_connection()
   #Логування статусу з'єднання
   logging.info("Wi-Fi connected: " + str(self.connected))
   #Очікування перед наступною перевіркою
   time.sleep(self.check_interval)

 def stop_monitoring(self):
  #Зупиняємо моніторинг з'єднання
  self.stop_check = True
```

```
  #Чекаємо завершення потоку моніторингу, якщо він працює
  if self.monitor_thread.is_alive():
   self.monitor_thread.join()

#Клас для захоплення та обробки відео з використанням OpenCV
class VideoProcessor:
 def __init__(self, video_source=0):

  #Зберігаємо параметр джерела відео
  self.video_source = video_source

  #Ініціалізуємо об'єкт захоплення відео
  self.capture = cv2.VideoCapture(self.video_source)

  #Перевіряємо, чи вдалося відкрити відеопотік
  if not self.capture.isOpened():

   #Логування помилки відкриття відеопотоку
   logging.error("Не вдалося відкрити відеопотік з джерела: " +
str(self.video_source))
   sys.exit(1)

  #Змінна для зупинки циклу захоплення
  self.stop_processing = False
  #Ініціалізуємо змінну для зберігання обробленого кадру
  self.processed_frame = None
  #Ініціалізуємо змінну для блокування доступу до кадру
  self.frame_lock = threading.Lock()

 def process_frame(self, frame):
  #Перетворюємо кадр у відтінки сірого
  gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

 #Накладаємо розмиття для зменшення шуму
  blurred_frame = cv2.GaussianBlur(gray_frame, (5, 5), 0)
  #Виконуємо детекцію країв за допомогою алгоритму Canny
  edges = cv2.Canny(blurred_frame, 50, 150)
  #Повертаємо оброблений кадр
  return edges

 def capture_loop(self):
  #Основний цикл захоплення та обробки відео
  while not self.stop_processing:
   #Читаємо кадр з відеопотоку
   ret, frame = self.capture.read()
   #Якщо кадр не зчитано, продовжуємо цикл
   if not ret:
    logging.warning("Кадр не зчитано, продовжуємо захоплення")
    continue
   #Обробляємо зчитаний кадр
   processed = self.process_frame(frame)
   #Отримуємо блокування для оновлення обробленого кадру
   with self.frame_lock:
    self.processed_frame = processed

   #Штучна затримка для симуляції обробки
   time.sleep(0.05)
  #Звільняємо ресурс відеопотоку після завершення циклу
  self.capture.release()

 def get_latest_frame(self):
  #Повертаємо останній оброблений кадр з блокуванням
  with self.frame_lock:
   return self.processed_frame



 def stop(self):
```

```
  #Зупиняємо цикл захоплення відео
  self.stop_processing = True

#Клас для серверу стрімінгу відео через Wi-Fi використовуючи сокети
class VideoStreamingServer:
 def __init__(self, host='0.0.0.0', port=8000, video_processor=None):

  #Зберігаємо параметри хоста та порту серверу
  self.host = host
  self.port = port

  #Зберігаємо посилання на об'єкт відеопроцесора для отримання кадрів
  self.video_processor = video_processor
  #Ініціалізуємо сокет серверу
  self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

  #Задаємо параметри повторного використання адреси
  self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  #Зв'язуємо сокет з хостом та портом
  self.server_socket.bind((self.host, self.port))

  #Слухаємо вхідні з'єднання
  self.server_socket.listen(5)
  #Змінна для зупинки серверу
  self.stop_server = False

 def start_server(self):
  #Створюємо окремий потік для прийому з'єднань
  self.accept_thread = threading.Thread(target=self._accept_connections)

  #Встановлюємо демон для потоку прийому з'єднань
  self.accept_thread.daemon = True

 #Запускаємо потік прийому з'єднань
  self.accept_thread.start()
  #Логування старту серверу
  logging.info("Відео стрім сервер запущено на порті: " + str(self.port))

 def _accept_connections(self):
  #Безкінечний цикл прийому з'єднань клієнтів
  while not self.stop_server:
   try:
    #Приймаємо вхідне з'єднання
    client_socket, addr = self.server_socket.accept()

    #Логування інформації про клієнта
    logging.info("Новий клієнт підключився: " + str(addr))

    #Створюємо окремий потік для обробки клієнта
    client_thread = threading.Thread(target=self._handle_client,
args=(client_socket,))
    #Встановлюємо демон для потоку обробки клієнта
    client_thread.daemon = True

    #Запускаємо потік обробки клієнта
    client_thread.start()
   except Exception as e:

    #Логування помилок прийому з'єднань
    logging.error("Помилка прийому з'єднання: " + str(e))
    break

 def _handle_client(self, client_socket):
  #Цикл передачі відео кадрів клієнту
  try:
   while not self.stop_server:
```

```
      #Отримуємо останній оброблений кадр з відеопроцесора
      frame = self.video_processor.get_latest_frame()
      #Якщо кадр відсутній, продовжуємо цикл
      if frame is None:
       time.sleep(0.01)
       continue

      #Кодуємо кадр у формат JPEG
      ret, jpeg = cv2.imencode('.jpg', frame)
      #Якщо кодування не вдалося, продовжуємо цикл
      if not ret:
       logging.warning("Не вдалося закодувати кадр")
       continue

      #Перетворюємо кодування у байти
      jpeg_bytes = jpeg.tobytes()
      #Формуємо заголовок з розміром кадру
      header = struct.pack("L", len(jpeg_bytes))
      #Надсилаємо заголовок клієнту
      client_socket.sendall(header)
      #Надсилаємо дані кадру клієнту
      client_socket.sendall(jpeg_bytes)

      #Штучна затримка для симуляції стрімінгу
      time.sleep(0.05)
    except Exception as e:
     #Логування винятків під час передачі даних клієнту
     logging.error("Помилка обробки клієнта: " + str(e))
    finally:
     #Закриваємо сокет клієнта після завершення передачі
     client_socket.close()

  def stop(self):
   #Зупиняємо сервер стрімінгу відео
   self.stop_server = True
   #Закриваємо серверний сокет
   self.server_socket.close()

#Клас для обробки запитів клієнтів та взаємодії з іншими системами
class CommandHandler:
  def __init__(self, video_processor, wifi_manager):
   #Зберігаємо посилання на об'єкт відеопроцесора
   self.video_processor = video_processor
   #Зберігаємо посилання на об'єкт Wi-Fi менеджера
   self.wifi_manager = wifi_manager
   #Ініціалізуємо змінну для зупинки обробки команд
   self.stop_commands = False

  def start_command_loop(self):
   #Створюємо окремий потік для циклічної обробки команд
   self.command_thread = threading.Thread(target=self._command_loop)
   #Встановлюємо демон для потоку обробки команд
   self.command_thread.daemon = True
   #Запускаємо потік обробки команд
   self.command_thread.start()

  def _command_loop(self):
   #Безкінечний цикл для прийому команд від користувача
   while not self.stop_commands:
    try:
     #Запитуємо команду від користувача через стандартний ввід
     command = input("Введіть команду (start/stop/status/exit): ")
     #Перевіряємо введену команду
     if command.lower() == "start":

      #Логування команди старту
      logging.info("Команда старт отримана")
```

```
      #Можлива логіка для запуску додаткових функцій
      self._execute_start()
     elif command.lower() == "stop":
      #Логування команди зупинки
      logging.info("Команда стоп отримана")
      #Можлива логіка для зупинки функцій
      self._execute_stop()
     elif command.lower() == "status":

      #Логування запиту статусу
      logging.info("Команда статус отримана")
      #Виведення статусу системи
      self._execute_status()
     elif command.lower() == "exit":
      #Логування команди виходу
      logging.info("Команда виходу отримана")
      #Завершення роботи програми
      self.stop_commands = True
      self._execute_exit()
     else:
      #Виведення повідомлення про невідому команду
      logging.warning("Невідома команда: " + command)
    except Exception as e:
     #Логування помилок під час обробки команд
     logging.error("Помилка при обробці команди: " + str(e))
     traceback.print_exc()

 def _execute_start(self):
  #Функція для виконання команди старту
  logging.info("Запуск додаткових функцій системи")
  #Симуляція додаткової роботи системи
  time.sleep(1)
  logging.info("Додаткові функції системи запущені")

 def _execute_stop(self):
  #Функція для виконання команди зупинки
  logging.info("Зупинка додаткових функцій системи")
  #Симуляція процесу зупинки
  time.sleep(1)
  logging.info("Додаткові функції системи зупинені")

 def _execute_status(self):
  #Функція для виконання команди статусу
  wifi_status = self.wifi_manager.connected
  frame = self.video_processor.get_latest_frame()
  if frame is not None:
   frame_status = "Оброблений кадр доступний"
  else:
   frame_status = "Оброблений кадр відсутній"
  #Виведення статусу системи
  print("Статус Wi-Fi: " + str(wifi_status))
  print("Статус відео: " + frame_status)

 def _execute_exit(self):
  #Функція для виконання команди виходу з програми
  logging.info("Завершення роботи системи")
  #Зупиняємо відеопроцесор
  self.video_processor.stop()
  #Зупиняємо моніторинг Wi-Fi
  self.wifi_manager.stop_monitoring()
  #Зупиняємо сервер відеострімінгу
  logging.info("Вихід з програми через команду exit")
  sys.exit(0)

#Основна функція, що ініціалізує всі компоненти системи
def main():
 #Створення об'єкта Wi-Fi менеджера
 wifi_manager = WiFiManager()
 #Запуск моніторингу з'єднання Wi-Fi
```

```
 wifi_manager.start_monitoring()
 #Створення об'єкта відеопроцесора з джерелом відео 0
 video_processor = VideoProcessor(video_source=0)
 #Створення окремого потоку для циклу захоплення відео
 video_thread = threading.Thread(target=video_processor.capture_loop)
 #Встановлюємо демон для потоку захоплення відео
 video_thread.daemon = True
 #Запускаємо потік захоплення відео
 video_thread.start()
 #Створення об'єкта сервера стрімінгу відео на порту 8000
 streaming_server = VideoStreamingServer(host='0.0.0.0', port=8000,
video_processor=video_processor)
 #Запуск серверу стрімінгу відео
 streaming_server.start_server()
 #Створення об'єкта обробника команд для взаємодії з користувачем
 command_handler = CommandHandler(video_processor, wifi_manager)
 #Запуск циклу обробки команд у окремому потоці
 command_handler.start_command_loop()
 #Основний цикл для підтримки роботи програми
 try:
  while True:
   #Штучна затримка для зменшення навантаження на процесор
   time.sleep(1)
 except KeyboardInterrupt:
  #Обробка виключення через натискання клавіш
  logging.info("KeyboardInterrupt отримано, завершуємо роботу")
  #Зупиняємо сервер стрімінгу відео
  streaming_server.stop()
  #Зупиняємо відеопроцесор
  video_processor.stop()
  #Зупиняємо моніторинг Wi-Fi
  wifi_manager.stop_monitoring()
  #Завершення програми
  sys.exit(0)

#Точка входу в програму
if __name__ == "__main__":
 #Виклик основної функції для запуску системи
 main()
```

Файл multicam.py

```
#!/usr/bin/env python3
#Файл: multicam.py
#Багатокамерний режим для одночасного захоплення відео з декількох джерел
#Імпортуємо необхідні модулі для роботи з відео та потоками
import cv2
import threading
import time
import logging

#Клас для управління декількома камерами з максимальною деталізацією
class MultiCameraManager:
    #Конструктор класу приймає список джерел відео
    def __init__(self, camera_sources):
        #Зберігаємо список джерел камер
        self.camera_sources = camera_sources
        #Словник для збереження об'єктів відеозахоплення для кожного джерела
        self.captures = {}
        #Словник для збереження останнього зчитаного кадру для кожного джерела
        self.frames = {}
        #Словник для синхронізації доступу до кадрів
        self.frame_locks = {}
        #Список потоків захоплення для кожного джерела
        self.threads = []
        #Прапорець для зупинки роботи потоків
        self.stop_flag = False
        #Налаштовуємо базовий логінг для системи
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')
        #Ініціалізація камер
        self._initialize_cameras()

    #Метод для ініціалізації всіх камер
    def _initialize_cameras(self):
        for source in self.camera_sources:
            try:
                #Створюємо об'єкт захоплення для поточного джерела
                capture = cv2.VideoCapture(source)
                #Перевірка відкриття відеопотоку
                if not capture.isOpened():
                    logging.error("Не вдалося відкрити камеру з джерела: " +
str(source))
                    continue
                #Зберігаємо об'єкт відеозахоплення
                self.captures[source] = capture
                #Ініціалізуємо останній кадр як None
                self.frames[source] = None
                #Створюємо блокування для захисту доступу до кадру
                self.frame_locks[source] = threading.Lock()
                logging.info("Камера з джерела " + str(source) + " успішно
ініціалізована")
            except Exception as ex:
```

```
                logging.error("Виняток при ініціалізації камери " + str(source)
+ ": " + str(ex))

    #Метод для запуску потоків захоплення кадрів для всіх камер
    def start_all(self):
        for source in self.captures:
            #Створюємо окремий потік для захоплення кадрів з конкретного джерела
            thread = threading.Thread(target=self._capture_loop, args=(source,))
            thread.daemon = True
            thread.start()
            self.threads.append(thread)
            logging.info("Запуск потоку захоплення для камери з джерела: " +
str(source))

    #Приватний метод циклічного захоплення кадрів з конкретного джерела
    def _capture_loop(self, source):
        capture = self.captures[source]
        while not self.stop_flag:
            try:
                ret, frame = capture.read()
                if ret:
                    with self.frame_locks[source]:
                        self.frames[source] = frame
                    time.sleep(0.03)
                else:
                    logging.warning("Не вдалося зчитати кадр з камери " +
str(source))
                    time.sleep(0.1)
            except Exception as e:
                logging.error("Помилка у потоці камери " + str(source) + ": " +
str(e))
                time.sleep(0.1)
        capture.release()
        logging.info("Зупинка захоплення для камери " + str(source))

    #Метод для отримання останнього кадру з вказаної камери
    def get_frame(self, source):
        if source in self.frames:
            with self.frame_locks[source]:
                return self.frames[source]
        else:
            logging.error("Камера з джерела " + str(source) + " не знайдена")
            return None

    #Метод для зупинки всіх потоків захоплення
    def stop_all(self):
        self.stop_flag = True
        for thread in self.threads:
            thread.join()
        logging.info("Всі потоки захоплення зупинено")
```

Файл video_archive_cloud.py

```
#!/usr/bin/env python3
#Файл: video_archive_cloud.py
#Модуль архівування відео та інтеграції з хмарними сервісами для резервного
копіювання
#Імпортуємо необхідні модулі для роботи з файлами, потоками та HTTP запитами
import cv2
import threading
import time
import os
import logging
import requests

#Клас для архівування відео з максимальною деталізацією
class VideoArchiver:
    #Конструктор класу архіватора
    def __init__(self, output_directory="video_archives", archive_format="avi",
frame_rate=20.0, frame_size=(640,480)):
        #Встановлюємо директорію для збереження архівів
        self.output_directory = output_directory
        if not os.path.exists(self.output_directory):
            os.makedirs(self.output_directory)
        #Задаємо формат файлу архіву
        self.archive_format = archive_format
        #Встановлюємо частоту кадрів
        self.frame_rate = frame_rate
        #Задаємо розмір кадру
        self.frame_size = frame_size
        #Ініціалізуємо об'єкт записувача відео як None
        self.video_writer = None
        #Прапорець для контролю запису
        self.recording = False
        #Налаштовуємо логування
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')

    #Метод для запуску архівування відео з генерацією унікального імені файлу
    def start_archiving(self, filename_prefix="archive"):
        timestamp = time.strftime("%Y%m%d_%H%M%S")
        filename = f"{filename_prefix}_{timestamp}.{self.archive_format}"
        file_path = os.path.join(self.output_directory, filename)
        fourcc = cv2.VideoWriter_fourcc(*'XVID')
        self.video_writer = cv2.VideoWriter(file_path, fourcc, self.frame_rate,
self.frame_size)
        self.recording = True
        logging.info("Розпочато архівування відео у файл: " + file_path)
        return file_path

    #Метод для запису кадру до архіву з детальним логуванням
    def write_frame(self, frame):
        if self.recording and self.video_writer is not None:
            try:
```

```
                self.video_writer.write(frame)
                logging.debug("Кадр успішно записано до архіву")
            except Exception as e:
                logging.error("Помилка запису кадру: " + str(e))

    #Метод для зупинки архівування відео та звільнення ресурсів
    def stop_archiving(self):
        if self.recording and self.video_writer is not None:
            self.video_writer.release()
            self.recording = False
            logging.info("Архівування відео завершено")

#Клас для інтеграції з хмарними сервісами для резервного копіювання відео
class CloudUploader:
    #Конструктор класу хмарного завантажувача
    def __init__(self, cloud_endpoint="http://example.com/upload",
api_key="YOUR_API_KEY"):
        self.cloud_endpoint = cloud_endpoint
        self.api_key = api_key
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')

    #Метод для завантаження файлу у хмару з максимальною деталізацією процесу
    def upload_file(self, file_path):
        try:
            with open(file_path, 'rb') as file_to_upload:
                files = {'file': file_to_upload}
                data = {'api_key': self.api_key}
                response = requests.post(self.cloud_endpoint, files=files,
data=data)
                if response.status_code == 200:
                    logging.info("Файл успішно завантажено у хмару: " +
file_path)
                else:
                    logging.error("Помилка завантаження файлу, статус код: " +
str(response.status_code))
        except Exception as ex:
            logging.error("Виняток під час завантаження файлу: " + str(ex))
```

Файл face_recognition.py

```
#!/usr/bin/env python3
#Файл: face_recognition.py
#Модуль розпізнавання облич у режимі реального часу з використанням алгоритмів
OpenCV
#Імпортуємо необхідні модулі для обробки зображень та потокової роботи
import cv2
import threading
import time
import logging

#Клас для розпізнавання облич з максимальною деталізацією
class FaceRecognizer:
    #Конструктор класу приймає шлях до XML файлу класифікатора Haar
    def __init__(self, cascade_path="haarcascade_frontalface_default.xml"):
        self.face_cascade = cv2.CascadeClassifier(cascade_path)
        if self.face_cascade.empty():
            logging.error("Не вдалося завантажити класифікатор для облич з
файлу: " + cascade_path)
        self.active = True
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')

    #Метод для обробки кадру і розпізнавання облич з деталізованим логуванням
    def recognize_faces(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = self.face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        logging.debug("Знайдено облич: " + str(len(faces)))
        return frame, faces

    #Метод для запуску циклічного розпізнавання облич у окремому потоці
    def start_recognition(self, video_source=0):
        thread = threading.Thread(target=self._recognition_loop,
args=(video_source,))
        thread.daemon = True
        thread.start()

    #Приватний метод циклічного захоплення кадрів і розпізнавання облич
    def _recognition_loop(self, video_source):
        cap = cv2.VideoCapture(video_source)
        if not cap.isOpened():
            logging.error("Не вдалося відкрити відеопотік для розпізнавання
облич з джерела: " + str(video_source))
            return
        while self.active:
            ret, frame = cap.read()
            if not ret:
                logging.warning("Не вдалося зчитати кадр для розпізнавання
облич")
```

```
            time.sleep(0.1)
            continue
        processed_frame, faces = self.recognize_faces(frame)
        time.sleep(0.05)
    cap.release()
```

Файл motion_detection.py

```
#!/usr/bin/env python3
#Файл: motion_detection.py
#Модуль детекції руху з високою деталізацією та сповіщенням користувача
#Імпортуємо необхідні модулі для обробки відео та роботи з зображеннями
import cv2
import numpy as np
import threading
import time
import logging

#Клас для детекції руху з використанням алгоритму порівняння кадрів
class MotionDetector:
    #Конструктор класу з параметрами чутливості та мінімальної площі руху
    def __init__(self, sensitivity=25, min_area=500):
        self.sensitivity = sensitivity
        self.min_area = min_area
        self.previous_frame = None
        self.active = True
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')

    #Метод для визначення наявності руху у кадрі
    def detect_motion(self, frame):
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray_frame = cv2.GaussianBlur(gray_frame, (21, 21), 0)
        motion_detected = False
        contours = []
        if self.previous_frame is None:
            self.previous_frame = gray_frame
            return motion_detected, contours
        frame_delta = cv2.absdiff(self.previous_frame, gray_frame)
        thresh = cv2.threshold(frame_delta, self.sensitivity, 255,
cv2.THRESH_BINARY)[1]
        thresh = cv2.dilate(thresh, None, iterations=2)
        contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        for contour in contours:
            if cv2.contourArea(contour) < self.min_area:
                continue
            motion_detected = True
            logging.debug("Детектовано рух, площа контуру: " +
str(cv2.contourArea(contour)))
        self.previous_frame = gray_frame
        return motion_detected, contours

    #Метод для запуску циклічного спостереження за рухом у окремому потоці
    def start_detection(self, video_source=0):
        thread = threading.Thread(target=self._detection_loop,
args=(video_source,))
        thread.daemon = True
```

```
        thread.start()

    #Приватний метод циклічного аналізу руху
    def _detection_loop(self, video_source):
        cap = cv2.VideoCapture(video_source)
        if not cap.isOpened():
            logging.error("Не вдалося відкрити відеопотік для детекції руху з
джерела: " + str(video_source))
            return
        while self.active:
            ret, frame = cap.read()
            if not ret:
                logging.warning("Кадр для детекції руху не зчитано")
                time.sleep(0.1)
                continue
            motion, contours = self.detect_motion(frame)
            if motion:
                logging.info("Рух виявлено у кадрі")
            time.sleep(0.05)
        cap.release()
```

Файл notifications.py

```
#!/usr/bin/env python3
#Файл: notifications.py
#Модуль системи сповіщень для надсилання повідомлень на email або мобільний пристрій
#Імпортуємо необхідні модулі для роботи з електронною поштою та потоками
import smtplib
import threading
import time
import logging
from email.mime.text import MIMEText

#Клас для надсилання сповіщень з максимальною деталізацією
class Notifier:
    #Конструктор класу налаштовує SMTP параметри
    def __init__(self, smtp_server="smtp.example.com", smtp_port=587, username="user@example.com", password="password"):
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
        self.username = username
        self.password = password
        self.active = True
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

    #Метод для надсилання email сповіщення з докладним описом процесу
    def send_email(self, subject, message, recipient):
        try:
            msg = MIMEText(message)
            msg["Subject"] = subject
            msg["From"] = self.username
            msg["To"] = recipient
            server = smtplib.SMTP(self.smtp_server, self.smtp_port)
            server.starttls()
            server.login(self.username, self.password)
            server.sendmail(self.username, [recipient], msg.as_string())
            server.quit()
            logging.info("Email сповіщення успішно надіслано на: " + recipient)
        except Exception as ex:
            logging.error("Помилка надсилання email: " + str(ex))

    #Метод для циклічного надсилання тестових сповіщень у окремому потоці
    def start_periodic_notifications(self, recipient, interval=60):
        thread = threading.Thread(target=self._notification_loop, args=(recipient, interval))
        thread.daemon = True
        thread.start()

    #Приватний метод циклічного надсилання сповіщень
    def _notification_loop(self, recipient, interval):
        while self.active:
```

```
            self.send_email("Тестове сповіщення", "Це тестове повідомлення від
системи сповіщень.", recipient)
            time.sleep(interval)
```

Файл video_encryption.py

```
#!/usr/bin/env python3
#Файл: video_encryption.py
#Модуль шифрування відеопотоку для забезпечення безпеки передачі даних
#Імпортуємо необхідні модулі для роботи з криптографією та системними ресурсами
import threading
import time
import logging
import os
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

#Клас для шифрування та дешифрування відеокадрів з максимальною деталізацією
class VideoEncryptor:
    #Конструктор класу приймає ключ шифрування або генерує його випадково
    def __init__(self, key=None):
        self.key = key if key is not None else os.urandom(16)
        self.iv = os.urandom(16)
        self.cipher = AES.new(self.key, AES.MODE_CBC, self.iv)
        self.active = True
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')

    #Метод для шифрування кадру, представленного у вигляді байтів
    def encrypt_frame(self, frame_bytes):
        try:
            padded_data = pad(frame_bytes, AES.block_size)
            encrypted_data = self.cipher.encrypt(padded_data)
            logging.debug("Кадр успішно зашифровано")
            return encrypted_data
        except Exception as e:
            logging.error("Помилка шифрування кадру: " + str(e))
            return None

    #Метод для дешифрування зашифрованих даних кадру
    def decrypt_frame(self, encrypted_bytes):
        try:
            decrypted_padded = self.cipher.decrypt(encrypted_bytes)
            decrypted_data = unpad(decrypted_padded, AES.block_size)
            logging.debug("Кадр успішно дешифровано")
            return decrypted_data
        except Exception as e:
            logging.error("Помилка дешифрування кадру: " + str(e))
            return None

    #Метод для оновлення вектору ініціалізації з максимальною деталізацією
    def update_iv(self):
        self.iv = os.urandom(16)
```

```
        self.cipher = AES.new(self.key, AES.MODE_CBC, self.iv)
        logging.info("Вектор ініціалізації оновлено")
```

Файл web_interface.py

```
#!/usr/bin/env python3
#Файл: web_interface.py
#Модуль веб-інтерфейсу управління системою для моніторингу та керування
#Імпортуємо необхідні модулі для створення веб-додатку
from flask import Flask, jsonify, request, render_template_string
import threading
import time
import logging

#Створення об'єкта Flask для веб-інтерфейсу
app = Flask(__name__)

#Глобальна змінна для збереження статусу системи
system_status = {
    "wifi": True,
    "video": "Активний",
    "last_update": time.strftime("%Y-%m-%d %H:%M:%S")
}

#Маршрут для отримання статусу системи у форматі JSON
@app.route('/status', methods=['GET'])
def get_status():
    return jsonify(system_status)

#Маршрут для керування системою через POST запит
@app.route('/control', methods=['POST'])
def control_system():
    data = request.json
    if data is None:
        return jsonify({"error": "Невірний формат запиту"}), 400
    action = data.get("action", "")
    if action == "restart":
        system_status["video"] = "Перезапуск"
        time.sleep(1)
        system_status["video"] = "Активний"
        system_status["last_update"] = time.strftime("%Y-%m-%d %H:%M:%S")
        return jsonify({"message": "Система перезапущена"})
    elif action == "shutdown":
        system_status["video"] = "Вимкнено"
        system_status["last_update"] = time.strftime("%Y-%m-%d %H:%M:%S")
        return jsonify({"message": "Система вимкнена"})
    else:
        return jsonify({"error": "Невідома дія"}), 400

#Маршрут для відображення HTML сторінки зі статусом системи
@app.route('/')
def index():
    html_content = """
```

```
    <html>
    <head>
    <title>Статус Системи</title>
    </head>
    <body>
    <h1>Поточний статус системи</h1>
    <ul>
        <li>Wi-Fi: {{ wifi_status }}</li>
        <li>Відео: {{ video_status }}</li>
        <li>Оновлено: {{ last_update }}</li>
    </ul>
    </body>
    </html>
    """
    return render_template_string(html_content,
wifi_status=system_status["wifi"], video_status=system_status["video"],
last_update=system_status["last_update"])

#Функція для запуску веб-сервера у окремому потоці
def start_web_interface(host="0.0.0.0", port=5000):
    def run_app():
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')
        app.run(host=host, port=port)
    thread = threading.Thread(target=run_app)
    thread.daemon = True
    thread.start()
```

Файл statistics_analysis.py

```
#!/usr/bin/env python3
#Файл: statistics_analysis.py
#Модуль аналізу статистики для збору та візуалізації даних про якість системи
#Імпортуємо необхідні модулі для збору даних, роботи з часом та генерації
випадкових чисел
import threading
import time
import logging
import random

#Клас для аналізу статистики з максимальною деталізацією
class StatsAnalyzer:
    #Конструктор класу, ініціалізує список даних
    def __init__(self):
        self.data = []
        self.active = True
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')

    #Метод для збору статистичних даних з детальним логуванням
    def collect_stats(self):
        stats = {
            "timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
            "wifi_quality": random.randint(70, 100),
            "frame_rate": random.uniform(18.0, 25.0),
            "cpu_usage": random.uniform(20.0, 80.0)
        }
        self.data.append(stats)
        logging.debug("Зібрано статистику: " + str(stats))

    #Метод для запуску циклічного збору даних у окремому потоці
    def start_collection(self, interval=5):
        thread = threading.Thread(target=self._collection_loop,
args=(interval,))
        thread.daemon = True
        thread.start()

    #Приватний метод циклічного збору статистики
    def _collection_loop(self, interval):
        while self.active:
            self.collect_stats()
            time.sleep(interval)

    #Метод для отримання зібраних статистичних даних
    def get_stats(self):
        return self.data
```

Файл iot_integration.py

```
#!/usr/bin/env python3
#Файл: iot_integration.py
#Модуль інтеграції IoT пристроїв для автоматизації управління середовищем
#Імпортуємо необхідні модулі для роботи з потоками, часом та генерації
випадкових затримок
import threading
import time
import logging
import random

#Клас для інтеграції з IoT пристроями з максимальною деталізацією
class IoTIntegrator:
    #Конструктор класу приймає параметри брокера MQTT
    def __init__(self, broker_address="mqtt.example.com", broker_port=1883):
        self.broker_address = broker_address
        self.broker_port = broker_port
        self.active = True
        #Список симульованих IoT пристроїв
        self.devices = ["SmartLight", "Thermostat", "SecuritySensor"]
        logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -
%(levelname)s - %(message)s')

    #Метод для симуляції підключення до IoT пристроїв
    def connect_devices(self):
        for device in self.devices:
            logging.info("Підключення до пристрою: " + device)
            time.sleep(0.5)
        logging.info("Всі IoT пристрої успішно підключені")

    #Метод для відправки команди на IoT пристрій з деталізованим логуванням
    def send_command(self, device, command):
        if device in self.devices:
            logging.info("Надсилання команди '" + command + "' до пристрою: " +
device)
            time.sleep(random.uniform(0.1, 0.5))
            logging.info("Команда '" + command + "' успішно виконана на
пристрої: " + device)
        else:
            logging.error("Пристрій " + device + " не знайдено")

    #Метод для запуску циклічного моніторингу та управління IoT пристроями у
окремому потоці
    def start_integration(self):
        thread = threading.Thread(target=self._integration_loop)
        thread.daemon = True
        thread.start()

    #Приватний метод циклічного управління IoT пристроями
    def _integration_loop(self):
        self.connect_devices()
        while self.active:
```

```
            for device in self.devices:
                self.send_command(device, "status_check")
            time.sleep(10)
```

Файл MultiCameraManager.py

```
import cv2
import threading
import time
import logging
import os
import requests
import numpy as np
from flask import Flask, jsonify, request, render_template_string

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')

class MultiCameraManager:
    def __init__(self, camera_sources):
        self.camera_sources = camera_sources
        self.captures = {}
        self.frames = {}
        self.frame_locks = {}
        self.threads = []
        self.stop_flag = False
        self._initialize_cameras()
    def _initialize_cameras(self):
        for source in self.camera_sources:
            try:
                capture = cv2.VideoCapture(source)
                if not capture.isOpened():
                    logging.error("Не вдалося відкрити камеру з джерела: " +
str(source))
                    continue
                self.captures[source] = capture
                self.frames[source] = None
                self.frame_locks[source] = threading.Lock()
                logging.info("Камера з джерела " + str(source) + " успішно
ініціалізована")
            except Exception as ex:
                logging.error("Виняток при ініціалізації камери " + str(source)
+ ": " + str(ex))
    def start_all(self):
        for source in self.captures:
            thread = threading.Thread(target=self._capture_loop, args=(source,))
            thread.daemon = True
            thread.start()
            self.threads.append(thread)
            logging.info("Запуск потоку захоплення для камери з джерела: " +
str(source))
    def _capture_loop(self, source):
        capture = self.captures[source]
```

```
        while not self.stop_flag:
            try:
                ret, frame = capture.read()
                if ret:
                    with self.frame_locks[source]:
                        self.frames[source] = frame
                    time.sleep(0.03)
                else:
                    logging.warning("Не вдалося зчитати кадр з камери " +
str(source))
                    time.sleep(0.1)
            except Exception as e:
                logging.error("Помилка у потоці камери " + str(source) + ": " +
str(e))
                time.sleep(0.1)
        capture.release()
        logging.info("Зупинка захоплення для камери " + str(source))
    def get_frame(self, source):
        if source in self.frames:
            with self.frame_locks[source]:
                return self.frames[source]
        else:
            logging.error("Камера з джерела " + str(source) + " не знайдена")
            return None
    def stop_all(self):
        self.stop_flag = True
        for thread in self.threads:
            thread.join()
        logging.info("Всі потоки захоплення зупинено")

class VideoArchiver:
    def __init__(self, output_directory="video_archives", archive_format="avi",
frame_rate=20.0, frame_size=(640,480)):
        self.output_directory = output_directory
        if not os.path.exists(self.output_directory):
            os.makedirs(self.output_directory)
        self.archive_format = archive_format
        self.frame_rate = frame_rate
        self.frame_size = frame_size
        self.video_writer = None
        self.recording = False
    def start_archiving(self, filename_prefix="archive"):
        timestamp = time.strftime("%Y%m%d_%H%M%S")
        filename = f"{filename_prefix}_{timestamp}.{self.archive_format}"
        file_path = os.path.join(self.output_directory, filename)
        fourcc = cv2.VideoWriter_fourcc(*'XVID')
        self.video_writer = cv2.VideoWriter(file_path, fourcc, self.frame_rate,
self.frame_size)
        self.recording = True
        logging.info("Розпочато архівування відео у файл: " + file_path)
        return file_path
    def write_frame(self, frame):
        if self.recording and self.video_writer is not None:
```

```
            try:
                self.video_writer.write(frame)
                logging.debug("Кадр успішно записано до архіву")
            except Exception as e:
                logging.error("Помилка запису кадру: " + str(e))
    def stop_archiving(self):
        if self.recording and self.video_writer is not None:
            self.video_writer.release()
            self.recording = False
            logging.info("Архівування відео завершено")

class CloudUploader:
    def __init__(self, cloud_endpoint="http://example.com/upload",
api_key="YOUR_API_KEY"):
        self.cloud_endpoint = cloud_endpoint
        self.api_key = api_key
    def upload_file(self, file_path):
        try:
            with open(file_path, 'rb') as file_to_upload:
                files = {'file': file_to_upload}
                data = {'api_key': self.api_key}
                response = requests.post(self.cloud_endpoint, files=files,
data=data)
                if response.status_code == 200:
                    logging.info("Файл успішно завантажено у хмару: " +
file_path)
                else:
                    logging.error("Помилка завантаження файлу, статус код: " +
str(response.status_code))
        except Exception as ex:
            logging.error("Виняток під час завантаження файлу: " + str(ex))

class FaceRecognizer:
    def __init__(self, cascade_path="haarcascade_frontalface_default.xml"):
        self.face_cascade = cv2.CascadeClassifier(cascade_path)
        if self.face_cascade.empty():
            logging.error("Не вдалося завантажити класифікатор для облич з
файлу: " + cascade_path)
        self.active = True
    def recognize_faces(self, frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = self.face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        logging.debug("Знайдено облич: " + str(len(faces)))
        return frame, faces
    def start_recognition(self, video_source=0):
        thread = threading.Thread(target=self._recognition_loop,
args=(video_source,))
        thread.daemon = True
        thread.start()
    def _recognition_loop(self, video_source):
```

```
        cap = cv2.VideoCapture(video_source)
        if not cap.isOpened():
            logging.error("Не вдалося відкрити відеопотік для розпізнавання
облич з джерела: " + str(video_source))
            return
        while self.active:
            ret, frame = cap.read()
            if not ret:
                logging.warning("Не вдалося зчитати кадр для розпізнавання
облич")
                time.sleep(0.1)
                continue
            processed_frame, faces = self.recognize_faces(frame)
            time.sleep(0.05)
        cap.release()

class MotionDetector:
    def __init__(self, sensitivity=25, min_area=500):
        self.sensitivity = sensitivity
        self.min_area = min_area
        self.previous_frame = None
        self.active = True
    def detect_motion(self, frame):
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray_frame = cv2.GaussianBlur(gray_frame, (21, 21), 0)
        motion_detected = False
        contours = []
        if self.previous_frame is None:
            self.previous_frame = gray_frame
            return motion_detected, contours
        frame_delta = cv2.absdiff(self.previous_frame, gray_frame)
        thresh = cv2.threshold(frame_delta, self.sensitivity, 255,
cv2.THRESH_BINARY)[1]
        thresh = cv2.dilate(thresh, None, iterations=2)
        contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        for contour in contours:
            if cv2.contourArea(contour) < self.min_area:
                continue
            motion_detected = True
            logging.debug("Детектовано рух, площа контуру: " +
str(cv2.contourArea(contour)))
        self.previous_frame = gray_frame
        return motion_detected, contours
    def start_detection(self, video_source=0):
        thread = threading.Thread(target=self._detection_loop,
args=(video_source,))
        thread.daemon = True
        thread.start()
    def _detection_loop(self, video_source):
        cap = cv2.VideoCapture(video_source)
        if not cap.isOpened():
```

```
            logging.error("Не вдалося відкрити відеопотік для детекції руху з
джерела: " + str(video_source))
            return
        while self.active:
            ret, frame = cap.read()
            if not ret:
                logging.warning("Кадр для детекції руху не зчитано")
                time.sleep(0.1)
                continue
            motion, contours = self.detect_motion(frame)
            if motion:
                logging.info("Рух виявлено у кадрі")
            time.sleep(0.05)
        cap.release()

app = Flask(__name__)
system_status = {
    "wifi": True,
    "video": "Активний",
    "last_update": time.strftime("%Y-%m-%d %H:%M:%S")
}
@app.route('/status', methods=['GET'])
def get_status():
    return jsonify(system_status)
@app.route('/control', methods=['POST'])
def control_system():
    data = request.json
    if data is None:
        return jsonify({"error": "Невірний формат запиту"}), 400
    action = data.get("action", "")
    if action == "restart":
        system_status["video"] = "Перезапуск"
        time.sleep(1)
        system_status["video"] = "Активний"
        system_status["last_update"] = time.strftime("%Y-%m-%d %H:%M:%S")
        return jsonify({"message": "Система перезапущена"})
    elif action == "shutdown":
        system_status["video"] = "Вимкнено"
        system_status["last_update"] = time.strftime("%Y-%m-%d %H:%M:%S")
        return jsonify({"message": "Система вимкнена"})
    else:
        return jsonify({"error": "Невідома дія"}), 400
@app.route('/')
def index():
    html_content = """
    <html>
    <head>
    <title>Статус Системи</title>
    </head>
    <body>
    <h1>Поточний статус системи</h1>
    <ul>
        <li>Wi-Fi: {{ wifi_status }}</li>
```

```
        <li>Відео: {{ video_status }}</li>
        <li>Оновлено: {{ last_update }}</li>
    </ul>
    </body>
    </html>
    """
    return render_template_string(html_content,
wifi_status=system_status["wifi"], video_status=system_status["video"],
last_update=system_status["last_update"])
def start_web_interface(host="0.0.0.0", port=5000):
    def run_app():
        app.run(host=host, port=port)
    thread = threading.Thread(target=run_app)
    thread.daemon = True
    thread.start()
```