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 гібридного кабельного відеонагляду з використанням технології PFCS”

КбПЗ_2025

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Рескаленко Я.Ю.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 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

Рескаленку Ярославу Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи гібридного кабельного відеонагляду з використанням технології PFCS*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 47-02 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 гібридного кабельного відеонагляду з використанням технології PFCS*

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*

КВПЗ-2025

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.	

КбПЗ_2025

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Рескаленко Я.Ю.
(прізвище та ініціали)

АНОТАЦІЯ

Рескаленко Я.Ю. Програмне забезпечення системи гібридного кабельного відеонагляду з використанням технології PFCS. 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 гібридного кабельного відеонагляду з використанням технології PFCS.

Метою розробки є програмне забезпечення системи гібридного кабельного відеонагляду з використанням технології PFCS.

Результат роботи – програмна реалізація системи гібридного кабельного відеонагляду з використанням технології PFCS.

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, відео нагляд, PFCS

КПЗ – 2025

ABSTRACT

Reskalenko Ya.Yu. Software for a hybrid cable video surveillance system using PFCS technology. 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, which is intended for a hybrid cable video surveillance system using PFCS technology.

The purpose of the development is software for a hybrid cable video surveillance system using PFCS technology.

The result of the work is a software implementation of a hybrid cable video surveillance system using PFCS technology.

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, video surveillance, PFCS

КОПЗ_2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи..... 5
1.2 Область застосування..... 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 8
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 12
2.3 Розгорнута постановка завдання 17
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 19
3.1 Опис функціонування системи 19
3.2 Розробка структурної схеми..... 23
3.3 Розробка функціональної схеми 27
3.4 Розробка діаграми процесів..... 29
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 31
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 31
4.2 Захист розробленого програмного забезпечення..... 44
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 50
6 ОСНОВНІ ВИСНОВКИ..... 56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58

КОПІЯ 2025

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АРП	–	автоматичне регулювання посилення
АТМ	–	автоматичний телевізійний моніторинг
ЕПТ	–	електронно-променева трубка
ІБ	–	інформаційна безпека
ІС	–	інформаційна система
НСД	–	несанкціонований доступ
ПЗ	–	програмне забезпечення
ПЗЗ	–	прилади із зарядовим зв'язком
СКД	–	системи контролю даних
СЦК	–	система централізованого керування
ТВЛ	–	телевізійні лінії
IP	–	Internet Protocol. Протокол міжмережної взаємодії
PIN-код	–	персональний ідентифікаційний код
TCP	–	Transmission Control Protocol. Протокол керування передачею (даних) Основний транспортний протокол у стеці протоколів TCP/IP. Встановлює зв'язок між двома ПК й організує обмін даними в дуплексному режимі
TCP/IP	–	Transmission Control Protocol/Internet Protocol. Стек протоколів, що забезпечують організацію зв'язку між комп'ютерами в мережі Інтернет

КбПЗ-2025

ВСТУП

Актуальність теми. Необхідність подачі електроживлення й забезпечення високої пропускної здатності каналів, які підводяться до кожної камери при їхньому масовому розгортанні, приводить до росту витрат і неоптимального розміщенню камер.

Камери спостереження високої чіткості здобувають все більшу популярність як засіб забезпечення безпеки лікарень, університетських комплексів і інших об'єктів. У той же час еволюція технологій відеозйомки й жорсткість вимог з боку замовників змушують постачальників модернізувати й кабельні системи, використовувані для їхнього підключення.

Системи відеоспостереження стають усе більше затребуваними: по даним Transparency Market Research, до 2027 року обсяг продажів на цьому ринку досягне 40 млрд доларів, а згідно із прогнозами securitysales.com, уже до 2026 року оберт у цьому сегменті складе 42 млрд доларів, причому в період з 2013 по 2026 рік щорічні темпи росту будуть перебувати на рівні 24,2%.

Звичайно, публикуємі прогнози носять оцінний характер, але всі аналітики сходяться в тому, що інтерес до встаткування для організації відеоспостереження буде рости. Збільшенню попиту на системи відеоспостереження сприяють безліч факторів, у числі яких посилення терористичних погроз, необхідність запобігання правопорушень, розвиток Інтернету речей і нових додатків, наприклад комерційної реклами. Представте, що при вході в торговий центр система «довідається» вас і починає демонструвати контекстну рекламу, що відповідає вашим купівельним перевагам.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи гібридного кабельного відеонагляду з використанням технології PFCS.

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

КПЗ-2025

– Огляд існуючих систем гібридного кабельного відеонагляду з використанням технології PFCS.

– Дослідження системи гібридного кабельного відеонагляду з використанням технології PFCS.

– Програмна реалізація системи гібридного кабельного відеонагляду з використанням технології PFCS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі гібридного кабельного відеонагляду з використанням технології PFCS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи гібридного кабельного відеонагляду з використанням технології PFCS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КбПЗ_2025

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

На сьогоднішній день системи безпеки приватних будинків і квартир розділяють на кілька типів:

– охоронна сигналізація (з підключенням на пульт охорони або дзвінком на ваш мобільний телефон);

– відеоспостереження (можливо навіть через інтернет);

– домофон (обмеження доступу).

У цьому розділі зупинимося більш докладно на системах відеоспостереження, тому що вони користуються найбільшою популярністю. Система відеоспостереження складається з однієї або декількох відеокамер, відеореєстратора й монітора. Відеореєстратор дозволяє записувати кілька тижнів підряд зображення з камер спостереження, зберігати відеозапису в архіві й передавати зображення за допомогою мережі Інтернет у будь-яку точку миру. Камери відеоспостереження можуть установлюватися на вулиці або усередині приміщень, вони відрізняються конструктивними особливостями й мають різні характеристики, наприклад, можуть передавати або кольорове, або чорно-біле зображення.

Ви вирішили встановити відеоспостереження. Варто визначити необхідний рівень захисту. Для квартири це буде «відеовічок» і камера на сходовій площадці. Таким чином, ви одержите інформацію про всіх ваших відвідувачів і гостей. Для заміського будинку знадобиться спостерігати за периметром ділянки, воротами, хвірткою й будівлями на території. Після цього, для організації ефективної системи безпеки, знадобиться скласти план квартири, ділянки або будинку, на якому потрібно відзначити зони огляду камер. Це допоможе визначити необхідну кількість камер спостереження, їхнє оптимальне

КПЗ-2025

розташування, спосіб і схему прокладки кабелю від відеореєстратора до відеокамер.

Для ефективної роботи системи відеоспостереження необхідно кілька камер. Як мінімум, це камера спостереження за центральним входом (хвірткою), воротами для в'їзду на територію й гаражем. Камери, розташовані на кутах будинки, дозволять контролювати всі підходи до нього. Установивши спостереження за дитячим майданчиком або басейном, ви будете бачити, що відбувається з вашими дітьми.

Важливими характеристиками камер є:

– кут огляду камери;

– її ефективність у темний час доби.

Не варто заощаджувати на кількості камер, установивши одну камеру з більшим кутом огляду. Адже, чим ширше кут огляду, тим далі й менше буде на зображенні спостережуваний об'єкт. Тоді не можна буде розглянути особа людини або номер машини. Кількість камер і кут огляду повинні бути ретельно підібрані й обґрунтовані. А для охоронних функцій системи відеоспостереження необхідним, звичайно ж, є здатність камери працювати й передавати зображення вночі. Для надійної роботи відеоспостереження в темний час доби, краще заздалегідь потурбуватися про вуличне висвітлення на своїй ділянці. Камерам спостереження необхідно мінімум висвітлення. Існують камери, які працюють при повній темряві. У них використовуються світлочутлива матриця й інфрачервоне висвітлення. Але вартість таких камер на порядок вище звичайних.

КПЗ-2025

1.2 Область застосування

Часто буває так, що, розташовані на видному місці камери одним своїм видом відлякують зловмисника. Бачачи, що будинок або квартира захищені системою безпеки, правопорушник може відмовитися від своєї витівки проникнення в таке приміщення. Адже поруч велика кількість менш захищених

квартир і будинків. Дизайн приладів системи відеоспостереження оцінюють головним чином, виходячи з функціональності системи в цілому. При виборі камер відеоспостереження для установки усередині вдома або квартири часто виникає необхідність того, щоб камери вписувалися в дизайн інтер'єра приміщення. Виробники пропонують різні купольні камери й мініатюрні камери різного виконання й колірних гам для гармонійності з інтер'єром. При установці камер на вулиці, важливим критерієм вибору є здатність камери працювати 24 години на добу в сніг, у дощ і на сонце. Вона повинна бути захищена від атмосферних опадів і перепадів температури, недоступна для зловмисників, які захочуть нанести їй шкода або зірвати її.

Відеозображення з камер спостереження можна вивести на телевізор у будинку. Використовуючи різні розгалуження, можна підключити трохи телевізорів у різних кімнатах. Сучасне відеоспостереження дозволяє інтегрувати аналогові камери спостереження в цифрові системи відеоспостереження й передавати зображення по комп'ютерній мережі й переглядати відеоспостереження через Інтернет у будь-якій точці миру. На відміну від старих відеомагнітофонів, запис у відеореєстраторі ведеться на цифровий носій інформації – на жорсткий диск, флеш пам'ять, DVD диски.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи гібридного кабельного відеонагляду з використанням технології PFCS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КПЗ-2025

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

Говорять, що Україна – це країна втраченого потенціалу, що в справжнім положенні речей ми можемо тільки імпортувати різну складну електронну апаратуру, а самі нічого зробити не здатні. Але не отут-те було! Після розпаду СРСР пройшло досить часу, щоб налагодити високотехнологічне виробництво. Компанія “Страж” саме це й зробила. Можливо, не все ідеально, але системи відеоспостереження, які робить ця фірма, реально працюють, причому чітко й без збоїв. Ще одним плюсом виробника є ціна. Через відсутність усяких митних зборів коштують ці системи на парі десятків доларів дешевше, ніж їхні закордонні побратими. На кілька таких систем сьогодні ми й оборотний увага.

Серед усього модельного ряду, пропонованого компанією “Страж”, зараз ми подивимося на три моделі: : «Превент 4В+ (УЛ-420С-4)», «Контрол 4В+(УЛ-420С-4)» і «Контрол 1Ц+ (ЦЛ-480С-1)». У дужках модель камер і їхньої кількості. Цифра після назви означає кількість камер, що підключаються. Тобто, в “Превент 4В+” можна підключити 4 камери, які йдуть у комплекті, в “Контрол 1Ц”, можна підключити чотири, а в комплекті йде тільки одна. Є ще “Превент” і “Контрол 4В” без +, різниця лише в тому, що “плюс” означає поліпшену камеру. У цьому випадку замість моделі камери 420С у версії без плюса коштує 480К, і корпус другої камери не такий крутий і стильний.

Всі системи, крім 1Ц, можуть використовуватия як на вулиці, так і в приміщенні, вони здатні “бачити” навіть у темряві, а 1Ц оснащена всього однією простою камерою без нічного “зору”, може висіти тільки в приміщенні й не витримає мінусових температур. Так що за допомогою 1Ц простежити з

КПЗ-2025

морозилки за злодієм ваших пельменів не вийде. А якщо серйозно, те ці системи дійсно професійні й можуть використовуватися для різних серйозних завдань, де важлива стабільність роботи й надійність усього механізму.

Корпус камери зроблений повністю з металу. Зверху є козирок (видимий, для захисту від сонця й дощу). Пофарбовано кожну камеру в темно-сірий колір, щоб бути не дуже помітною на вулиці. Складання якісний і щільна, виробник обіцяє, що камери не виключаться при температурі -10 і +50°C и вологості 95%, але є одна проблемка: під час роботи камери злегка нагріваються. Не сказав би, що розжарюються, як поверхня Сонця, вони просто тепленькі. Місця з'єднання комутації не занадто ізольовані, так що з такою системою для більшої надійності прикуповуйте моток ізоленти або термоусадки (для правильних радіотехніків).

Стекло, що захищає лінзу, досить товсте й міцне, так що проблем з ним не буде.

У камерах, у яких є ІЧ-підсвічування, самі ІЧ-світильники перебувають під склом у самому корпусі камери й уночі злегка світяться.

З камери виходить два дроти: сигнальний і силовий: по одному йде картинка, а по-іншому надходить живлення в камеру. Дроти, що веде від камери до декодера, спарений, тобто дроти один, а коннектора – чотири. Для підключення камер до мережі в коробці лежить блок живлення з безліччю виходів. Сигнальний кабель включається безпосередньо в декодер.

Ще в комплекті є набори для кріплення камер: обертова ніжка й набір саморезів і дюбелів. Прикріпити камеру до стіни або до стелі не складе проблем, якщо ви вмієте працювати дрилем. Ще внизу корпуса камери є отвір з універсальним різьбленням 1/5” – це те ж кріплення, що й на “черевику” стандартного штатива.

У цій маленькій коробочці для роботи на вас укладені в різних системах 3 види камер: 420С, 480К, 480С.

В 420С установлена ⅓” матриця виробництва Sony з розрішення 420 ТВЛ (телевізійних ліній), що приблизно відповідає розрішенню 760 х 420 пікселів, і

КОПЗ-2025

знімає вона в кольорі, що добре для камер спостереження. Фокусна відстань (ФВ) – 3,6 мм, що дає кут огляду 53 градуса обрій /41 градус вертикаль. У цій камері 14 ІЧ-діодів.

В 480К коштує ?" CMOS-матриця з розрішення 480 ТВЛ, або 853 х 480 пікселів. Теж кольорова картинка, ФР 6 мм і кут огляду 67 градуса обрій /53 градус вертикаль. Отут коштує 24 ІЧ-діода.

В 480С всі те ж, що й в 480К, тільки матриця й процесор Sony. І висіти вона може тільки в приміщенні, у ніжних і умовах, що щадять. Без інфрачервоного підсвічування.

В цих камерах (крім 480С) є режим нічного бачення, що дозволяє стежити навіть за тими речами, яких звичайне людське око не побачить. Камера сама перемкнеться, коли зрозуміє, що висвітлення недостатньо. Продуктивна відстань підсвічування – 15 м. Якщо таке поставити в себе у дворі, я думаю, підсвічування вистачить.

Усе, що знімають камери, надходить у відео-реєстратор (він лежить у тій же коробці й, по суті, є серцем всієї системи).

Разом із цим "чорним ящиком" у комплект входить пульт ДУ, блок живлення й мишка "Спайглас ультра киборг фантом инно 8д рейзер х15". Насправді, просто USB-мишка.

Відеореєстраторів теж є два види. В "Превент" маленький білий пластиковий корпус, у комплект із яким не входить мишка, і запис відео йде на флешку до 32 ГБ. Попереду – 3 індикатори роботи, а позаду – 4 «тюльпанових» входи для відеокамер, 2 аудіовхода й, природно, відео- і аудіо виходи. Ще на задній стороні є Ethernet-вхід і VGA-вихід на монітор.

У системі "Контрол" все небагато по-іншому – корпус розміром з DVD-програвач чорного кольору зроблений з металу, а передня панель – пластикова. На ній перебувають основні кнопки керування відеоспостереженням і світлодіодні індикатори роботи.

КПЗ_2025

Позаду можна застромити сигнальні дроти від камер, дроти від мікрофона або двох, відразу – вхід для кабелю живлення, VGA-вхід (або іншими словами для тру – D-SUB), щоб підключати монітор, два входи USB 2.0 і Ethernet-вхід. У USB можна підключити мишку (ту саму, з набору), трекпед або флешку для бекапа відеозаписів. До речі, основний запис відбувається на жорсткий диск, що вставляється усередину тіла самого реєстратора, і для цього прийде його розкрутити. Виробник заявляє, що HDD можна вставляти до 2 ТБ, але поставив на 4, і знаєте, що? Усе працювало. Запис іде у форматі H.264, так що проблем із читанням відео на комп'ютері бути не повинне.

Працює все це на Linux. Виробникові окреме спасибі, що не стали винаходити велосипед. Інтерфейс дуже прост і зрозумілий, управляти можна пультом, мишкою або кнопками на корпусі. У меню можна настроїти програму дій при закритті камери, при русі камери або заданої зони. Система може відправляти вам e-mail / видавати звук або робити різні інші штуки.

Є цікава функція – ви можете спостерігати за об'єктом вашого спостереження віддаленно, через мобільні пристрої або через простий браузер, при цьому вам не обов'язково мати статичний IP системи спостереження. У смартфоні або планшеті вам потрібно встановити додаток, що є в AppStore, GooglePlay, MarketPlay (коротше, на все, що тільки завгодно, можна поставити цей додаток), після цього підключаєтеся до вашої системи – і дивитеся за всім, звідки захочете (там, де Інтернет дозволяє). До речі, можна набудовувати якість стриммінга в мережу, розрішення і кількість кадрів, якщо хочете дивитися по HD.

Програма зберігає всі логи, є система користувачів, що може обмежувати доступ до різних функцій певного кола осіб.

До речі, всю роботу спостереження можна настроїти на певний календарний графік аж до хвилини – працює все як годинники.

У режимі відтворення на таймлайні буде червоним кольором виділена область, на якій замічене рух у кадрі, при цьому можна дивитися синхронний запис із всіх камер одночасно. Система крута сама по собі й ще тому, що будить

КПЗ-2025

патріотичні почуття. Ціни коливаються від 925 (за найпростішу систему) до 4800(за систему з більшим функціоналом) гривень.

Рисунок 2.1 – Інтерфейс користувача

Зроблена із совістю, зовні, може бути, не Apple, але ця штука виконує своє завдання на 101%, і в її надійності можна не сумніватися. Тепер ваш будинок, офіс, дача або холодильник будуть під доглядом.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктноорієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

КБПЗ_2025

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктноорієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веброзробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв’язанні проблеми. Їм не потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

КПЗ – 2025

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

КОПЗ – 2025

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов’язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик

КПЗ–2025

виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати, перетворювати, розділяти, об’єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

КПЗ-2025

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веброзробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

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 гібридного кабельного відеонагляду з використанням технології PFCS.

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;

КПЗ – 2025

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.

КбПЗ_2025

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Відеоспостереження будинку дозволить контролювати роботу обслуговуючого персоналу, будівельників, допоможе забезпечити схоронність майна співробітникам охорони, доглядати за дитиною, спостерігати за всім, що відбувається з будь-якого комп'ютера який має доступ в інтернет.

Завдання системи відеоспостереження:

– Контроль поводження обслуговуючого персоналу.

– Спостереження за дитиною.

– Спостереження за периметром ділянки.

– Архівування відеозображення з камер.

– Відеоспостереження через інтернет.

Система відеоспостереження для будинку повинна забезпечити відеоспостереження за ділянкою, контроль внутрішнього двору, і візуальний контроль за внутрішніми приміщеннями котеджу. Так само система допоможе спостерігати за маленькою дитиною яка знаходиться в іншій кімнаті або залишена одна дома. Крім того, компоненти системи відео спостереження повинні гармонійно вписуватися в інтер'єр приміщень.

Важливою вимогою відеоспостереження для будинку, є забезпечення зберігання архівної відеоінформації протягом не менш 2-х тижнів (найпоширеніша тривалість від'їзду на відпочинок). Природно, власник будинку повинен мати можливість перегляду обстановки в будинку з будь-якої точки земної кулі (за умови наявності в будинку виходу в мережу Інтернет. Передбачено надійний захист передачі, тому ніхто крім вас не зможе переглядати відеоспостереження.

КОПІЯ 2025

На ключових місцях установлюються відеокамери вуличного й кімнатного виконання. Відеокамери вуличного виконання допускають експлуатацію при негативних температурах і мають герметичний корпус. Удень відеоспостереження в будинку дозволяє бути в курсі того, що роблять діти, що грають на ділянці, яка машина під'їхала до в'їзних воріт, але більшість крадіжок відбувається вночі, тому необхідно, щоб камери могли передавати якісне зображення як уночі, так і вдень. Виду того, що в темний час доби складно забезпечити достатній рівень освітленості, краще використовувати чорно-білі або кольорові камери з автоматичним перемиканням день-ніч. Такі камери мають більше високу світлочутливість і не вимагають додаткового висвітлення – звичайний або інфрачервоний прожектори. Вуличні камери оснащені убудованим підігрівом і не бояться навіть найдужчих морозів. Для відеоспостереження усередині приміщення краще використовувати більше інформативні кольорові камери купольного виконання, які достатньо гармонійно вписуються в будь-який інтер'єр.

Запис зображення виробляється на жорсткий диск цифрового відеореєстратора за розкладом, за детекцією руху, або в постійному режимі. Вихід USB і убудований DVD-RW дають можливість швидкого копіювання тимчасового проміжку відеоархіву, що цікавить Вас, на флешку йди диск для перегляду поза будинком або для надання доказів правоохоронним органам. Українськомовне, зручне меню допоможе без проблем освоїти керування системою відеоспостереження для будинку.

Готове рішення відеоспостереження для квартири

Система відеоспостереження для квартири дозволить контролювати роботу обслуговуючого персоналу, будівельників відсутності, що перебуває у квартирі під час, хазяїв, доглядати за маленькою дитиною яка знаходиться в іншій кімнаті або залишеним одного будинку, спостерігати за всім, що відбувається з будь-якого комп'ютера який має доступ через інтернет (для системи з відеореєстратором працюючих у мережі Internet).

КПЗ - 2025

Завдання системи відеоспостереження для квартири:

– Контроль поводження обслуговуючого персоналу.

– Спостереження за дитиною.

– Спостереження за сходовим прольотом.

– Спостереження за припаркованим автомобілем.

– Архівування відеозображення з камер спостереження.

– Передача відеоспостереження через інтернет для перегляду з будь-якої точки світу.

Система відеоспостереження повинна забезпечити відеоспостереження у квартирі, контроль сходового прольоту, і, як варіант, відеоспостереження за припаркованим автомобілем. Камери відеоспостереження системи крім виконання своїх основний функцій, повинні гармонійно вписуватися в інтер'єр квартири. У даному комлекті поставляються відеокамери в купольному виконанні, які досить непомітні й можуть кріпитися як на стелю так і на стіни. Поставляється в комплекті LCD монітор може бути встановлений поруч із вхідними дверима за допомогою настінних кріплень, при такій установці буде зручно дивитися зображення передане відеовічком. Відеовічко має широкий кут огляду що дозволить спостерігати за всім що відбувається за вашими дверима й Ви будете бачити особу Вашого гостя, навіть якщо він відійде від дверей на достатню відстань. За допомогою відеовічка буде записуватися якісне відеозображення всіх вхідних до Вас у квартиру або людей, що наближаються до вхідних дверей.

Важливою вимогою відеоспостереження за квартирою, є забезпечення зберігання архівної відеоінформації протягом не менш 2-х тижнів (найпоширеніша тривалість від'їзду на відпочинок). Природно, хазяїн квартири повинен мати можливість перегляду обстановки в будинку з будь-якої точки земної кулі онлайн (за умови наявності в будинку виходу в глобальну мережу Інтернет. Ви зможете дивитися що відбувається в себе у квартирі будучи на роботі або у відпустці. Передбачено надійний захист передачі, тому ніхто крім

КПЗ-2025

вас не зможе переглядати відеозображення. Відеореєстратор записує на жорсткий диск зображення з камер спостереження за розкладом, за детекцією руху, або в постійному режимі. Вихід USB дає можливість швидкого копіювання тимчасового проміжку відеоархіву, що цікавить Вас, на flash карту для перегляду поза будинком або для надання доказів правоохоронним органам. Українськомовне, зручне меню допоможе без проблем освоїти керування системою відеоспостереження.

Завдання системи відеоспостереження:

– Візуальний контроль освітленості під'їздів і дворських територій у темний час доби. Контроль якості й своєчасності збирання територій, вивозу сміття й т.п.

– Цілодобова візуальна оцінка обстановки в житловому секторі шляхом телевізійного спостереження за під'їздами й дворськими територіями, а також, при необхідності, входами в технічні приміщення житлових будинків.

– Оперативне оповіщення служб охорони правопорядку й інших екстрених служб міста про виникнення або підозру на виникнення ситуацій, що загрожують життя й здоров'ю людей, схоронності їхнього майна, а також схоронності муніципального майна.

– Надання службам охорони правопорядку й інших зацікавлених служб міста архівної інформації телевізійного спостереження для відновлення ходу подій, підтримки дроти оперативно-слідчих заходів і т.п.

Як правило всі сучасні житлові комплекси мають цілодобову охорону (консьєржа). У допомогу йому, для забезпечення максимальної безпеки й спокою на території всього житлового комплексу ми пропонуємо свої сучасні технічні рішення.

Всі входи й в'їзди в житловий сектор, автопаркінг, під'їзди, ігрові площадки повинні обов'язково попадати під об'єктиви відеокамер. При недостатнім висвітленні для одержання якісного зображення встановлюються ІЧ підсвічування або ІЧ прожектор.

КПЗ-2025

У даному технічному рішенні представлені відеокамери високого розрішенню в термокожухах з кутами огляду до 90 градусів. Високе розрішення дає можливість більш точно визначити особистість правопорушника. Широкий кут огляду дозволить охопити більшу територію житлового комплексу, зменшивши площу мертвих зон і кількість відеокамер, скоротивши тим самим витрати на систему в цілому. При необхідності на під'їзди, а також на сходові прольоти встановлюються вандалозахищені відеокамери. Виключено доступ небажаних сторонніх осіб на територію двору.

3.2 Розробка структурної схеми

У минулому досить широке поширення одержали аналогові камери відеоспостереження з низьким розрішення – 920Н і нижче. Однак по якості їх навіть не можна порівнювати з новим поколінням відеокамер високої чіткості, що підключаються до мереж IP.

Відеокамери високої чіткості підтримують розрішення 720p (720 вертикальних ліній) або 1080p. Чим вище розрішення у камер спостереження, тим краще: стають ясно помітні невеликі об'єкти, особи людей, номерні знаки на автомобілях та інші елементи. З переходом на стандарти 4K з'явиться можливість підтримувати розрішення 3840 пікселів на 2160 ліній, або 2160p, а виробники вже проводять експерименти з відео 8K, коли розрішення збільшується ще у два рази.

Камери 4K уже посприяли значному вдосконаленню систем безпеки. Їх можна розміщати на більшому видаленні від контрольованої зони, при цьому цифрове збільшення забезпечує більше детальний розгляд невеликих об'єктів. В одному з тестів за допомогою відеокамери, розташованої в 3 км від центра міста, і технології цифрового зума вдалося успішно зчитувати номерні знаки автомобілів.

Безпека підвищується й за рахунок застосування недорогих серверів, що обробляють потоки відео для розпізнавання осіб.

КПЗ - 2025

У міру збільшення розрішенню камер росте потреба в підвищенні пропускної здатності каналів зв'язку. Колишні пристрої можна було підключати по коаксіальному кабелі, оскільки високої пропускної здатності не було потрібно. За деякими оцінками, для однієї відеокамери 4K, що використовує сучасні алгоритми стиску відео, потрібно не менш 15 Мбіт/с (без стиску ще більше). А коли територія комплексу будинків обладнається цілою системою таких камер, потреба в пропускній здатності помітно зростає. У звіті «The Video Surveillance Report, 2010», підготовленому ABI Research, затверджується, що в середньому по розмірах казино Лас-Вегаса встановлено більше 2 тис. камер. Саме по собі це ще не вимагає переходу на волоконно-оптичний кабель, але у випадку збільшення відстаней традиційного мідного виявляється недостатньо.

Крім каналів з високою пропускною здатністю, камерам відеоспостереження потрібні джерела живлення, підключення яких може виявитися досить проблематичним. На території студентських городків, наприклад, камери часто розміщають на ліхтарних стовпах, але в денні годинники електроенергія на ці стовпи подається рідко (відключення ініціюється фотоелементами), так що забезпечити пристрою живленням протягом 24 год у добу не вдасться. В інших випадках, коли камери кріпляться до стін або встановлюються на електричних стовпах, подачу електроживлення на них доводиться погоджувати із власником будинку або комунальною службою й щомісяця оплачувати рахунку.

Якщо приймається рішення про підключення до локального джерела енергії, робота камер буде залежати від його надійності. Щоб запобігти відключенням, мережні оператори іноді встановлюють ІБЖ із резервними батареями, але обслуговувати такі батареї й складно, і дорого.

Інститут інженерів по електротехніці й електроніці (IEEE) затвердив стандарт для передачі даних і живлення по загальному кабелі Ethernet. Технологія зветься Power over Ethernet (PoS). Стандарти 802.3af (PoS) і 802.3at (PoS Plus) визначають спосіб доставки даних і живлення на відстань до 100 м по

КПІ-2025

категорійному кабелі. Стандарт PoS передбачає набагато більше, ніж просту передачу постійного струму: він описує необхідні функції безпеки, програмне узгодження параметрів при встановленні зв'язку й багато чого іншого.

Розміщення камер відеоспостереження на всій території комплексу будинків і інших великих об'єктів припускає їхня далекість на сотні метрів від центра контролю. За підтримкою великої дальності передачі можна скоротити число таких центрів і тим самим зменшити капітальні й операційні витрати. У таких випадках при організації з'єднань перевага найчастіше віддається волоконно-оптичному кабелю.

Одномодове волокно дозволяє передавати фактично необмежені потоки даних, і його можливостей виявляється цілком достатньо для підключення камер з розрішення 4K і 8K, а довжина кабелю може досягати декількох кілометрів. Багатомодове волокно (особливо більше нові 50-мікронні волокна OM3 і OM4) забезпечує передачу на сотні метрів, і в багатьох ситуаціях теж виявляється цілком життєздатним рішенням.

Отже, волоконно-оптичний кабель – відмінний варіант для передачі даних на більші відстані, але як у цьому випадку забезпечити енергопостачання?

Оптична кабельна система з підтримкою живлення (Powered Fiber Cable System, PFCS) дозволяє організувати як подачу живлення, так і передачу даних. PFCS містить у собі встановлюваний у стійку прикінцевий блок для оптичного волокна й живлення, гібридний кабель із одномодовим або багатомодовими волокнами, а також електричними провідниками й віддалений вузол, до якого підключається камера.

У системах PFCS кожний кабель здатний забезпечити енергопостачання пристроїв PoS на видаленні до 3 км, що в 30 разів перевищує можливості стандартних систем PoS. Комбінація інтегрованого конвертера між середовищами передачі з набором мікросхем PoS забезпечує подачу сигналів по стандартах PoS і PoS Plus через рознімання RJ45. Система забезпечує трьохрівневий електричний захист при роботі як у твердих зовнішніх умовах, так

КОПЗ-2025

і усередині приміщень, а застосовувана технологія перетворення постійного струму дозволяє обійтися без розрахунку потужності ліній постійного струму.

Застосування PFCS дозволяє істотно скоротити як вартість, так і час розгортання завдяки поверненню контролю за енергопостачанням операторові мережі. Кабельні системи такого роду спрощують технічні виміри й узгодження при організації енергопостачання, а також скорочують розмір рахунків, що виставляються щомісяця, за електроенергію, спожиту на віддалених площадках. Процес розгортання системи оптимізується за рахунок розміщення в одному місці встаткування для енергопостачання й передачі даних.

Розгортання камер спостереження з PFCS

При використанні системи PFCS процедура розгортання відеокамер спрощується. PFCS застосовується як свій рід подовжувача, що доставляє дані й електроенергію в будь-яке потрібне місце. Таким чином, камери розміщаються там, де необхідно, а не там, де є електрика. Крім того, PFCS відповідає вимогам National Electrical Code (NEC) Class 2 і Safety Extra Low Voltage (SELV) до малопотужних й електрично безпечних систем, а тому залучати кваліфікованих електриків не потрібно. Кабелі можна прокладати по тимі ж трасам, що й проводку Ethernet або звичайні волоконно-оптичні кабелі.

Крім того, PFCS дозволяє заощадити на прокладці окремих ліній для передачі даних і енергопостачання. Приміром, на одній із площадок для гри в гольф у Південній Флориді проект по розгортанню 36 камер відеоспостереження з використанням окремих електричних і волоконно-оптичних кабелів оцінювався в 981 тис. доларів, а підключення їх за допомогою системи PFCS дозволило скоротити витрати до 839 тис. дол., забезпечивши економію в 142 тис. доларів, або 14,5%.

При переході до камер з розрішення 4K і вище використання систем PFCS дозволить істотно скоротити строки й вартість процедури розгортання.

КОПЗ – 2025

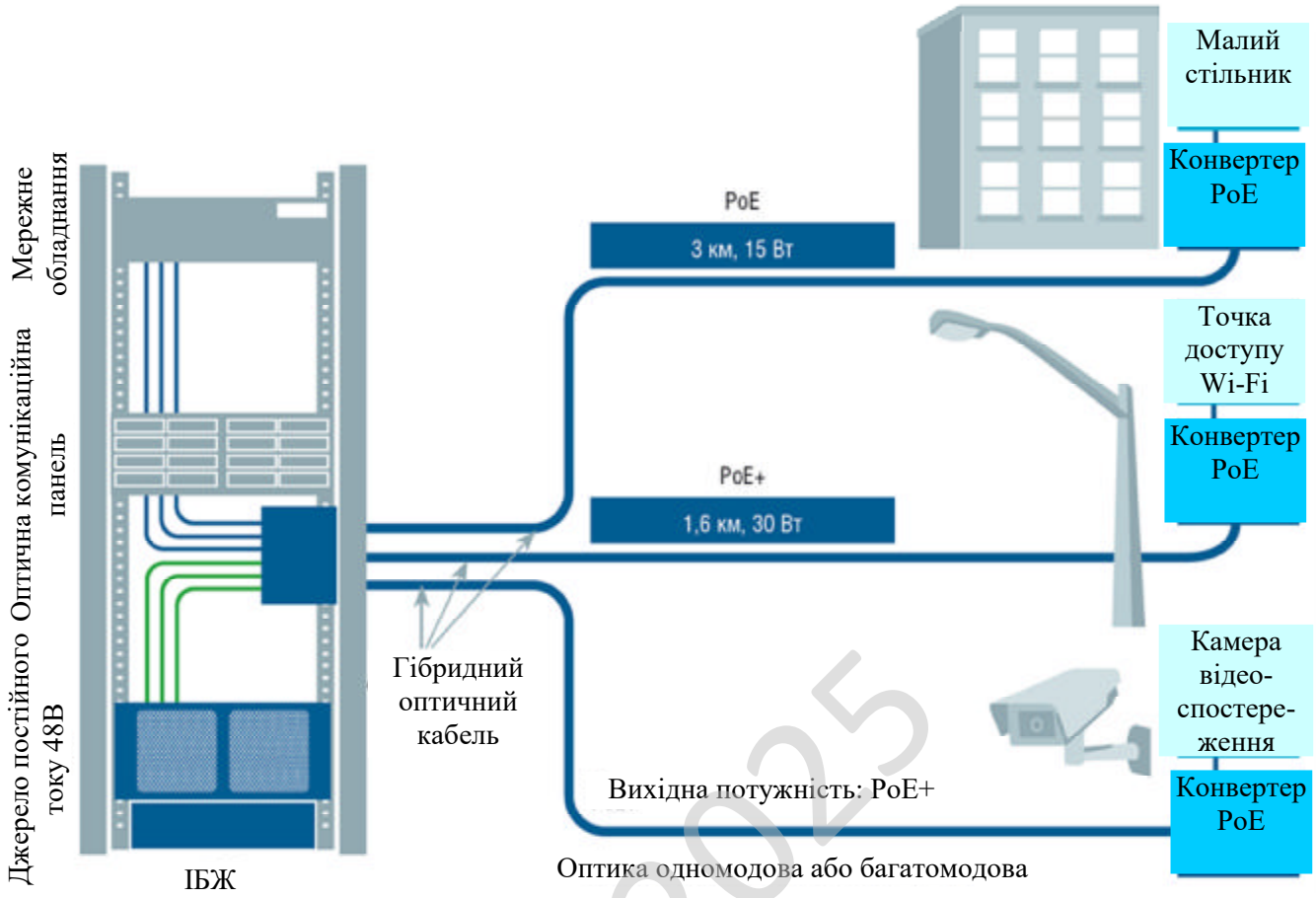

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Найпершим блоком у схемі є кінцевий об'єкт (сектор) за яким власне і ведеться відеоспостереження. Далі за допомогою відеокамер спостереження фірми DLink за розробленою схемою (рисунок 3.1) і застосуванні локального обладнання фірми Dlink (PFCS) сигнал надходить у кімнату операторів де обробляється на сервері за допомогою розробленого ПЗ.

Спочатку проходить обробка PFCS даних з послідуючим аналізом. Всі дані зберігаються у файловому архіві у форматі *.avi. Кожну добу інформація з камер спостереження формується у відповідну папку (шаблон папки ЧИСЛО_МІСЯЦЬ_РІК). Для зміни налаштувань чи перегляду документації

КбПЗ – 2025

існують відповідні вікна.

Також для організації маштабуємості системи й контролю дій над співробітниками сигнал також через мережне обладнання фірми Dlink (PFCS) надходить до керівництва. Розробляючи апаратну й програмну частину дипломного проектування враховувався ціновий фактор, а також особливості застосування камер в умовах України, а саме – застосовувався ударостійкий корпус, були обрані камери з розширеними температурними діапазонами роботи (-40 С до +55 С) і великим кутом огляду камери.

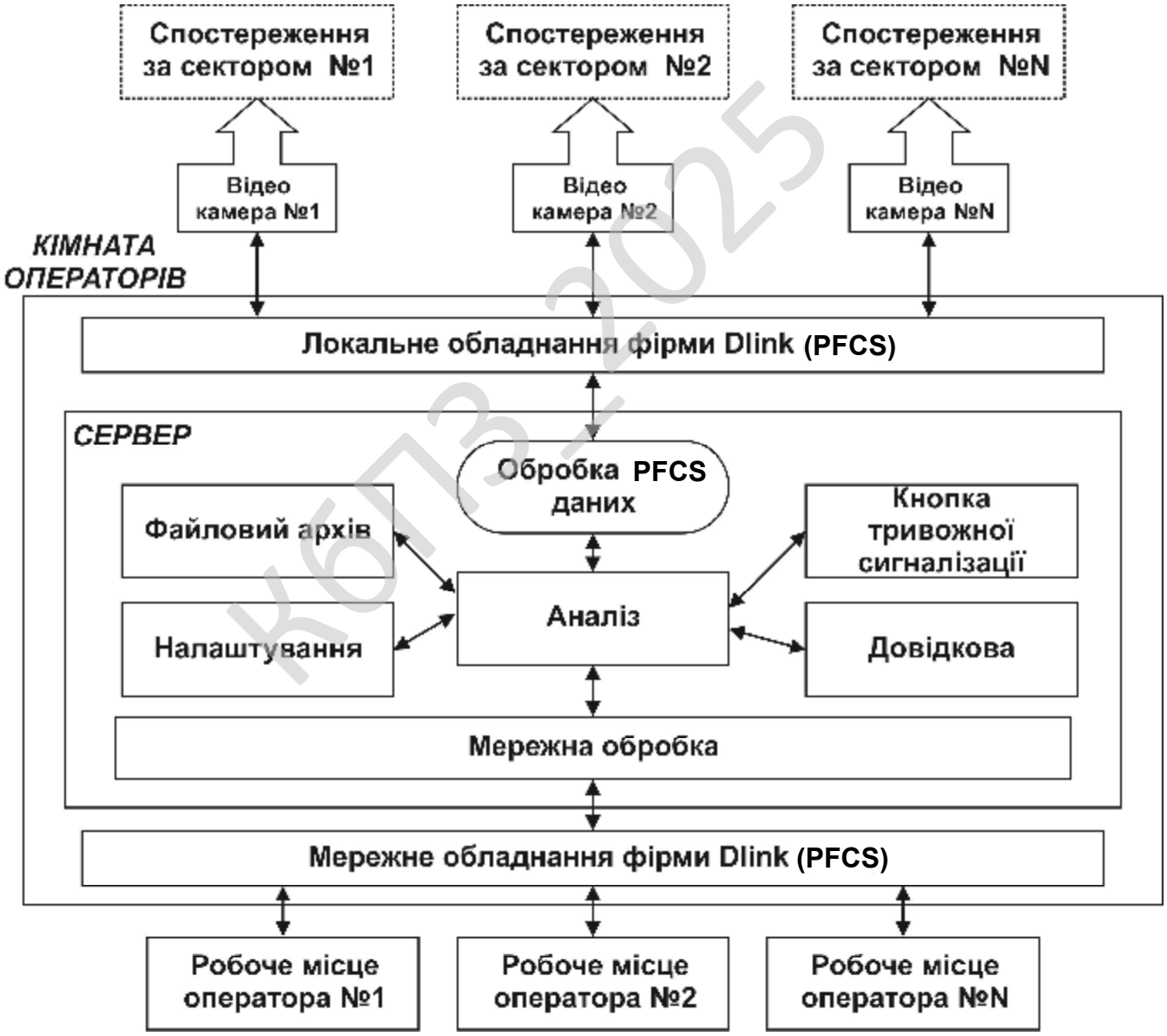

Рисунок 3.2 – Функціональна схема системи

КБПЗ-2025

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

3.4 Розробка діаграми процесів

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.

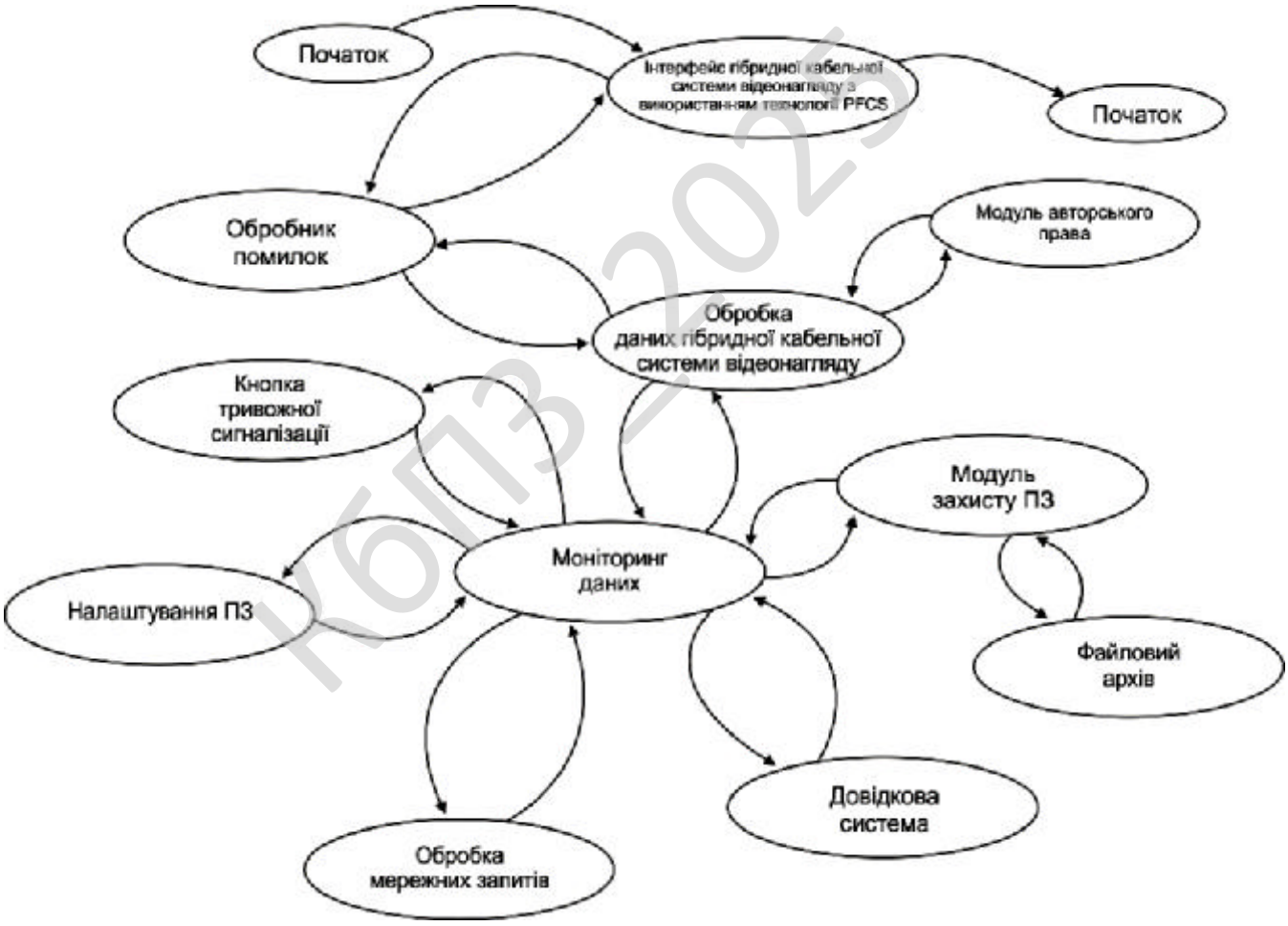

Рисунок 3.3 – Діаграма взаємодії процесів

КБПЗ 2025

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

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.

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.

КбПЗ_2025

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки 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, також те, що при розробці програми слід надати особливу увагу модулю гібридної кабельної системи відеонагляду з використанням технології PFCS.

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

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

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 – перевірки поточного стану та поверненням на початок схеми чи з завершенням роботи розробленого ПЗ.

КОПЗ-2025

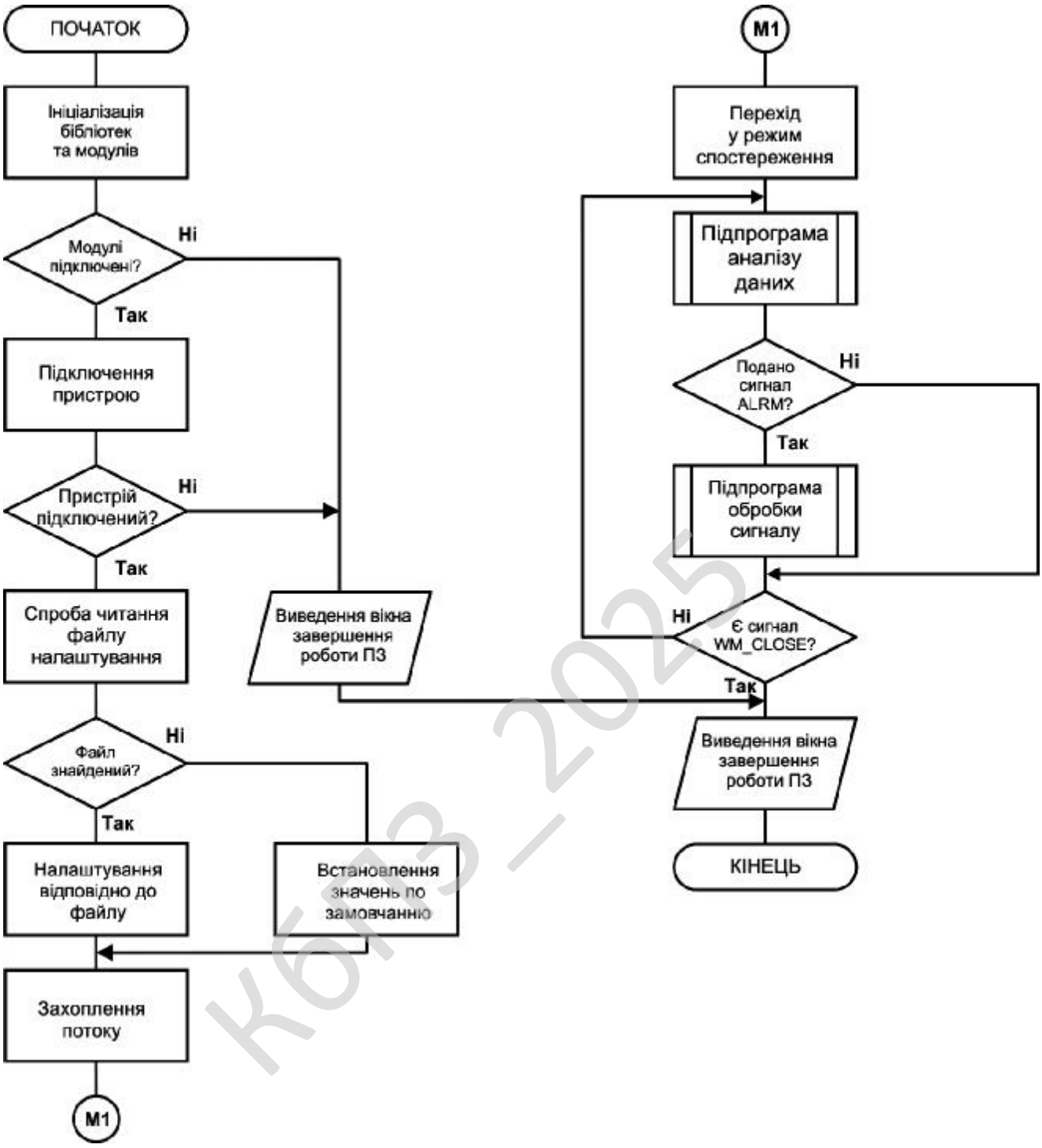

Рисунок 4.1 – Блок-схема основної програми

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.

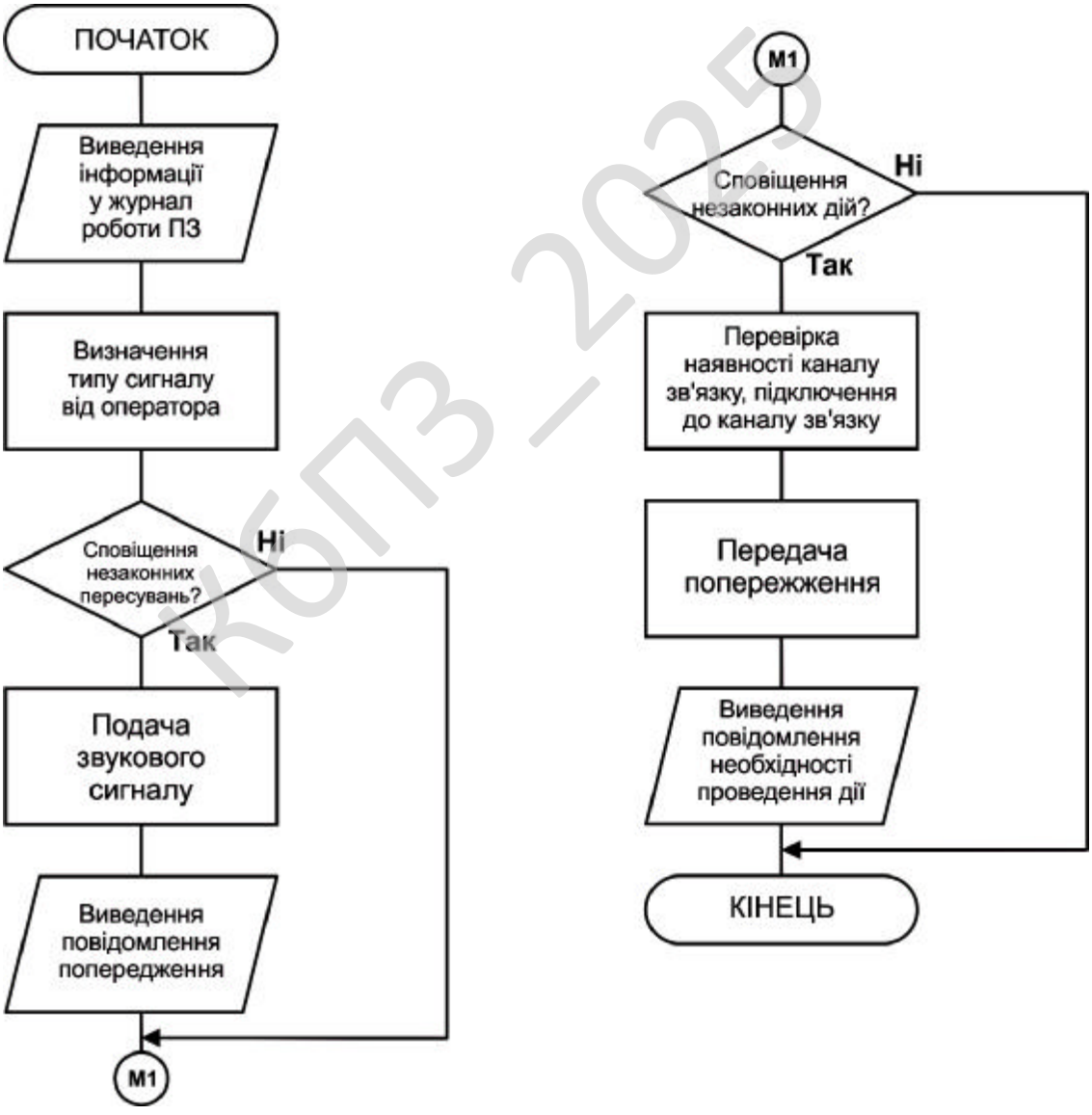

Рисунок 4.2 – Блок-схема роботи підпрограми

КОПЗ_2025

Опис системи гібридного кабельного відеонагляду

Система гібридного кабельного відеонагляду з технологією PFCS забезпечує надійний контроль та моніторинг об'єктів у режимі реального часу.

Архітектура системи включає апаратну та програмну складові, що взаємодіють між собою через мережеві та кабельні з'єднання.

Основні компоненти системи включають відеокамери, сервер обробки, сховище даних та програмне забезпечення, яке реалізує алгоритми аналізу та управління відеопотоком.

Архітектура системи

Система складається з кількох функціональних блоків. Відеокамери підключаються до серверного обладнання через кабельні або бездротові з'єднання.

Сервер обробки приймає відеопотік, здійснює його обробку та передає в систему зберігання даних. Програмне забезпечення реалізує алгоритми компресії, розпізнавання об'єктів та передачі даних користувачам.

PFCS (Packet Flow Control System) використовується для управління потоками даних, забезпечуючи мінімізацію затримок при передачі відео. Завдяки цій технології система підтримує адаптивне керування потоком, що дозволяє зменшити навантаження на мережеві з'єднання та підвищити якість переданого відео.

Основні програмні модулі

Програмне забезпечення системи розробляється мовою Python. Основні модулі включають:

– Модуль захоплення відео отримує потік з камер та передає його на обробку.

– Модуль аналізу відеопотоку виконує розпізнавання об'єктів та їх класифікацію

– Модуль управління потоками PFCS реалізує алгоритми адаптивного управління передачею відеоданих.

КОПЗ-2025

– Модуль збереження та архівування забезпечує запис та відтворення відеозаписів.

– Модуль інтерфейсу користувача забезпечує віддалене спостереження та керування системою.

Опис функцій та фрагментів коду

Програмне забезпечення містить функції для взаємодії з відеокамерами, обробки потоків даних та аналізу відео.

```
import cv2
import numpy as np

def capture_video(source):
camera = cv2.VideoCapture(source)
while camera.isOpened():
ret, frame = camera.read()
if not ret:
break
yield frame
camera.release()

def process_video(frame):
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150)
return edges

def save_video(frames, output_path):
height, width = frames[0].shape
video_writer = cv2.VideoWriter(output_path,
cv2.VideoWriter_fourcc(*'XVID'), 20, (width, height))
for frame in frames:
video_writer.write(frame)
video_writer.release()
```

Функція capture_video отримує відеопотік з камери та передає кожен кадр на обробку. process_video виконує перетворення кадру у чорно-білий формат та виявляє контури об'єктів. save_video записує оброблене відео у файл.

КбПЗ_2025

Зчитування відеопотоку, обробку кадрів, адаптивне керування потоком даних за допомогою PFCS.

```
import cv2
import numpy as np
import threading
import queue
import time
class VideoStream:
    def __init__(self, source=0):
        self.camera = cv2.VideoCapture(source)
        self.queue = queue.Queue(maxsize=10)
        self.running = True
        self.thread = threading.Thread(target=self._capture_frames,
                      daemon=True)
        self.thread.start()
    def _capture_frames(self):
        while self.running:
            ret, frame = self.camera.read()
            if ret:
                if not self.queue.full():
                    self.queue.put(frame)
            time.sleep(0.02)
    def get_frame(self):
        if not self.queue.empty():
            return self.queue.get()
        return None
    def release(self):
        self.running = False
        self.thread.join()
        self.camera.release()
def process_frame(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    return edges

class PFCS:
    def __init__(self, threshold=100):
        self.threshold = threshold
    def adapt_stream(self, frame_rate, network_load):
        if network_load > self.threshold:
            return max(5, frame_rate - 5)
```

КОПЗ_2025

```
            return min(30, frame_rate + 5)
    def save_video(frames, output_path, fps=20):
        if not frames:
            return
        height, width = frames[0].shape[:2]
        video_writer = cv2.VideoWriter(output_path,
                       cv2.VideoWriter_fourcc(*'XVID'), fps, (width, height))
        for frame in frames:
            video_writer.write(cv2.cvtColor(frame, cv2.COLOR_GRAY2BGR))
        video_writer.release()
    def main():
        stream = VideoStream(0)
        pfcs = PFCS(threshold=80)
        frames = []
        frame_rate = 20
        network_load = 50
        while True:
            frame = stream.get_frame()
            if frame is None:
                continue
            processed_frame = process_frame(frame)
            frames.append(processed_frame)
            network_load = np.random.randint(50, 120)
            frame_rate = pfcs.adapt_stream(frame_rate, network_load)
            cv2.imshow("Processed Video", processed_frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        save_video(frames, "output.avi", fps=frame_rate)
        stream.release()
        cv2.destroyAllWindows()
    if __name__ == "__main__":
        main()
```

КОПЗ_2025

Розрахунки та обґрунтування проектних рішень

При проектуванні системи враховується кілька параметрів. Основні розрахунки включають:

1. Пропускна здатність мережі залежить від роздільної здатності відеопотоку та кількості камер. Наприклад, камера 1080p при 30 кадрах за секунду потребує приблизно 4-6 Мбіт/с.

2. Об'єм сховища визначається тривалістю збереження відео. Для 10 камер із записом 24/7 у форматі H.264 потрібно близько 2 ТБ дискового простору на тиждень.

3. Затримка передачі відеопотоку мінімізується за рахунок використання PFCS. Випробування показують, що середня затримка не перевищує 200 мс.

Гібридна архітектура дозволяє інтегрувати різні типи відеокамер, оптимізувати використання ресурсів та забезпечити стабільну роботу системи. PFCS допомагає контролювати навантаження на мережу, що підвищує ефективність роботи відеоспостереження.

Jira – Комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі-системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – Фотографіями, скриншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між

КПЗ – 2025

станами можливі, визначається через настроюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи російську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи – плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько $50 за користувача для локального і $7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

КПЗ-2025

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

– проект використовує ліцензії, схвалені Open Source Initiative;

– Вихідний код проекту доступний для скачування;

– у проекту є публічно доступна веб-сайт;

– програмне забезпечення від Atlassian є на веб-сайті проекту.

Управління вимогами це процес запису, аналізу, трасування, пріоритезації і узгодження вимог та контролю змін і доведення до їх зацікавлених сторін. Це безперервний процес протягом всього життя проекту. Вимога – якість, якій мають відповідати результати проекту (продукту або послуги).

Мета управління вимогами полягає в тому, щоб переконатися, що організація відповідає потребам і очікуванням своїх клієнтів, внутрішніх або зовнішніх зацікавлених сторін. Управління вимогами починається з аналізу і виявлення цілей і обмежень організації. Управління вимогами додатково включає в себе підтримку планування вимог, інтеграції вимог і організації роботи з ними (атрибути для вимог).

Управління вимогами передбачає спілкування між членами проектної групи і зацікавленими сторонами, і адаптацію до змін у вимогах протягом всього проекту. Щоб запобігти перетину поля одного класу вимог з іншим, постійні зв'язки між членами команди розробників є критичними. Наприклад, при розробці програмного забезпечення для внутрішнього використання у бізнесу можуть бути настільки сильні потреби, що він може проігнорувати вимоги користувачів, або вважати, що створені сценарії використання покриють також і користувальницькі вимоги.

Відслідковування вимоги фактично означає документування всього життєвого циклу вимоги. Часто необхідно дізнатися першоджерело кожної вимоги. Для цього всі зміни вимог повинні бути задокументовані, щоб досягти стану повного відстеження. Відстежувати треба бути навіть використання реалізованих вимог.

КПЗ – 2025

Вимоги мають різні джерела, такі як ділова людина, що замовляє продукт, менеджер зі збуту і фактичний користувач. У всіх цих людей є різні вимоги до продукту. Використовуючи відслідковування вимог, реалізована в системі функція може бути простежена назад до людини або групі, яка замовляла її під час збору вимог. Ця особливість може, наприклад, використовуватися в процесі розробки для пріоритезації вимог, визначаючи, наскільки цінною є дана вимога для певного користувача.

Відслідковування може також використовуватися після розгортання продукту. Наприклад, коли вивчення використання системи показує, що якась функція не використовується, можна визначити навіщо вона була потрібна спочатку.

Завдання управління вимогами

На кожному етапі процесу розробки існують ключові методи і задачі пов'язані з управлінням вимогами. Для ілюстрації, розглянемо наприклад стандартний процес розробки з п'ятьма фазами: дослідженням, аналізом здійсненності, дизайном, розробкою та тестуванням і випуском.

Дослідження. Під час фази дослідження збираються перші три класи вимог від користувачів, бізнесу і команди розробників. У кожній області задають однакові питання: які цілі, які обмеження, які використовуються процеси та інструменти і так далі. Тільки коли ці вимоги добре зрозумілі, можна приступати до розробки функціональних вимог.

Тут необхідне застереження: незалежно від того, як сильно група намагається це зробити, вимоги не можуть бути повністю визначені на початку проекту. Деякі вимоги змінюються, або тому що вони просто не були знайдені спочатку, або тому що внутрішні чи зовнішні сили торкаються проекту в середині циклу. Таким чином, учасники групи повинні спочатку погодитися, що головна умова успіху – гнучкість у мисленні та діях.

Результатом стадії дослідження є документ – специфікація вимог, схвалений усіма членами проекту. Пізніше, в процесі розробки, цей документ

КПЗ-2025

буде важливий для запобігання розповзання меж проекту або непотрібних змін. Оскільки система розвивається, кожна нова функція відкриває світ нових можливостей, таким чином специфікація вимог прив'язує команду до оригінального бачення системи і дозволяє контрольоване обговорення змін.

У той час як багато організацій все ще використовують звичайні документи для керування вимогами, інші управляють своїми базовими вимогами, використовуючи програмні інструменти.

Ці інструменти керують вимогами використовуючи базу даних, і зазвичай мають функції автоматизації відстеження (наприклад, дозволяючи створювати зв'язки між батьківськими і дочірніми вимогами, або між тестами і вимогами), управління версіями, і управління змінами. Зазвичай такі інструментальні засоби містять функцію експорту, яка дозволяє створювати звичайний документ, експортуючи дані вимог.

Аналіз здійсненності

На стадії аналізу здійсненності визначається вартість вимог. Для користувальницьких вимог поточна вартість роботи порівнюється з майбутньою вартістю встановленої системи. Задаються питання такі як: «Скільки нам зараз варті помилки введення даних?» Або, «Яка вартість втрати даних через помилки оператора пов'язаної з використовуваним інтерфейсом?». Фактично, потреба в новому інструменті часто розпізнається, коли подібні питання потрапляють до уваги людей, що займаються в організації фінансами.

Ділова вартість включає відповіді на такі питання як: «У якого відділу є бюджет на це?» «Який рівень повернення коштів від нового продукту на ринку?» «Який рівень скорочення внутрішніх витрат на навчання і підтримку, якщо ми зробимо нову, більш просту в використанні систему?»

Технічна вартість пов'язана з вартістю розробки програмного забезпечення та апаратною вартістю. «Чи є у нас потрібні люди, щоб створити інструмент?» «Чи потребуємо ми нове устаткування для підтримки нової системи?»

КПЗ-2025

Подібні питання дуже важливі. Група повинна з'ясувати, чи буде новий автоматизований інструмент мати достатню ефективність аби перенести частину тягаря користувачів на систему і зекономити час людей.

Ці питання також вказують на основну суть управління вимогами. Людина і інструмент формують систему, і це розуміння особливо важливе, якщо інструмент – комп'ютер або новий додаток на комп'ютері. Людський розум вкрай ефективний у паралельній обробці та інтерпретації тенденцій з недостатніми даними. Комп'ютерний процесор ефективний у послідовній обробці і точному математичному обчисленні. Основна мета управління вимогами для програмного проекту полягала б у тому, щоб гарантувати, що автоматизована робота призначена «правильному» процесору.

Наприклад, «не змушуйте людину пам'ятати, де вона знаходиться в системі. Примусьте інтерфейс завжди повідомляти про місцезнаходження людини в системі ». Або «не змушуйте людини вводити ті ж самі дані в два екрани. Примусьте систему зберігати дані і заповнювати їх де необхідно автоматично». Результатом стадії аналізу здійсненності є бюджет і графік проекту.

Дизайн. Припускаючи, що вартість точно визначена і переваги, які будуть отримані, є досить великими, проект може перейти до стадії проектування.

На стадії дизайну основна діяльність управління вимогами полягає в тому, щоб перевіряти чи відповідають результати дизайну документу вимог, щоб упевнитися, що робота залишається в межах проекту.

І знову, гнучкість є ключем до успіху. Ось класичний приклад змін проекту, які відмінно працювали. Проектувальники Форда на початку 1980-х очікували, що ціни на бензин піднімуться до 3,18 дол за галон до кінця десятиліття. На середині процесу дизайну автомобіля Ford Taurus, ціни встановилися приблизно на рівні 1,50 дол за галон. Колектив дизайнерів вирішив, що вони могли б створити більший, більш зручний, і більш потужний автомобіль, якщо б ціни на бензин залишилися низькими. Таким чином, вони перепроектувати

КПЗ-2025

автомобіль. Коли новий автомобіль вийшов, він встановив загальнонаціональні рекорди продажів.

У більшості випадків, однак, відступ від оригінальних вимог до такої міри не працює. Таким чином документ вимог стає ключовим інструментом, який допомагає команді приймати рішення про зміни дизайну.

Розробка та тестування. На стадії розробки і тестування, основна діяльність управління вимогами – це гарантувати, що робота і ціна залишаються в межах графіка і бюджету, і що створюваний інструмент дійсно відповідає вимогам. Основним інструментом, використовуваним на цій стадії, є створення прототипу і ітераційне тестування. Для програмного додатка користувацький інтерфейс може бути створений на папері і перевірений з потенційними користувачами, в той час як створюється основа програми. Результати цих тестів записуються в керівництві по дизайну користувацького інтерфейсу і передаються колективу дизайнерів. Це економить їх час і робить їх завдання набагато простіше.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Serpent – симетричний блочний алгоритм шифрування, розроблений Россом Андерсоном, Елі Біхамом та Ларсом Кнудсеном. Алгоритм був одним з фіналістів 2-го етапу конкурсу AES. Як і інші алгоритми, які брали участь у конкурсі AES, Serpent має розмір блоку 128 біт і можливі довжини ключа 128, 192 або 256 біт. Алгоритм являє собою 32-раундовий шифр на основі SP-мережі, і працює з блоком з чотирьох 32-бітових слів. Serpent був розроблений так, що всі операції можуть бути виконані паралельно, використовуючи 32-а 1-бітних «потоків».

При розробці Serpent використовувався консервативніший підхід до безпеки, ніж у інших фіналістів AES, проектувальники шифру вважали, що 16

КПЗ – 2025

раундів достатньо, щоб протистояти відомим видам криптоаналізу, але збільшили число раундів до 32, щоб алгоритм міг краще протистояти ще не відомим методам криптоаналізу.

Шифр Serpent не запатентований і є громадським надбанням.

Алгоритм створювався під гаслом «криптографічний алгоритм 21 століття» для участі в конкурсі AES. При створенні нового алгоритму Serpent його автори дотримувалися консервативних поглядів на проектування, що підтверджується первісним рішенням про використання таблиць підстановки з відомого багато років раніше алгоритму шифрування DES, який протягом довгого часу вивчався провідними фахівцями в області криптографії та захисту інформації і чиї властивості і особливості були добре відомі науковому світу. Одночасно з цим до нового алгоритму міг бути застосований вичерпний аналіз, вже розроблений для DES. Не використовувалися нові, неперевірені і невипробувані технології при створенні шифру, який у разі прийняття був би використаний для захисту величезних масивів фінансових транзакцій та урядової інформації. Основною вимогою до учасників конкурсу AES було те, що алгоритм-претендент повинен бути швидшим, ніж 3DES, і надавати як мінімум такий же рівень безпеки: він повинен мати блок даних довжиною 128 біт і ключ завдовжки 256 біт. 16-раундовий Serpent був би таким же надійним, як 3DES, але в два рази швидшим. Однак, автори вирішили, що для більшої надійності варто збільшити кількість раундів до 32. Це зробило шифр таким же швидким, як DES, і набагато надійнішим, ніж 3DES.

Структура алгоритму

Алгоритм Serpent є SP-мережею, у котрій весь блок даних довжиною 128 біт на кожному раунді розбивається на 4 слова довжиною 32 біти. Всі значення, що використовуються при шифруванні є бітовими потоками. Бітові індекси змінюють значення від 0 до 31 для 32-бітових слів, від 0 до 127 – для 128-бітових блоків та від 0 до 255 для 256-бітових ключів тощо. Для внутрішніх обчислень всі біти величин представлені в прямому порядку (little-endian).

КПЗ-2025

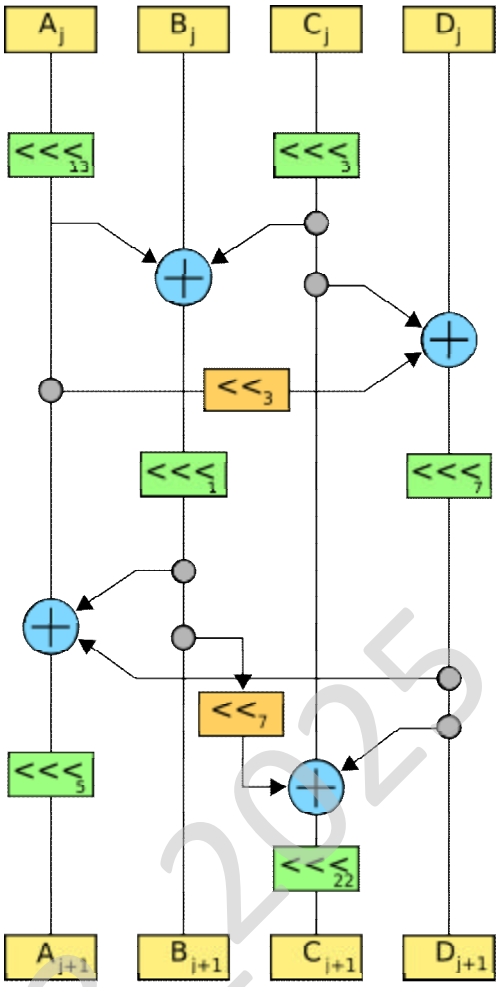

Рисунок 4.3 – Структура алгоритму Serpent

Serpent шифрує відкритий текст P довжиною 128 біт в шифротекст C довжиною таких же 128 біт за 32 раунд за допомогою 33 подключів K{0},...,K{32}, довжиною 128 біт. Довжина використовуваного ключа може приймати різні значення, але для конкретики зафіксуємо їх довжину в 128, 192 або 256 біт. Короткі ключі довжиною менше 256 біт доповнюються до повної довжини в 256 біт.

Шифрування складається з наступних основних кроків:

– Початкова перестановка.

– 32 раунд, кожен з яких складається з операції змішування з 128-бітовим ключем (побітове логічне виключаюче «або»), таблична заміна (S-box) і лінійне

КБПЗ_2025

перетворення. В останньому раунді лінійне перетворення замінюється додатковим накладанням ключа.

– Кінцева перестановка.

Початкова і кінцева перестановки не мають будь-якої криптографічного значущості. Вони використовуються для спрощення оптимізованої реалізації алгоритму і підвищення обчислювальної ефективності.

Розширення ключа

Як і інші алгоритми, що брали участь в конкурсі AES, Serpent має можливі довжини ключа 128, 192 або 256 біт. «Неповний» ключ довжиною менше 256 біт доповнюється за наступним правилом: додається одиничний біт справа, за ним слід стільки нульових бітів, щоб довжина ключа стала дорівнює 256 бітам.

Початкова перестановка IP

Дана перестановка IP задається таблицею, де вказується позиція, на яку перейде відповідний біт (наприклад, біт 1 перейде на 32 позицію):

S-бокси (таблиці замін)

В алгоритмі Serpent таблиці замін є 4-бітовими перестановками з властивостями стійкості до диференціального криптоаналізу, до лінійного криптоаналізу і такою властивістю, що порядок вихідних біт, як функції вхідних повинен бути максимальний, тобто бути рівним 3.

Таблиця підстановки генерується з відомих і добре вивчених таблиць для алгоритму DES в ітераційному процесі, поки не будуть отримані бажані диференціальні й лінійні властивості. Таким чином, створюється 8 таблиць підстановки.

Лінійне перетворення LT

Лінійне перетворення LTзадається таблицею, де біти перераховані від 0 до 127 (наприклад, вихідний 2 біт утворений 2, 9, 15, 30, 76, 84, 126 битами, складеними за модулем 2) . В кожному рядку описується 4 вихідних біти, які разом складають вхідні дані на одну таблицю замін в наступному раунді. Варто

КБПЗ-2025

зазначити, що даний набір являє собою таблицю IP(LT(FP(x))), де LT і є те лінійне перетворення.

Таблиця зворотного лінійного перетворення, яке використовується при розшифровці ILT.

Кінцева перестановка FP

Дана перестановка є зворотною до початкової, тобто $FP=IP^{-1}$ і задається наступною таблицею.

Ефективна реалізація алгоритму

Бажання авторів зробити алгоритм саме таким, яким він є стає зрозумілим при розгляді його ефективної низькорівневої реалізації.

Serpent був створений таким чином, щоб всі операції в процесі шифрування і розшифрування одного блоку могли бути виконані паралельно в 32 потоках. До того ж низькорівневий опис алгоритму набагато простішій, ніж стандартний опис. Ніяких початкових і кінцевих перестановок не потрібно.

Шифрування складається з 32 раундів. Відкритий текст є першими проміжними даними $B_0 = P$. Потім виконується 32 раунди, кожен і-й раунд складається з:

– Змішування з ключем. Проводиться побітове виключаюче «або» проміжних даних B_i з ключем довжиною 128 біт.

– Застосування таблиць підстановки. Вхідні дані довжиною 128 біт поділяються на 4 слова по 32 біта. Таблиця підстановки, реалізована послідовністю логічних операцій (як якщо це було б реалізовано апаратно), застосовується до цих 4 словам. В результаті виходить 4 вихідних слова. Таким чином, центральний процесор виконує підстановку по 32 копій таблиці одночасно.

– Лінійне перетворення. 32-бітові слова перетворюються заданим порядком.

Першою причиною вибору такого лінійного перетворення є максимізація лавинного ефекту. Такі таблиці підстановки мають властивість, що зміна кожного

КОПЗ_2025

вхідного біта призведе до зміни 2 вихідних бітів. Таким чином, кожен вхідний біт відкритого тексту вже через 3 раунди впливає на всі вихідні біти. Аналогічно кожен біт ключа впливає на результат шифрування.

Друга причина полягає в простоті перетворення. Воно може бути реалізоване на будь-якому сучасному процесорі з мінімальними витратами.

КБПЗ_2025

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ гібридної кабельної системи відеонагляду з використанням технології PFCS яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Налаштування; Камери; Вікна; Довідка.

– Розділу обрання камери.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

КбПЗ – 2025

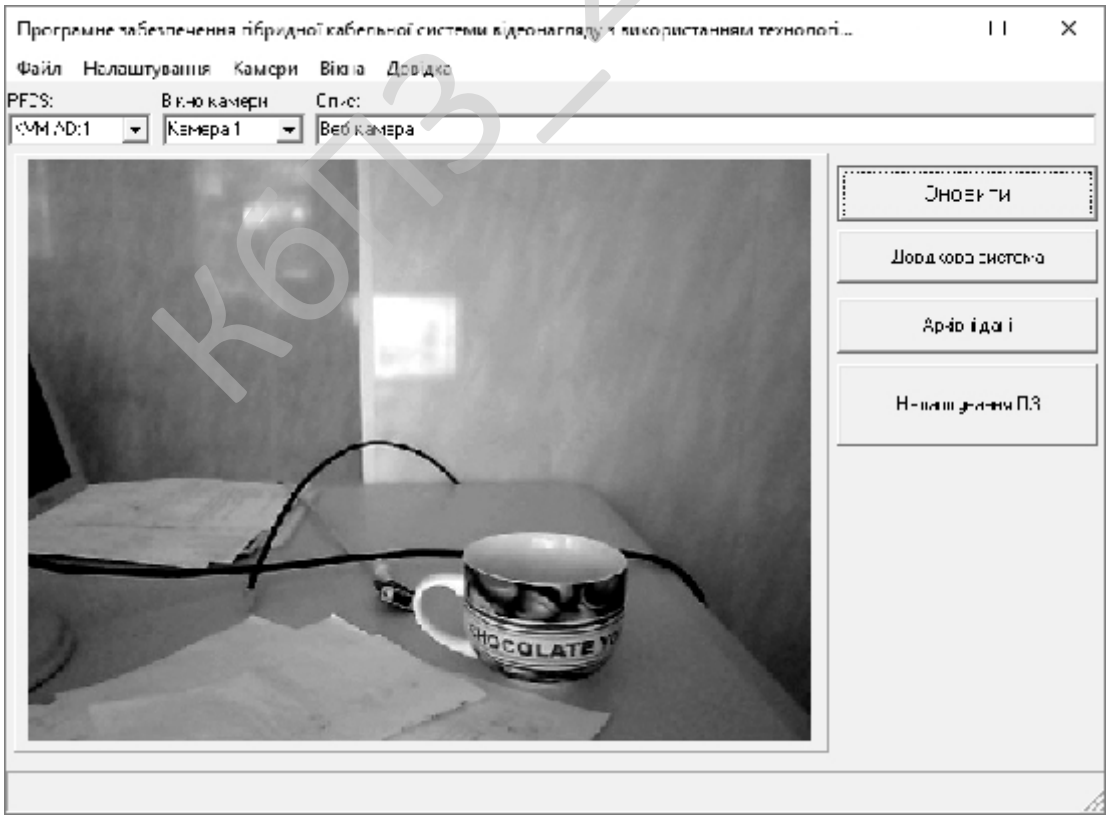

Рисунок 5.1 – Головне вікно ПЗ

Система відеоспостереження складається з однієї або декількох відеокамер, відеореєстратора й монітора. Відеореєстратор дозволяє записувати кілька тижнів підряд зображення з камер спостереження, зберігати відеозапису в архіві й передавати зображення за допомогою мережі Інтернет у будь-яку точку миру. Камери відеоспостереження можуть установлюватися на вулиці або усередині приміщень, вони відрізняються конструктивними особливостями й мають різні характеристики, наприклад, можуть передавати або кольорове, або чорно-біле зображення.

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.

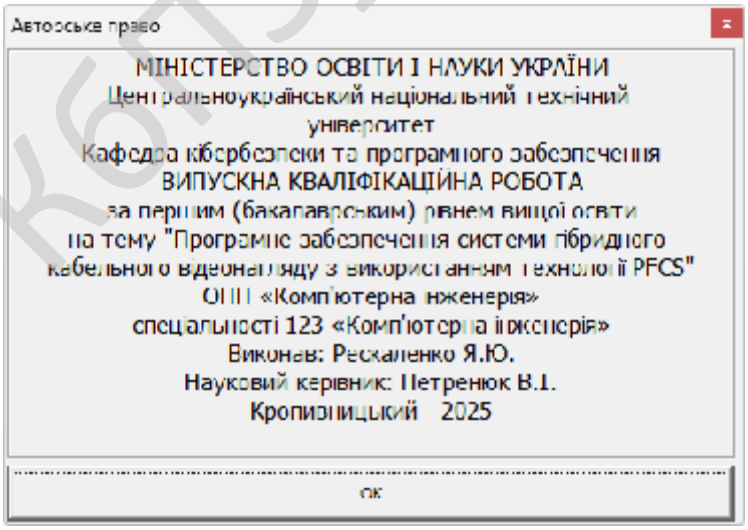

Рисунок 5.2 – Авторське право

КПЗ_2025

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 IT рішення для автоматизації операцій усередині саме цих

КбПЗ_2025

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 ІТ 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.

КПЗ-2025

Проводилось тестування форматом 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 OT).

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 OT.

КОПЗ-2025

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 – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

КОПІЯ 2025

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи гібридного кабельного відеонагляду з використанням технології PFCS.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем гібридного кабельного відеонагляду з використанням технології PFCS.

– Досліджена система гібридного кабельного відеонагляду з використанням технології PFCS.

– На основі отриманих результатів досліджень створена програмна реалізація системи гібридного кабельного відеонагляду з використанням технології PFCS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання гібридного кабельного відеонагляду з використанням технології PFCS.

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 гібридного кабельного відеонагляду з використанням технології PFCS.

КОПЗ_2025

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 Serpent.

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.

КбПЗ_2025

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.

КПЗ-2025

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

КОПЗ–2025

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.

КОПІЯ 2025

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,

КОПЗ_2025

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

КБПЗ – 2025

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.

КОПЗ_2025

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Перелік документів, що розробляються.....5
8 Етапи розробки.....6
9 Порядок контролю та приймання.....6

КОПЗ-2025

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи гібридного кабельного відеонагляду з використанням технології PFCS.

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. № 47-02 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 гібридного кабельного відеонагляду з використанням технології PFCS.

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;

КБПЗ_2025

– 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 гібридного кабельного відеонагляду з використанням технології PFCS;

– 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.

КБПЗ_2025

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 забезпечення.

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.

КбПЗ – 2025

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 – 63 аркуші.

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).

КБПЗ_2025

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 5.06.2025 р.

КбПЗ_2025

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Петренюк В.І.

Програмне забезпечення системи гібридного кабельного відеонагляду з використанням технології PFCS

КбПЗ_2025

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
# системf гібридного кабельного відеонагляду з використанням технології PFCS
# Імпортуємо необхідні модулі для роботи з часом
# випадковими подіями, потоками, логуванням та чергами

import time
import random
import threading
import logging
import queue

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

# Глобальні змінні для керування станом системи
SYSTEM_RUNNING = True
CAMERA_COUNT = 4

# Клас PFCSController відповідає за імітацію роботи технології PFCS
class PFCSController:
# Ініціалізація контролера PFCS
    def __init__(self):
# Ініціалізуємо чергу для команд
        self.command_queue = queue.Queue()
# Ініціалізуємо чергу для відповідей
        self.response_queue = queue.Queue()
# Встановлюємо початковий стан контролера
        self.state = "IDLE"
# Записуємо в лог повідомлення про ініціалізацію контролера
        logging.debug("PFCSController initialized with state IDLE")

# Метод для відправки команди на пристрій PFCS
    def send_command(self, command):
# Логування відправки команди
        logging.debug("Sending command to PFCS device: %s", command)
# Додаємо команду в чергу
        self.command_queue.put(command)
# Симулюємо затримку передачі команди
        time.sleep(random.uniform(0.1, 0.3))
# Отримуємо відповідь на команду
        response = self.receive_response(command)
        return response

# Метод для отримання відповіді на команду
    def receive_response(self, command):
# Список можливих відповідей з пристрою PFCS
        responses = ["ACK", "NACK", "ERROR", "SUCCESS"]
# Випадковим чином вибираємо відповідь
        response = random.choice(responses)
# Логування отриманої відповіді
        logging.debug("Received response '%s' for command: %s", response,
command)
        return response

# Метод для безперервної обробки команд з черги
    def process_commands(self):
```

КОПЗ_2025

```
# Обробка команд у циклі, поки система працює
        while SYSTEM_RUNNING:
            if not self.command_queue.empty():
# Отримуємо команду з черги
                command = self.command_queue.get()
# Логування обробки команди
                logging.debug("Processing command from queue: %s", command)
# Отримуємо відповідь на команду
                response = self.receive_response(command)
# Додаємо відповідь в чергу відповідей
                self.response_queue.put(response)
            time.sleep(0.05)

# Метод для запуску обробки команд у фоновому потоці
    def start_processing(self):
# Логування запуску потоку обробки команд PFCS
        logging.debug("Starting PFCS command processing thread")
# Створення та запуск нового потоку для обробки команд
        thread = threading.Thread(target=self.process_commands, daemon=True)
        thread.start()

# Клас CableInterface моделює інтерфейс кабельного з'єднання та його перевірку
class CableInterface:
# Ініціалізація інтерфейсу кабелю
    def __init__(self):
# Встановлюємо початковий стан підключення
        self.connected = False
# Початкова сила сигналу
        self.signal_strength = 0.0
# Викликаємо метод ініціалізації інтерфейсу
        self.initialize_interface()

# Метод для ініціалізації інтерфейсу кабелю
    def initialize_interface(self):
# Логування процесу ініціалізації інтерфейсу
        logging.debug("Initializing cable interface")
# Стан підключення встановлено в True
        self.connected = True
# Генеруємо випадкову силу сигналу в межах високої якості
        self.signal_strength = random.uniform(0.8, 1.0)
# Логування параметрів ініціалізації
        logging.debug("Cable interface initialized: connected=%s,
signal_strength=%.2f", self.connected, self.signal_strength)

# Метод для перевірки сили сигналу кабелю
    def check_signal(self):
# Якщо кабель не підключено, логування помилки
        if not self.connected:
            logging.error("Cable interface not connected")
            return 0.0
# Оновлюємо силу сигналу
        self.signal_strength = random.uniform(0.7, 1.0)
# Логування перевіреної сили сигналу
        logging.debug("Cable signal strength checked: %.2f",
self.signal_strength)
        return self.signal_strength

# Метод для симуляції переривання кабельного з'єднання
    def simulate_disruption(self):
# Логування симуляції переривання зв'язку
        logging.debug("Simulating cable disruption")
# Встановлюємо стан відключення
        self.connected = False
```

КОТБ – 2025

```
# Затримка для імітації переривання
        time.sleep(random.uniform(0.5, 1.0))
# Переініціалізуємо інтерфейс кабелю після переривання
        self.initialize_interface()

# Клас Camera моделює роботу відеокамери з інтеграцією PFCS технології
class Camera:
# Ініціалізація камери з її ідентифікатором та залежностями
    def __init__(self, camera_id, pfcs_controller, cable_interface):
# Присвоюємо унікальний ідентифікатор камери
        self.camera_id = camera_id
# Зберігаємо екземпляр контролера PFCS
        self.pfcs_controller = pfcs_controller
# Зберігаємо екземпляр кабельного інтерфейсу
        self.cable_interface = cable_interface
# Початковий стан потокової передачі даних
        self.streaming = False
# Лічильник оброблених кадрів
        self.frame_count = 0
# Логування успішної ініціалізації камери
        logging.debug("Camera %d initialized", self.camera_id)

# Метод для запуску потокової передачі відео
    def start_stream(self):
# Перевірка якості сигналу перед запуском стріму
        if self.cable_interface.check_signal() < 0.75:
# Логування попередження про слабкий сигнал
            logging.warning("Camera %d: Weak cable signal, cannot start stream",
self.camera_id)
            return
# Встановлюємо стан потокової передачі в True
        self.streaming = True
# Логування старту потокової передачі
        logging.debug("Camera %d stream started", self.camera_id)
# Запуск нового потоку для імітації відеопотоку
        thread = threading.Thread(target=self.stream_video, daemon=True)
        thread.start()

# Метод для зупинки потокової передачі відео
    def stop_stream(self):
# Встановлюємо стан потокової передачі в False
        self.streaming = False
# Логування зупинки потоку
        logging.debug("Camera %d stream stopped", self.camera_id)

# Метод для симуляції процесу потокового передавання відео
    def stream_video(self):
# Циклічне виконання, поки камера передає відео
        while self.streaming:
# Симуляція затримки між кадрами
            time.sleep(0.1)
# Збільшуємо лічильник кадрів
            self.frame_count += 1
# Захоплення кадру з камери
            frame = self.capture_frame()
# Обробка отриманого кадру
            processed_frame = self.process_frame(frame)
# Формування команди для передачі інформації через PFCS
            command = f"CAM{self.camera_id}_FRAME_{self.frame_count}"
# Відправка команди через PFCS контролер та отримання відповіді
            response = self.pfcs_controller.send_command(command)
# Логування результату передачі кадру
```

КПТВ_2025

```
            logging.debug("Camera %d sent frame %d, PFCS response: %s",
self.camera_id, self.frame_count, response)
# Періодичне виконання профілактичних операцій після певної кількості кадрів
            if self.frame_count % 50 == 0:
                self.perform_maintenance()

# Метод для симуляції захоплення кадру з камери
    def capture_frame(self):
# Формуємо дані кадру з унікальним номером
        frame = f"Frame_{self.frame_count}_Data"
# Логування процесу захоплення кадру
        logging.debug("Camera %d captured frame %d", self.camera_id,
self.frame_count)
        return frame

# Метод для обробки захопленого кадру
    def process_frame(self, frame):
# Замінюємо частину даних для імітації обробки кадру
        processed = frame.replace("Data", "Processed")
# Логування обробки кадру
        logging.debug("Camera %d processed frame %d", self.camera_id,
self.frame_count)
        return processed

# Метод для проведення профілактичного обслуговування камери
    def perform_maintenance(self):
# Логування початку профілактики
        logging.debug("Camera %d performing maintenance", self.camera_id)
# Перевірка стану апаратного забезпечення камери
        status = self.check_hardware()
# Логування результатів перевірки апаратної частини
        logging.debug("Camera %d hardware status: %s", self.camera_id, status)

# Метод для перевірки апаратного стану камери
    def check_hardware(self):
# Визначаємо можливі стани апаратної частини
        statuses = ["OK", "WARNING", "FAIL"]
# Випадковим чином обираємо стан апаратного забезпечення
        status = random.choice(statuses)
# Логування результату апаратної перевірки
        logging.debug("Camera %d hardware check: %s", self.camera_id, status)
        return status

# Клас SurveillanceManager відповідає за управління всіма камерами та загальним
контролем системи
class SurveillanceManager:
# Ініціалізація менеджера системи з кількістю камер
    def __init__(self, camera_count):
# Створюємо екземпляр контролера PFCS
        self.pfcs_controller = PFCSController()
# Створюємо екземпляр кабельного інтерфейсу
        self.cable_interface = CableInterface()
# Ініціалізуємо список камер
        self.cameras = []
# Запам'ятовуємо кількість камер
        self.camera_count = camera_count
# Налаштовуємо всі камери
        self.setup_cameras()
# Запускаємо обробку команд контролера PFCS
        self.pfcs_controller.start_processing()

# Метод для налаштування та додавання камер в систему
    def setup_cameras(self):
```

КПЗ-2025

```
# Логування процесу налаштування камер
        logging.debug("Setting up %d cameras", self.camera_count)
# Ініціалізація кожної камери з унікальним ідентифікатором
        for i in range(1, self.camera_count + 1):
            cam = Camera(i, self.pfcs_controller, self.cable_interface)
# Додавання камери в список системи
            self.cameras.append(cam)
# Логування додавання камери до системи
            logging.debug("Camera %d added to the system", i)
            time.sleep(0.05)

# Метод для запуску потоків всіх камер
    def start_all_streams(self):
# Логування початку запуску потоків камер
        logging.debug("Starting all camera streams")
# Запуск потокової передачі для кожної камери
        for cam in self.cameras:
            cam.start_stream()
            time.sleep(0.1)

# Метод для зупинки потоків всіх камер
    def stop_all_streams(self):
# Логування процесу зупинки всіх потоків
        logging.debug("Stopping all camera streams")
# Зупинка кожної камери окремо
        for cam in self.cameras:
            cam.stop_stream()
            time.sleep(0.05)

# Метод для запиту статусу кожної камери
    def query_camera_status(self):
# Логування початку запиту статусу камер
        logging.debug("Querying status of all cameras")
# Ініціалізація звіту статусу
        status_report = {}
# Проходимо по всіх камерах в системі
        for cam in self.cameras:
            status = {
                "streaming": cam.streaming,
                "frame_count": cam.frame_count,
                "hardware_status": cam.check_hardware()
            }
# Формуємо звіт для кожної камери
            status_report[f"Camera_{cam.camera_id}"] = status
# Логування отриманого статусу камери
            logging.debug("Status for Camera %d: %s", cam.camera_id, status)
            time.sleep(0.02)
        return status_report

# Метод для проведення комплексної перевірки всієї системи
    def perform_system_check(self):
# Логування початку системної перевірки
        logging.debug("Performing comprehensive system check")
# Отримання статусу контролера PFCS
        pfcs_status = self.check_pfcs_status()
# Перевірка сили сигналу кабелю
        cable_status = self.cable_interface.check_signal()
# Отримання статусу камер
        camera_status = self.query_camera_status()
# Формування загального звіту системи
        system_status = {
            "pfcs_status": pfcs_status,
            "cable_signal": cable_status,
```

КПТЗ_2025

```
            "camera_status": camera_status
        }
# Логування завершення системної перевірки
        logging.debug("System check complete: %s", system_status)
        return system_status

# Метод для перевірки статусу контролера PFCS
    def check_pfcs_status(self):
# Логування початку перевірки PFCS контролера
        logging.debug("Checking PFCS controller status")
# Симулюємо перевірку та визначення статусу контролера
        status = "OPERATIONAL" if random.random() > 0.1 else "ERROR"
# Логування отриманого статусу PFCS контролера
        logging.debug("PFCS controller status: %s", status)
        return status

# Клас QueryHandler відповідає за обробку запитів до системи відеонагляду
class QueryHandler:
# Ініціалізація обробника запитів із прив'язкою до менеджера системи
    def __init__(self, surveillance_manager):
# Прив'язуємо менеджер системи до обробника запитів
        self.surveillance_manager = surveillance_manager

# Метод для обробки запитів і повернення результатів
    def handle_query(self, query):
# Логування отриманого запиту
        logging.debug("Handling query: %s", query)
# Обробка запиту для отримання загального статусу системи
        if query == "STATUS":
            result = self.surveillance_manager.perform_system_check()
# Обробка запиту для конкретної камери
        elif query.startswith("CAMERA"):
            parts = query.split()
            if len(parts) > 1:
                try:
                    cam_id = int(parts[1])
                    if 1 <= cam_id <= self.surveillance_manager.camera_count:
                        cam = self.surveillance_manager.cameras[cam_id - 1]
                        result = {
                            "camera_id": cam.camera_id,
                            "streaming": cam.streaming,
                            "frame_count": cam.frame_count,
                            "hardware_status": cam.check_hardware()
                        }
                    else:
                        result = {"error": "Invalid camera ID"}
                except ValueError:
                    result = {"error": "Camera ID must be an integer"}
            else:
                result = {"error": "No camera ID provided"}
# Обробка невідомих запитів
        else:
            result = {"error": "Unknown query"}
# Логування результату обробки запиту
        logging.debug("Query result: %s", result)
        return result

# Функція для періодичного виконання запитів до системи у окремому потоці
def periodic_query(query_handler):
# Циклічне виконання запитів протягом роботи системи
    while SYSTEM_RUNNING:
# Затримка між запитами для стабільної роботи системи
        time.sleep(2)
```

КоПЗ_2025

```
# Отримання загального статусу системи
        result = query_handler.handle_query("STATUS")
# Логування періодичного звіту статусу системи
        logging.info("Periodic system status: %s", result)
        time.sleep(2)
# Отримання запиту до випадкової камери
        random_cam = random.randint(1, CAMERA_COUNT)
        query = f"CAMERA {random_cam}"
        cam_result = query_handler.handle_query(query)
# Логування результату запиту до камери
        logging.info("Periodic camera query (%s): %s", query, cam_result)
        time.sleep(1)

# Основна функція для запуску системи відеонагляду
def main():
# Логування початку ініціалізації менеджера системи
    logging.debug("Initializing Surveillance Manager")
# Створення менеджера системи з заданою кількістю камер
    manager = SurveillanceManager(CAMERA_COUNT)

# Запуск потокової передачі для всіх камер
    manager.start_all_streams()

# Ініціалізація обробника запитів із системним менеджером
    query_handler = QueryHandler(manager)

# Створення та запуск потоку для періодичних запитів
    query_thread = threading.Thread(target=periodic_query,
args=(query_handler,), daemon=True)
    query_thread.start()

# Задаємо тривалість роботи системи в секундах
    run_duration = 30  # seconds
    start_time = time.time()
# Циклічний цикл для роботи системи протягом встановленого часу
    while time.time() - start_time < run_duration:
        time.sleep(0.5)

# Зупинка потокової передачі для всіх камер перед завершенням роботи
    manager.stop_all_streams()
# Логування ініціації завершення роботи системи
    logging.debug("Surveillance system shutdown initiated")

# Додаткові допоміжні функції для розширеної роботи системи

def advanced_logging():
# Функція для реалізації розширеного логування з детальним часовим відображенням
    logging.debug("Advanced logging feature activated")
    for i in range(5):
# Логування детального запису стану системи
        logging.debug("Advanced log entry %d: System running smoothly", i+1)
        time.sleep(0.05)

def simulate_network_requests():
# Функція для симуляції мережевих запитів до контролера PFCS
    logging.debug("Simulating network requests for advanced control")
    for i in range(3):
# Формування команди мережевого запиту
        command = f"NET_REQ_{i+1}"
# Випадковий вибір відповіді для мережевого запиту
        response = random.choice(["OK", "TIMEOUT", "ERROR"])
# Логування результату мережевого запиту
        logging.debug("Network request %s: Response %s", command, response)
```

КБТЗ_2025

```
        time.sleep(0.1)

def perform_diagnostics():
# Функція для проведення діагностики системи та перевірки її здоров'я
    logging.debug("Performing system diagnostics")
# Формування результатів діагностики за трьома основними параметрами
    diagnostic_results = {}
    diagnostic_results["network"] = random.choice(["GOOD", "AVERAGE", "POOR"])
    diagnostic_results["hardware"] = random.choice(["STABLE", "UNSTABLE",
"CRITICAL"])
    diagnostic_results["software"] = random.choice(["OPTIMAL", "DEGRADED",
"FAIL"])
# Логування отриманих результатів діагностики
    logging.debug("Diagnostics results: %s", diagnostic_results)
    return diagnostic_results

def run_additional_services():
# Функція для запуску додаткових сервісів віддаленого моніторингу та керування
    logging.debug("Starting additional remote monitoring services")
    advanced_logging()
    simulate_network_requests()
    diagnostics = perform_diagnostics()
# Логування завершення роботи додаткових сервісів
    logging.debug("Additional services completed with diagnostics: %s",
diagnostics)

def start_additional_services():
# Функція для запуску додаткових сервісів у окремому потоці
    logging.debug("Initializing additional services thread")
    services_thread = threading.Thread(target=run_additional_services,
daemon=True)
    services_thread.start()

# Точка входу в програму
if __name__ == "__main__":
# Запуск додаткових сервісів для розширеного моніторингу
    start_additional_services()
# Запуск основної системи відеонагляду
    main()
# Виконуємо фінальну діагностику системи після зупинки роботи
    final_diagnostics = perform_diagnostics()
# Логування остаточних результатів діагностики системи
    logging.debug("Final system diagnostics: %s", final_diagnostics)
# Логування завершення роботи системи відеонагляду
```

КБПЗ_2025

Файл detect_faces.py

```
#!/usr/bin/env python3
# Імпортуємо базові бібліотеки для роботи системи
import logging
import time
import random
import threading
import numpy as np
import cv2
import queue
from flask import Flask, jsonify, request

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

# ================================================================
# Клас FaceRecognitionModule - модуль розпізнавання облич у відеопотоці
# ================================================================
class FaceRecognitionModule:
    # Ініціалізація модуля розпізнавання облич
    def __init__(self):
        # Завантаження попередньо навченої моделі для детекції облич
        self.face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
        # Логування успішного завантаження моделі
        logging.debug("FaceRecognitionModule ініціалізовано з використанням
каскадного класифікатора")

    # Метод для детекції облич на переданому кадрі
    def detect_faces(self, frame):
        # Перетворення кадру в відтінки сірого для спрощення обробки
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Виконання детекції облич за допомогою класифікатора
        faces = self.face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
        # Логування кількості виявлених облич
        logging.debug("Виявлено %d облич на кадрі", len(faces))
        return faces

    # Метод для симуляції генерації відеокадру
    def simulate_frame(self):
        # Генерація випадкового зображення розміром 480x640 з трьома каналами
кольору
        frame = np.random.randint(0, 256, (480, 640, 3), dtype=np.uint8)
        # Логування генерації симульованого кадру
        logging.debug("Симульовано новий відеокадр")
        return frame

# ================================================================
# Клас AnomalyDetectionModule - модуль аналізу аномалій у відеопотоці
# ================================================================
class AnomalyDetectionModule:
    # Ініціалізація модуля аналізу аномалій
    def __init__(self):
        # Встановлення порогового значення яскравості для визначення аномалій
        self.brightness_threshold = 180
        # Логування налаштувань модуля аномалій
        logging.debug("AnomalyDetectionModule ініціалізовано з порогом
яскравості %d", self.brightness_threshold)
```

КОПІЯ - 2025

```
    # Метод для виявлення аномалій на базі аналізу кадру
    def detect_anomalies(self, frame):
        # Обчислення середнього значення яскравості кадру
        avg_brightness = np.mean(frame)
        # Логування обчисленої середньої яскравості
        logging.debug("Середня яскравість кадру: %.2f", avg_brightness)
        # Визначення аномалії, якщо середня яскравість перевищує порогове
значення
        if avg_brightness > self.brightness_threshold:
            logging.debug("Аномалія виявлена: середня яскравість перевищує
поріг")
            return True
        else:
            logging.debug("Аномалія не виявлена: середня яскравість у нормі")
            return False

# ========================================================================
# Клас NotificationSystem – система сповіщень для надсилання повідомлень
# ========================================================================
class NotificationSystem:
    # Ініціалізація системи сповіщень
    def __init__(self):
        # Ініціалізація параметрів підключення для email, SMS, push
        self.email_server = "smtp.example.com"
        self.sms_gateway = "sms.example.com"
        self.push_service = "push.example.com"
        # Логування налаштувань системи сповіщень
        logging.debug("NotificationSystem ініціалізовано з серверами: email=%s,
sms=%s, push=%s",
                      self.email_server, self.sms_gateway, self.push_service)

    # Метод для надсилання email-повідомлення
    def send_email_notification(self, message):
        # Симуляція відправлення email-повідомлення
        logging.debug("Відправлення email-повідомлення: %s", message)
        time.sleep(0.1)
        logging.debug("Email-повідомлення відправлено успішно")

    # Метод для надсилання SMS-повідомлення
    def send_sms_notification(self, message):
        # Симуляція відправлення SMS-повідомлення
        logging.debug("Відправлення SMS-повідомлення: %s", message)
        time.sleep(0.1)
        logging.debug("SMS-повідомлення відправлено успішно")

    # Метод для надсилання push-повідомлення
    def send_push_notification(self, message):
        # Симуляція відправлення push-повідомлення
        logging.debug("Відправлення push-повідомлення: %s", message)
        time.sleep(0.1)
        logging.debug("Push-повідомлення відправлено успішно")

# ========================================================================
# Клас AnalyticsStatisticsModule – модуль збору аналітики та статистичних даних
# ========================================================================
class AnalyticsStatisticsModule:
    # Ініціалізація модуля аналітики
    def __init__(self):
        # Ініціалізація списку для зберігання подій системи
        self.events = []
```

КОПЗ-2025

```
        # Логування запуску модуля аналітики
        logging.debug("AnalyticsStatisticsModule успішно ініціалізовано")

    # Метод для запису події в систему аналітики
    def record_event(self, event_type, details):
        # Створення запису події з позначкою часу
        event = {
            "timestamp": time.time(),
            "event_type": event_type,
            "details": details
        }
        # Додавання події до списку подій
        self.events.append(event)
        # Логування запису події
        logging.debug("Записано подію: %s", event)

    # Метод для генерації звіту з аналітики на основі зібраних даних
    def generate_report(self):
        # Ініціалізація змінних для підрахунку різних типів подій
        report = {}
        # Проходимо по всіх записах подій
        for event in self.events:
            event_type = event["event_type"]
            if event_type not in report:
                report[event_type] = 0
            report[event_type] += 1
        # Додавання детальної інформації до звіту
        report["total_events"] = len(self.events)
        report["generated_at"] = time.time()
        # Логування згенерованого звіту
        logging.debug("Звіт аналітики: %s", report)
        return report

# ================================================================================
# Клас WebInterface - веб-інтерфейс управління системою на базі Flask
# ================================================================================
class WebInterface:
    # Ініціалізація веб-інтерфейсу з переданим модулем аналітики
    def __init__(self, analytics_module):
        # Створення об'єкта Flask
        self.app = Flask(__name__)
        # Збереження посилання на модуль аналітики для використання у маршрутах
        self.analytics_module = analytics_module
        # Виклик методу налаштування маршрутів
        self.setup_routes()
        # Логування ініціалізації веб-інтерфейсу
        logging.debug("WebInterface ініціалізовано та маршрути налаштовано")

    # Метод для налаштування веб-маршрутів
    def setup_routes(self):
        # Маршрут для отримання статусу системи
        @self.app.route('/status', methods=['GET'])
        def status():
            # Формування відповіді зі статусом системи
            response = {
                "status": "Система працює",
                "timestamp": time.time()
            }
            # Логування запиту до маршруту /status
            logging.debug("Оброблено запит до /status")
            return jsonify(response)
```

КОПІЯ - 2025

```
        # Маршрут для отримання звіту аналітики
        @self.app.route('/analytics', methods=['GET'])
        def analytics():
            # Отримання звіту з аналітичного модуля
            report = self.analytics_module.generate_report()
            # Логування запиту до маршруту /analytics
            logging.debug("Оброблено запит до /analytics")
            return jsonify(report)

        # Маршрут для отримання інформації про події
        @self.app.route('/events', methods=['GET'])
        def events():
            # Повернення повного списку подій
            logging.debug("Оброблено запит до /events")
            return jsonify(self.analytics_module.events)

        # Маршрут для симуляції отримання повідомлення від клієнта
        @self.app.route('/notify', methods=['POST'])
        def notify():
            # Отримання даних з запиту
            data = request.get_json()
            # Логування отриманих даних
            logging.debug("Отримано дані в /notify: %s", data)
            # Повернення підтвердження отримання даних
            return jsonify({"status": "дані отримано", "data": data})

    # Метод для запуску Flask веб-сервера
    def run(self, host='0.0.0.0', port=5000):
        # Логування запуску веб-сервера
        logging.debug("Запуск Flask веб-сервера на %s:%d", host, port)
        # Запуск сервера із вимкненим режимом налагодження
        self.app.run(host=host, port=port, debug=False, use_reloader=False)

# ================================================================
# Головна функція інтеграції всіх модулів та запуску системи
# ================================================================
def main():
    # Логування старту головної функції
    logging.debug("Старт головної функції інтегрованої системи")

    # Ініціалізація модуля розпізнавання облич
    face_module = FaceRecognitionModule()
    # Ініціалізація модуля аналізу аномалій
    anomaly_module = AnomalyDetectionModule()
    # Ініціалізація системи сповіщень
    notification_system = NotificationSystem()
    # Ініціалізація модуля аналітики та статистики
    analytics_module = AnalyticsStatisticsModule()
    # Ініціалізація веб-інтерфейсу управління системою
    web_interface = WebInterface(analytics_module)

    # Запуск веб-сервера у фоновому потоці
    web_thread = threading.Thread(target=web_interface.run, kwargs={'host':
'127.0.0.1', 'port': 5000}, daemon=True)
    web_thread.start()
    logging.debug("Веб-сервер запущено у фоновому потоці")

    # Задаємо тривалість симуляції роботи системи (в секундах)
    simulation_duration = 60
    simulation_start = time.time()
```

КОПВ-2025

```
    # Основний цикл симуляції роботи системи
    while time.time() - simulation_start < simulation_duration:
        # Генеруємо новий симульований кадр за допомогою модуля розпізнавання
облич
        frame = face_module.simulate_frame()

        # Виконуємо детекцію облич на симульованому кадрі
        faces = face_module.detect_faces(frame)
        # Якщо виявлено обличчя, записуємо подію в модуль аналітики
        if len(faces) > 0:
            analytics_module.record_event("face_detection", {"face_count":
len(faces)})

        # Виконуємо аналіз кадру на наявність аномалій
        anomaly_found = anomaly_module.detect_anomalies(frame)
        # Якщо виявлено аномалію, фіксуємо подію та надсилаємо сповіщення
        if anomaly_found:
            analytics_module.record_event("anomaly_detection", {"detail":
"Аномалія в кадрі"})
            notification_system.send_email_notification("Попередження: аномалія
виявлена в відеопотоці")
            notification_system.send_sms_notification("Аномалія виявлена!")
            notification_system.send_push_notification("Увага: аномалія у
відео")

        # Затримка між ітераціями симуляції для стабільної роботи
        time.sleep(1)

    # Після завершення симуляції генеруємо фінальний звіт аналітики
    final_report = analytics_module.generate_report()
    # Логування фінального звіту аналітики
    logging.info("Фінальний звіт аналітики: %s", final_report)

    # Підтримуємо роботу системи короткий час для обслуговування веб-запитів
    logging.debug("Очікування завершення роботи системи для обслуговування веб-
запитів")
    time.sleep(10)
    logging.debug("Завершення роботи інтегрованої системи")

# ================================================================
# Точка входу в програму
# ================================================================
if __name__ == "__main__":
    # Запуск головної функції інтегрованої системи спостереження
    main()
```

КБПЗ-2025

Файл basicConfig.py

```
#!/usr/bin/env python3
# Імпортуємо базові бібліотеки для роботи розширеного функціоналу
import logging
import time
import random
import threading
import queue

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

# ========================================================================
# Клас MobileAppInterface – модуль мобільного додатку для віддаленого керування
# ========================================================================
class MobileAppInterface:
# Ініціалізація інтерфейсу мобільного додатку
    def __init__(self):
# Список сповіщень для відображення на мобільному додатку
        self.notifications = []
# Черга для обробки дій користувача
        self.user_action_queue = queue.Queue()
# Логування ініціалізації мобільного інтерфейсу
        logging.debug("MobileAppInterface ініціалізовано")

# Метод для відображення головного меню мобільного додатку
    def display_menu(self):
# Логування відображення меню
        logging.debug("Відображення меню мобільного додатку")
# Вивід меню користувачу
        print("=== Mobile App Interface ===")
        print("1. Переглянути сповіщення")
        print("2. Надіслати запит на керування")
        print("3. Оновити дані системи")
        print("4. Вихід")

# Метод для обробки введення користувача
    def handle_user_input(self):
# Безперервний цикл обробки дій користувача
        while True:
            self.display_menu()
            choice = input("Виберіть опцію: ")
            logging.debug("Користувач вибрав: %s", choice)
            if choice == '1':
                self.show_notifications()
            elif choice == '2':
                self.send_control_request()
            elif choice == '3':
                self.refresh_system_data()
            elif choice == '4':
                logging.debug("Користувач вийшов з мобільного додатку")
                break
            else:
                print("Невірна опція. Спробуйте ще раз.")
            time.sleep(0.5)

# Метод для відображення сповіщень
    def show_notifications(self):
        if not self.notifications:
            print("Немає нових сповіщень.")
            logging.debug("Сповіщень немає")
```

КПІ - 2025

```
        else:
            for note in self.notifications:
                print("Сповіщення:", note)
            self.notifications.clear()
            logging.debug("Сповіщення відображено та очищено")

# Метод для симуляції надсилання запиту на керування системою
    def send_control_request(self):
        request_text = input("Введіть запит керування: ")
        logging.debug("Надіслано запит: %s", request_text)
        print("Запит на керування надіслано.")

# Метод для оновлення даних системи у мобільному додатку
    def refresh_system_data(self):
        logging.debug("Оновлення даних системи в мобільному додатку")
        print("Дані системи оновлено.")

# Метод для додавання сповіщення в мобільний додаток
    def add_notification(self, message):
        self.notifications.append(message)
        logging.debug("Додано сповіщення: %s", message)

# ============================================================================
# Клас CloudStorageModule – модуль хмарного зберігання даних
# ============================================================================
class CloudStorageModule:
# Ініціалізація модуля хмарного зберігання
    def __init__(self):
# Створення словника для зберігання даних
        self.storage = {}
# Логування ініціалізації хмарного зберігання
        logging.debug("CloudStorageModule ініціалізовано")

# Метод для завантаження даних у хмарне сховище
    def upload_data(self, data_id, data):
        logging.debug("Завантаження даних з ID: %s", data_id)
        time.sleep(random.uniform(0.1, 0.3))
        self.storage[data_id] = data
        logging.debug("Дані %s завантажено успішно", data_id)

# Метод для завантаження даних з хмарного сховища
    def download_data(self, data_id):
        logging.debug("Отримання даних з ID: %s", data_id)
        time.sleep(random.uniform(0.1, 0.2))
        data = self.storage.get(data_id, None)
        if data is None:
            logging.debug("Дані %s не знайдено", data_id)
        else:
            logging.debug("Дані %s завантажено", data_id)
        return data

# Метод для видалення даних з хмарного сховища
    def delete_data(self, data_id):
        logging.debug("Видалення даних з ID: %s", data_id)
        if data_id in self.storage:
            del self.storage[data_id]
            logging.debug("Дані %s видалено", data_id)
        else:
            logging.debug("Дані %s не знайдено для видалення", data_id)
```

КОПІЯ 2025

```
# Метод для отримання списку всіх збережених даних
    def list_data(self):
        logging.debug("Отримання списку даних з хмарного сховища")
        return list(self.storage.keys())

# ================================================================================
# Клас SecuritySystemIntegration - інтеграція з іншими системами безпеки
# ================================================================================
class SecuritySystemIntegration:
# Ініціалізація інтеграції з системами безпеки
    def __init__(self):
# Створення словника інтегрованих систем
        self.integrated_systems = {
            "alarm_system": False,
            "door_control": False,
            "fire_alarm": False
        }
# Логування ініціалізації інтеграції
        logging.debug("SecuritySystemIntegration ініціалізовано")

# Метод для підключення до системи сигналізації
    def connect_alarm_system(self):
        logging.debug("Підключення до системи сигналізації")
        time.sleep(random.uniform(0.1, 0.3))
        self.integrated_systems["alarm_system"] = True
        logging.debug("Система сигналізації підключена")

# Метод для підключення до системи контролю доступу
    def connect_door_control(self):
        logging.debug("Підключення до системи контролю доступу")
        time.sleep(random.uniform(0.1, 0.3))
        self.integrated_systems["door_control"] = True
        logging.debug("Система контролю доступу підключена")

# Метод для підключення до системи протипожежного сигналу
    def connect_fire_alarm(self):
        logging.debug("Підключення до системи протипожежного сигналу")
        time.sleep(random.uniform(0.1, 0.3))
        self.integrated_systems["fire_alarm"] = True
        logging.debug("Система протипожежного сигналу підключена")

# Метод для отримання статусу інтегрованих систем
    def get_system_status(self):
        logging.debug("Отримання статусу інтегрованих систем")
        status_report = {}
        for system, status in self.integrated_systems.items():
            status_report[system] = "Підключено" if status else "Відключено"
        logging.debug("Статус інтегрованих систем: %s", status_report)
        return status_report

# ================================================================================
# Клас NetworkOptimizationModule - модуль оптимізації мережевого з'єднання
# ================================================================================
class NetworkOptimizationModule:
# Ініціалізація модуля оптимізації мережевого з'єднання
    def __init__(self):
# Встановлення початкової пропускної здатності мережі (Mbps)
        self.current_bandwidth = 10.0
# Історія вимірювань пропускної здатності
        self.bandwidth_history = []
```

КПЗ-2025

```
# Логування ініціалізації оптимізації мережевого з'єднання
        logging.debug("NetworkOptimizationModule ініціалізовано з пропускною
здатністю %.2f Mbps", self.current_bandwidth)

# Метод для вимірювання поточної пропускної здатності
    def measure_bandwidth(self):
        fluctuation = random.uniform(-2.0, 2.0)
        self.current_bandwidth = max(1.0, self.current_bandwidth + fluctuation)
        self.bandwidth_history.append(self.current_bandwidth)
        logging.debug("Поточна пропускна здатність: %.2f Mbps",
self.current_bandwidth)
        return self.current_bandwidth

# Метод для оптимізації мережевого з'єднання
    def optimize_connection(self):
        logging.debug("Оптимізація мережевого з'єднання почата")
        current_bw = self.measure_bandwidth()
        if current_bw < 5.0:
            adjustment = random.uniform(1.0, 3.0)
            self.current_bandwidth += adjustment
            logging.debug("Збільшення пропускної здатності на %.2f Mbps",
adjustment)
        elif current_bw > 15.0:
            adjustment = random.uniform(1.0, 3.0)
            self.current_bandwidth -= adjustment
            logging.debug("Зменшення пропускної здатності на %.2f Mbps",
adjustment)
        else:
            logging.debug("Пропускна здатність у нормі")
        self.bandwidth_history.append(self.current_bandwidth)
        logging.debug("Оптимізація завершена, нова пропускна здатність: %.2f
Mbps", self.current_bandwidth)
        return self.current_bandwidth

# ====================================================================
# Клас FaultPredictionModule - модуль прогнозування збоїв у системі
# ====================================================================
class FaultPredictionModule:
# Ініціалізація модуля прогнозування збоїв
    def __init__(self):
# Історія зібраних показників роботи системи
        self.historical_data = []
# Встановлення порогового значення для прогнозування збоїв
        self.failure_threshold = 0.7
# Логування ініціалізації модуля прогнозування збоїв
        logging.debug("FaultPredictionModule ініціалізовано з порогом %.2f",
self.failure_threshold)

# Метод для збору даних про роботу системи
    def collect_data(self, metric):
        self.historical_data.append(metric)
        logging.debug("Зібрано метрику: %.2f", metric)

# Метод для аналізу даних та прогнозування збоїв
    def predict_failure(self):
        logging.debug("Початок прогнозування збоїв")
        if not self.historical_data:
            logging.debug("Дані для прогнозування відсутні")
            return False
        avg_metric = sum(self.historical_data) / len(self.historical_data)
        logging.debug("Середня метрика: %.2f", avg_metric)
        if avg_metric > self.failure_threshold:
            logging.debug("Прогноз збоїв: потенційний збій виявлено")
```

КПЗ_2025

```
            return True
        else:
            logging.debug("Прогноз збоїв: система стабільна")
            return False

# ========================================================================
# Головна функція інтеграції розширеного функціоналу
# ========================================================================
def main():
# Логування старту інтеграції розширеного функціоналу
    logging.debug("Старт інтеграції модулів розширеного функціоналу")

# Ініціалізація модуля мобільного додатку
    mobile_app = MobileAppInterface()

# Ініціалізація модуля хмарного зберігання
    cloud_storage = CloudStorageModule()

# Ініціалізація інтеграції з іншими системами безпеки
    security_integration = SecuritySystemIntegration()
    security_integration.connect_alarm_system()
    security_integration.connect_door_control()
    security_integration.connect_fire_alarm()

# Ініціалізація модуля оптимізації мережевого з'єднання
    network_optimization = NetworkOptimizationModule()

# Ініціалізація модуля прогнозування збоїв
    fault_prediction = FaultPredictionModule()

# Симуляція завантаження даних у хмарне сховище
    for i in range(1, 6):
        data_id = f"data_{i}"
        data_content = f"Відеоархів_{i}_даних"
        cloud_storage.upload_data(data_id, data_content)
        time.sleep(0.2)

# Симуляція оптимізації мережевого з'єднання
    for _ in range(5):
        network_optimization.optimize_connection()
        time.sleep(0.3)

# Симуляція збору даних для прогнозування збоїв
    for _ in range(10):
        metric = random.uniform(0.5, 1.0)
        fault_prediction.collect_data(metric)
        time.sleep(0.1)

# Перевірка прогнозування збоїв та надсилання сповіщення
    if fault_prediction.predict_failure():
        logging.debug("Попередження: прогнозовано збій системи")
        mobile_app.add_notification("Попередження: прогнозовано збій системи")
    else:
        logging.debug("Система стабільна, збій не прогнозується")
        mobile_app.add_notification("Система стабільна, збій не прогнозується")

# Запуск мобільного додатку в окремому потоці
    mobile_thread = threading.Thread(target=mobile_app.handle_user_input,
daemon=True)
    mobile_thread.start()
```

КПТВ-2025

```
# Функція для періодичної перевірки даних у хмарному сховищі
    def periodic_cloud_check():
        while True:
            stored_data = cloud_storage.list_data()
            logging.debug("Перевірка хмарного сховища: %s", stored_data)
            time.sleep(5)

# Запуск перевірки хмарного сховища в окремому потоці
    cloud_thread = threading.Thread(target=periodic_cloud_check, daemon=True)
    cloud_thread.start()

# Основний цикл симуляції роботи інтегрованих модулів
    simulation_duration = 30
    start_time = time.time()
    while time.time() - start_time < simulation_duration:
        network_optimization.measure_bandwidth()
        metric = random.uniform(0.5, 1.0)
        fault_prediction.collect_data(metric)
        time.sleep(1)

# Завершення симуляції розширеного функціоналу
    logging.debug("Симуляція розширеного функціоналу завершена")

# ========================================================================
# Точка входу в програму
# ========================================================================
if __name__ == "__main__":
    main()
```

КбПЗ_2025