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 мережевих сховищ на основі технології NVR”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Дьяченко Р.В.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *123 "*Комп'ютерна інженерія*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

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

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

Дьяченку Роману Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи мережевих сховищ на основі технології NVR*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 48-02 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 мережевих сховищ на основі технології NVR*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Дьяченко Р.В.
(прізвище та ініціали)

АНОТАЦІЯ

Дьяченко Р.В. Програмне забезпечення системи мережевих сховищ на основі технології NVR. 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 мережевих сховищ на основі технології NVR.

Метою розробки є програмне забезпечення системи мережевих сховищ на основі технології NVR.

Результат роботи – програмна реалізація системи мережевих сховищ на основі технології NVR.

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, система мережевих сховищ, NVR

ABSTRACT

Dyachenko R.V. Software for a network storage system based on NVR 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 that is intended for a network storage system based on NVR technology.

The purpose of the development is software for a network storage system based on NVR technology.

The result of the work is a software implementation of a network storage system based on NVR 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, network storage system, NVR

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БПД	–	бездротова передача даних
ПЗ	–	програмне забезпечення
СЗД	–	системи зберігання даних
СПД	–	системи передачі даних
ACK	–	повідомлення підтвердження прийому
ARQ	–	протокол повторної передачі даних
BPSK	–	Binary phase-shift keying
FFD	–	повнофункціональний пристрій
GFSK	–	Gaussian frequency-shift keying
MAC	–	шар механізму доступу
NACK	–	повідомлення непідтвердження прийому
NVR	–	Network Video Recorder, або мережний відеореєстратор
OSI	–	мережна модель
P2P	–	однорангові мережі
PAN	–	персональна мережа
PPS	–	Portable Protocol Stack
RFD	–	пристрій з полегшеними функціями
TDMA	–	часовий поділ
Wi-Fi	–	бездротова технологія

ВСТУП

Актуальність теми. NVR (мережевий відеореєстратор) – це незамінний пристрій у системах відеоспостереження багатьох підприємств, який служить для зберігання та керування відеоматеріалами, знятими IP-камерами. Завдяки високій роздільній здатності та ефективним можливостям стиснення відео, цей пристрій дозволяє користувачам чітко та чітко переглядати відео, не витрачаючи при цьому надмірного місця.

NVR (мережевий відеореєстратор) – це мережевий пристрій запису, який зберігає та керує відео, знятими з IP-камер через підключення до Інтернету. Цей пристрій працює, підключаючись безпосередньо до IP-мережі для захоплення та зберігання відео на жорсткому диску, що дозволяє користувачам легко керувати ними або переглядати їх за потреби.

NVR – це система, яка записує та зберігає захоплені відео на жорсткому диску, хмарному сховищі або інших пристроях великої ємності. Зазвичай цей пристрій підключається до цифрових IP-камер для створення комплексної, безпечної системи спостереження, що гарантує безпеку користувачів. Завдяки своїй здатності розширювати діапазон моніторингу без зниження якості, NVR став найкращим рішенням безпеки для багатьох закладів, фінансових установ та підприємств, які бажають дистанційно контролювати товари.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи мережевих сховищ на основі технології NVR.

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

– Огляд існуючих систем мережевих сховищ на основі технології NVR.

– Дослідження системи мережевих сховищ на основі технології NVR.

– Програмна реалізація системи мережевих сховищ на основі технології NVR.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі мережевих сховищ на основі технології NVR.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи мережевих сховищ на основі технології NVR, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації мережних сховищ на основі технології NVR.

Робота відеореєстраторів з мережними NAS сховищами (NAS + DVR/NVR/HVR)

Як відомо, будь-який сучасний відеореєстратор є вузькоспеціалізованим комп'ютером, а його можливості обмежені й оптимізовані під конкретні завдання. Тобто щось додати до функціонала реєстратора або важко, або взагалі неможливо.

З постійним збільшенням розв'язної здатності відеокамер росте й потреба в більшій сумарній ємності накопичувачів. Тобто перша найпоширеніша проблема – це недостатня глибина архіву, що як правило лімітується кількістю підтримуваних накопичувачів і їхньою ємністю.

Другою проблемою, пов'язаної зі зберіганням даних, є забезпечення дзеркального запису, найчастіше навіть на пристрій, що територіально перебуває в іншому місці. За статистикою жорсткий диск є одним із самих уразливих місць у системі відеоспостереження. Основний накопичувач може банально вийти з ладу, може бути викрадений, відформатований, а також навмисне зіпсований. Тому резервування відеоархіву – це досить корисна річ у багатьох випадках.

У технічних характеристиках більшості сучасних професійних відеореєстраторів заявлена підтримка NAS сховищ або протоколу NFS. Цей функціонал дозволяє з легкістю вирішити вищеописані завдання.

Якщо звернутися до «Вікіпедії», то NAS (англ. Network Attached Storage) – це мережна система зберігання даних, мережне сховище. Являє собою комп'ютер, підключений до мережі й призначений для надання сервісів зберігання даних іншим пристроям.

Простіше говорячи, це мережний зовнішній жорсткий диск, що може бути використаний як основний накопичувач, а також як засіб для збільшення ємності відеоархіву або його резервування.

На сьогоднішній день готове NAS-сховище можна придбати в будь-якому великому магазині, що торгує комп'ютерною технікою. А також якщо в коморі завалявся системний блок із застарілим залізом, те його цілком реально задіяти для створення саморобного мережного сховища на базі безкоштовної операційної системи типу NAS4Free або FreeNAS.

При виборі мережного сховища під певні завдання звичайно в першу чергу керуються кількістю дисків, що підключаються.

Звичайно до реєстратора можна підключити кілька мережних сховищ одночасно для прямого запису або для резервування даних, що зберігаються на жорсткому диску відеореєстратора. При цьому NAS може перебувати як в одній локальній мережі з реєстратором прямо на об'єкті, так і віддалено, наприклад, в іншому місті.

Наприклад, якщо до одного реєстратора підключити 8 мережних сховищ із максимальною кількістю жорстких дисків 24, то при обліку, що диски мають ємність по 4 Тб кожний, сумарний архів буде досягати 768 Тб. Тобто при побудові такої системи відеоспостереження ємність відеоархіву практично не обмежена.

NAS-сховище або IP-реєстратор? (NAS + IPC vs. NAS + NVR/DVR/HVR)

У лінійках професійного встаткування багатьох виробників не тільки реєстратори мають підтримку роботи з мережними NAS-сховищами, але й IP-камери. Виходячи із цього, виникає питання про доцільність покупки спеціалізованого відеореєстратора, коли можна використовувати універсальний NAS. Але не всі так просто, як здається. Навіть найпростіше мережне сховище на 1 жорсткий диск коштує дорожче 4-канальні реєстратори, у якого буде ще ряд додаткових переваг. Тому як основний засіб зберігання відеоархіву використовувати NAS не зовсім доцільно.

Дана функція в IP-камер актуальна й затребувана у випадку, якщо сховище вже є й використовується для зберігання, наприклад, медіаконтенту в домашніх умовах. І отут виникає думка поставити пару IP-камер. У цьому випадку дійсно зовсім не обов'язково здобувати відеореєстратор і можна обійтися вже наявним пристроєм.

Виходячи із усього вищеописаного, можна зробити виводи, що мережні NAS сховища досить просто вирішують ряд завдань, пов'язаних зі зберіганням записів, при роботі з відеореєстраторами. За допомогою NAS можна розширити ємність відеоархіву реєстратора, а також підвищити його надійність шляхом резервування даних.

Система NAS + IP-камера має місце бути тільки тоді, коли сховище вже є й використовується не тільки для зберігання записів з камер, але й для зберігання іншої інформації. Тобто якщо вибирати між покупкою спеціалізованого реєстратора й NAS тільки під систему відеоспостереження, те мережні сховища програють реєстраторам як за ціною, так і по функціоналу.

1.2 Область застосування

Областю застосування є хмарне відеоспостереження. Хмарне відеоспостереження – це тип відеобезпеки, в якому відзнятий матеріал зберігається віддалено в хмарі, а не на локальному сервері чи пристрої.

Хмарні системи поєднують функціональність традиційної системи камер безпеки з потужністю та зручністю хмарних технологій. Вони надають багато переваг, яких немає в традиційних системах, зокрема вбудований віддалений доступ, централізоване керування та значне зменшення обсягу системного обладнання.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи мережевих сховищ на основі технології NVR, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Що таке NAS NVR? Це NAS з можливостями відеореєстратора або NVR з можливостями мережного сховища? Що за конвергенція відбувається у виробників комп'ютерного встаткування? Яка елементна база в цього класу пристроїв?

На ці й деякі інші, цілком звичайні для споживача відеореєстраторів питання покликане відповісти даний огляд.

Для початку згадаємо класифікацію відеореєстраторів для IP-відеокамер (NVR – Network Video Recorder, або мережний відеореєстратор). Принципово NVR є якимось мережним сервером, здатним записувати медіапотоки з мережних відеокамер. Тому що IP-відеоспостереження опирається на мережні стандарти, прийняті на комп'ютерному ринку, сполучення NVR і IP-камер у єдиній мережі зовсім не є проблемою – і два класи пристроїв, здатних працювати як відеореєстратори, приходять до нас із комп'ютерного ринку.

NAS NVR

На першому рівні поставимо NAS NVR. Мережні сховища (NAS – Network Attached Storage), навіть не маючи спеціальне програмне забезпечення, можуть служити як сховище відеоархіву для IP-камер прямо, тому що багато камер можуть писати в найкращому разі в NFS-папку, а в гіршому (ненадійно) – в FTP-папку. Звичайно, зручність такої системи досить сумнівно, оскільки архів прийде читати, просто заходячи в ці папки. Однак виробники NAS швидко відчули перспективність суміжного ринку й розробили спеціальне програмне забезпечення, тим або іншим способом розширювальної можливості мережних

сховищ як відеореєстраторів. Цьому сприяє й постійний ріст частоти процесорів і обсягу оперативної пам'яті даних пристроїв. Самі потужні NAS давно являють собою комп'ютерні платформи зі спеціалізованим програмним забезпеченням.

Non-PC NVR

Другий, середній клас іде до нас із ринку традиційного відеоспостереження – non-PC NVR. Вони працюють на тих же платформах, DSP-I ARM-процесорах, що й звичайні DVR, до яких ми давно звикли. Фактично їхньою головною відмінністю є те, що не потрібно компресувати відео, отримане з аналогових камер, а потрібно, навпаки, декодувати його – так, як традиційні DVR роблять це під час перегляду архіву. Із цим зв'язані звичайно більше тверді обмеження по кількості каналів і дозволу, з яким працюють non-PC

NVR – адже вони в більшості випадків (у відмінність він NAS NVR) показують відео на підключені безпосередньо до них моніторах.

PC NVR

Ну й третя платформа, що прийшла також з комп'ютерного ринку, із самого початку була основою для побудови IP-відеоспостереження. Це, зрозуміло, комп'ютери.

Самі гнучкі й максимальні можливості, що мають, як по розширенню, так і по продуктивності, PC NVR можуть бути як Net-Top (невеликий комп'ютер на базі найпростіших і дешевих процесорів типу Atom), так і багатопроцесорними серверами з операційною системою на ваш вибір. Та й програмне забезпечення для роботи з IP-камерами, втім, ви теж вільні вибирати. Причому зараз ринок програмного забезпечення для IP-відеоспостереження переходить у стадію, коли воно вже не є ціноутворюючим у системі – є й безкоштовні версії, цілком здатні вирішувати невеликі завдання, є й потужні інтегровані комплекси, але знову ж їхня вартість розчиниться в ціні загального кошторису комп'ютерного й мережного встаткування.

Проведений нами огляд цього разу навіть не порівняльний, це скоріше якийсь зріз ринку, що демонструє майже весь арсенал доступних NAS NVR – від

SOHO-рішень до корпоративних. Метою є не "зштовхнути чолами" пристрою для різних цінових ніш, а показати, які рішення якими можливостями володіють. Ну й уперше даний огляд доповнений відеоматеріалами, які дозволяють побачити пристрої в роботі і їхні деякі особливості.

QNAP VS-5020 і VS-8040

Від QNAP ми включили в огляд два пристрої, хоча програмне забезпечення для них уніфіковано.

– QNAP VS-5020 – NAS NVR на 5 жорстких дисків і з можливістю запису до 20 IP-камер;

– QNAP VS-8040 – пристрій на 8 дисків з можливістю запису до 40 IP-відеокамер. Воно є самим потужним в огляді й розраховано на корпоративне застосування.

Рисунок 2.1 – QNAP VS-5020 і VS-8040

Обоє мають по два гігабітних мережних інтерфейсу, LCD-дисплей для швидкого налаштування й керування функціями реєстратора, USB-роз’єм, VGA-

порт, зовнішні порти eSATA. Охолодження виробляється за рахунок вентиляторів.

Робота із пристроєм

Для знаходження пристрою в мережі використовується прикладений у комплекті розвідувач. QNAP, як уже сказано, підтримує до 20 і 40 IP-камер від великої кількості виробників.

Рисунок 2.2 – Робота із пристроєм

Пристрій має потужну апаратну базу (Intel Celeron для 5020 і Core2Duo для 8040), що забезпечує високу продуктивність – як запису великої кількості камер, так і доступу до них значного числа користувачів.

Пристрої також мають широкі можливості забезпечення безпеки:

– керування правами доступу;

– можливість блокування IP-адрес (на базі політик);

– автоматичне завершення сесії – якщо пристроєм не користуватися протягом 10 хв.;

– запис всіх підключень користувачів;

– можливість призначати права доступу до кожної камери індивідуально для кожного користувача (до 32 користувачів);

– докладний журнал подій (детальний запис системних подій, помилок жорстких дисків, підключень, стану джерела безперебійного живлення, мережних служб, доступу до даних, активності користувачів).

VS-5020 підтримує кілька конфігурацій дискового масиву: RAID (0, 1, 5, 6, 5/6 + Spare), однодискові томи й JBOD. Зрозуміло, що RAID (особливо 5-6) дає можливість збільшення обсягу зберігання з одночасним підвищенням надійності зберігання, при цьому жорсткі диски можуть бути замінені без вимикання пристрою. Підтримується збільшення рівня RAID-Масиву в "гарячому" режимі, без зупинки роботи додатків і пристрою.

Переваги:

– Потужна апаратна база.

– Велика кількість підтримуваних виробників IP-камер.

– Широкі можливості по організації RAID.

Недоліки:

Пошук IP-камер працює некоректно.

Muuo NVRmini IMV-4160S

Nuuo NVRmini NV-4160S – це мережний відеореєстратор, представлений у компактному пластиковому корпусі з можливістю установки до 4 жорстких

дисків. Модель підтримує запис із 16 мережних відеокамер з досить широким вибором виробників. На задній панелі даної моделі розташовані два USB-порти, вентилятор для охолодження дисків, роз'єм eSATA, мережний роз'єм й кнопка включення. На передній панелі – індикатори живлення, роботи жорстких дисків і мережі. Може непогано підійти для роботи в невеликому офісі.

Рисунок 2.3 – Muuo NVRmini IMV-4160S

Робота із пристроєм

В NVRmini NV-4160S, як і майже у всіх інших NVR, є присутня функція автоматичного пошуку підтримуваних камер у локальній мережі, що сильно спрощує пошук і установку камер. Процес ініціалізації відеоспостереження не займає багато часу: після того як підходящі камери знайдені, запис починається

автоматично. У цілому інтерфейс дружелюбний до користувача й простий в експлуатації.

Система має чотири рівні безпеки – адміністратор, досвідчений користувач, користувач і гість. Через підключений віддалено по мережі CMS-клієнт або за допомогою браузера можна переглядати до 16 каналів як живого, так і архівного відео (64 каналу з декількох пристроїв). Крім можливості запису на убудований масив з 4 дисків, NV-4160S здатний підтримувати зовнішній масив – до 16 Тбайт сумарно. Компанія Nuuo була колись відома як розроблювач програмного забезпечення й змогла дати NVRmini додаткові можливості – наприклад, він володіє 5 типами інтелектуального пошуку. Крім того, підтримується віддалене резервне копіювання відео й POS-транзакцій (да-да, цей єдиний пристрій, що підтримує зовнішні інтеграції).

Рисунок 2.4 – Робота із пристроєм

Переваги:

– Велика кількість підтримуваних виробників IP-камер.
– Підтримка інтеграції з POS-системами.
– Гарне додаткове програмне забезпечення.
– Можливості інтелектуального пошуку.

Недоліки:

Особливих не відзначено.

D-Link DNS 722-4

Мережний відеореєстратор від компанії D-Link представлений у компактному пластиковому корпусі з можливістю установки двох жорстких дисків у спеціальні відсіки. (Диски просто вставляються у відсіки – спеціальне оснащення не потрібна.) На задній панелі розташовані USB-порт, вентилятор для охолодження дисків, кнопка Reset і мережний роз'єм. На передній панелі перебуває кнопка включення. Передня панель знімна, під нею – слоти для жорстких дисків.

Рисунок 2.5 – D-Link DNS 722-4

Робота із пристроєм

Модель DNS 722-4 підтримує запис одночасно з 4 мережних відеокамер. Як і в інших NVR, тут є присутня функція автоматичного пошуку підтримуваних камер у локальній мережі. Процес ініціалізації відеоспостереження не займає багато часу: після того як підходящі камери знайдені, запис починається автоматично. Інтерфейс у цілому зручний і не викликає питань, особливо якщо ви хоча б раз користувалися пристроями NAS.

Є присутня функція автоматичного перезапису у випадку переповненого простору на жорсткому диску (Overwrite), але відсутній детектор руху камер в убудованому ПЗ. Тому що можна встановити тільки два диски, варіанти створення масивів – тільки RAID 0, 1 і JBOD. D-Link DNS 722-4 прекрасно підійде для використання в невеликому офісі або навіть у приватному будинку.

Рисунок 2.6 – Робота із пристроєм

Переваги:

– Найнижча ціна серед представлених пристроїв.

– Зручний інтерфейс.

Недоліки:

Невелике число підтримуваних виробників IP-камер.

Synology 1511 +

Модель Synology 1511 + представлена в металевому корпусі із пластиковою передньою панеллю. На передній панелі перебувають 5 вертикальних відсіків, куди можна встановити жорсткі диски, кнопка включення, індикатори мережі й живлення. На задній панелі перебувають чотири роз'єми USB, два – eSATA для підключення додаткових дисків, VGA-порт і два мережних гігабітних порти. Охолодження виробляється двома вентиляторами. Відеореєстратор здатний вести запис із 20 камер. Може підійти для роботи в невеликому офісі.

Рисунок 2.7 – Synology 1511 +

Робота із пристроєм

Synology 1511 + має багатий інтерфейс із гарним вибором інструментів. Розроблювач особливо відзначає своє операційне середовище верхнього рівня DSM 3.0, що має віконний інтерфейс, робочий стіл і можливість самостійної установки додаткових модулів (їхня кількість постійно росте).

Адміністратор може набудовувати доступ до системи для гостей і користувачів, регулюючи при цьому ступінь їхніх повноважень. Даючи користувачам можливість запису з відеокамер, Synology не обмежує їх у використанні NAS – його можна експлуатувати і як сервер для зберігання будь-яких даних, мультимедіа, а за допомогою сервісу Mail Station реєстратор Synology можна зробити поштовим сервером.

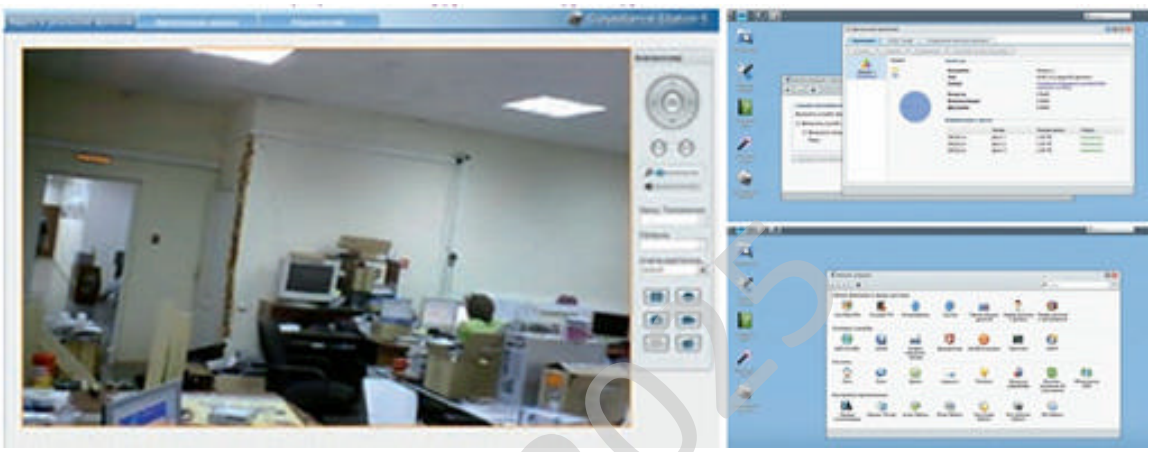

Рисунок 2.8 – Робота із пристроєм

Synology 1511 + підтримує 35 виробників і більше 600 моделей IP-камер і лідирує по цьому показнику серед представлених відеореєстраторів.

Фактично, здобуваючи Synology, ви можете використовувати його як "комбайн" для невеликого офісу (як сервер пошти, сховища й сервера системи відеоспостереження) – непогана економія (хоча на безпеці краще не заощаджувати й використовувати сервер для відеоспостереження монопольно).

Переваги:

– Унікальна програмна оболонка DSM 3.0.

– Можливість розширення програмними модулями.

– Найбільша кількість підтримуваних виробників IP-камер.

Недоліки:

Немає.

Thecus N4200

Відеореєстратор від компанії Thecus представлений у компактному металевому корпусі з передньою панеллю із пластику. На передній панелі перебувають дисплей, два USB-порти, кнопка включення, чотири кнопки для керування меню, а саме: пересування на пункт уперед, пересування на пункт назад, уведення й ESC. Крім того, на передній панелі перебувають чотири слота для жорстких дисків і індикатори мережі живлення й стани жорстких дисків. На задній панелі – роз'єм для акумулятора, роз'єм WAN і LAN, чотири USB-порти й два порти eSATA.

Рисунок 2.9 – Thecus N4200

Робота із пристроєм

Робота з даним відеореєстратором сильно утруднена через непристосований для відеоспостереження інтерфейсу. Для того щоб приступитися до роботи з відеокамерами, необхідно для початку додати й установити додатковий модуль IP Cam. Після чого варто додавати камери вручну, тому що відсутній автоматичний пошук підтримуваних моделей.

Проте в інтерфейсі Thecus N4200 є присутнім велика кількість різноманітних інструментів. Дана модель також здатна до автоматичного резервного копіювання й підтримує RAID 10.

Особливість моделі Thecus N4200 полягає в наявності модуля резервного живлення, що дає пристрою тимчасовий захист при збоях у зовнішнім живленні.

Рисунок 2.10 – Робота із пристроєм

Відзначимо, що даний пристрій є більше накопичувачем NAS, ніж відеореєстратором. Однак NVR на 4 диски й 20 камер усього за 800 доларів ще варто пошукати!

Переваги:

– Гарна технічна оснащеність.

– Наявність убудованого акумулятора для резервного живлення.

– Невисока ціна.

– Велика кількість налаштувань.

Недоліки:

Програмне забезпечення пристрою мало пристосовано для відеоспостереження.

NAS NVR мають право бути

Саме головне, що варто відзначити: ринок IP-відеоспостереження став уже настільки цікавий, що виробники NAS не просто роблять додаткове програмне забезпечення для своїх пристроїв, але й виділяють їх в окремі лінійки й позиціонують саме як NVR-відеореєстратори. Ці пристрої відрізняються фактично програмним забезпеченням – одне орієнтоване на ринок сховищ і має більше відповідного функціонала, а друге – на ринок відеоспостереження.

Навіть D-Link у прицілі на відеоспостереження випускає спеціалізовану модель, що відрізняється й дизайном, і начинкою. Synology і QNAP мають ясну орієнтацію на надання повної лінійки NAS для відеоспостереження. А QNAP виділяє частина NAS в окрему лінійку, повністю переорієнтувавши їх на IP-камери й роботу в складі системи відеоспостереження.

Друга тенденція NAS NVR, властиво, трохи ламає класифікацію, наведену спочатку, – із всіх представлених NAS тільки D-Link не базується на x 86-платформі. Скрізь на цьому ринку тенденція така, що, починаючи вже з 2-дискових пристроїв, їх починають будувати на платформах Intel Atom, а могутніші – і на Core2Duo. І нехай туди встановлена "спеціалізована" Linux, сказати, що це не PC-based NVR, уже досить складно. Звичайно, є й аргумент, що NAS NVR не мають можливості відображення на підключені до них безпосередньо монітори, але й тут стали з'являтися моделі – QNAP заявив про таку лінійку, Synology у всіх "атомних" рішеннях має VGA-вихід, використання якого, видимо, – справа недалекого майбутнього. І хоча non-PC NVR уже оснащені HDMI-виходами, як видно, битва між ними не за горами й споживач від її тільки виграє.

Характеристики

Що можна сказати про характеристики розглянутих NAS NVR? Представлені моделі підтримують від 4 до 40 каналів – тобто рішення на базі NAS NVR перекриває всі цінові сегменти й може використовуватися як на об'єктах типу маленького офісу, де відеореєстратор може бути навіть сполучений з локальним сховищем, до рішень типу QNAP 8040, розрахованих на серйозний корпоративний рівень.

Програмне забезпечення кожного виробника має свої особливості.

– D-Link – це нормальний міцний SOHO-сегмент; жаль, що лінійка складається всього з одного пристрою.

– Thecus – потрібно ще попрацювати над прошиванням для орієнтації саме на відеоспостереження, хоча він і виграє за ціною.

– QNAP – має найбільшу лінійку рішень для відеоспостереження, і його програмне забезпечення сполучить все необхідне для роботи як з убудованим сховищем для створення надійних масивів аж до RAID 6, так і з камерами і їхніми архівами.

– Synology – виділяється дизайном свого DSM 3.0 – убудованого програмного забезпечення, що не тільки красиво, але і являє собою операційну систему верхнього рівня з віконним інтерфейсом, робочим столом і навіть дозволяє встановлювати додаткові програмні модулі, у тому числі й інших компаніях.

Даремно порівнювати пристрої з дозволу записуваних відеокамер, тому що "їм однаково" – відображенням вони не займаються. Значення, як і в будь-якому NVR, має сумарний потік на пристрій, що виражається в мегабітах або мегабайтах. А тут істотну роль грає продуктивність дискової підсистеми – вона важливіше навіть, ніж швидкість і кількість мережних інтерфейсів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктно-орієнтована 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:

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

– Об’єктно-орієнтована: 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 веб-сайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веб-розробці, 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.

– 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.

– 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 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.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веб-розробника.

– 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 мережевих сховищ на основі технології NVR.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Щоб забезпечити оптимальну роботу відеореєстратора в системі камер, виробники зазвичай розробляють пристрій з трьома основними компонентами:

– IP-камера: Зазвичай відеореєстратор поєднується з IP-камерами (кулеподібного або купольного типу) для покращення якості зображення.

– Відеореєстратор: це критично важлива частина системи камер безпеки, здатна приймати відеозаписи через мережеве з’єднання та зберігати їх для подальшого перегляду користувачами. Наразі відеореєстратори часто оснащені кількома портами Ethernet, що дозволяє користувачам підключати кілька камер одночасно (наприклад, 8-канальний відеореєстратор може підтримувати до 8 камер).

– Мережа Ethernet: У наш час IP-камери безпеки з відеореєстраторами часто використовують дротове або бездротове з'єднання. У випадку дротових систем камери підключаються до відеореєстратора через мережу Ethernet.

Переваги NVR у системах відеоспостереження

Порівняно з традиційними відеореєстраторами, відеореєстратори пропонують багато переваг, підвищуючи ефективність спостереження та управління безпекою. Отже, які переваги відеореєстраторів?

Підтримка багатьох каналів

Більшість мережевих відеореєстраторів (NVR), представлених на ринку сьогодні, можуть одночасно обробляти відеопотоки з кількох камер. Залежно від моделі та бренду камери, кількість каналів, що підтримуються NVR, може коливатися від кількох до десятків, що допомагає компаніям легко керувати великомасштабними системами безпеки.

Висока роздільна здатність

Сучасні відеореєстратори зазвичай записують відео у високій роздільній здатності, такій як 720p, 1080p та 4K Ultra HD, що дозволяє користувачам переглядати чіткий матеріал навіть за умов слабкого освітлення.

Стиснення камери

Однією з найпомітніших функцій пристроїв NVR є стиснення камери, яке зменшує розмір файлу відеоматеріалів, зберігаючи при цьому високу якість зображення. Поширені формати стиснення відео включають:

– H.264: Найпопулярніший формат стиснення відео для IP-камер, що дозволяє користувачам переглядати чіткі кадри, не витрачаючи забагато пропускної здатності та місця для зберігання.

– H.265: Розширена версія, яка пропонує вдвічі вищу ефективність кодування відео порівняно з H.264.

Відеоаналітика

Фактично, передові відеореєстратори також пропонують функції відеоаналітики, такі як підрахунок людей, ідентифікація транспортних засобів, розпізнавання номерних знаків та розпізнавання облич. Крім того, ці пристрої можуть автоматично виявляти незвичайні рухи або несанкціоновані вторгнення.

Ці відеореєстратори можна підключити до систем сигналізації або складніших пристроїв спостереження. Виявивши аномальні ознаки, відеореєстратор увімкне тривогу, надішле сигнал користувачеві та почне запис поточної події.

Віддалений доступ та управління

Системи відеоспостереження NVR зазвичай дозволяють користувачам отримувати доступ до даних та керувати відео віддалено через локальну мережу або підключення до Інтернету. Це дозволяє окремим особам та власникам бізнесу легко контролювати потоки камер або отримувати доступ до них будь-коли та будь-де, забезпечуючи при цьому ефективне спостереження.

Інтеграція з іншими системами

Окрім вищезазначених переваг, деякі відеореєстратори також можуть інтегруватися з іншими системами безпеки, такими як пристрої контролю доступу та сигналізації виявлення вторгнень. Це дозволяє користувачам створювати комплексну та ефективну систему спостереження, яка максимізує безпеку території.

Недоліки NVR у системах відеоспостереження

Незважаючи на численні переваги, пристрої NVR у системах відеоспостереження все ще мають деякі обмеження. Отже, які ж недоліки NVR?

– Висока вартість: Системи відеоспостереження NVR пропонують високоякісне відео та просте дистанційне керування, що часто призводить до вищої ціни.

– Обмежена сумісність з IP-камерами: Не всі IP-камери сумісні з системами NVR, тому власникам бізнесу необхідно ретельно перевіряти та вибирати пристрої, щоб переконатися, що вони купують правильний продукт.

– Потенційна втрата сигналу: Коли мережеве з'єднання втрачається, вся система спостереження (включно з поточним записом) буде перервана, що призведе до збоїв у роботі моніторингу безпеки.

– Потрібне підключення до Інтернету: Загалом, для повного використання функцій моніторингу безпеки системам відеоспостереження потрібне підключення до Інтернету, що може обмежувати місця встановлення та збільшувати щомісячні витрати на електроенергію.

Як працює система відеореєстратора (NVR)?

Зазвичай системи відеореєстраторів NVR працюють за таким принципом:

– IP-камери підключаються через проміжний комутатор, який потім веде до відеореєстратора (NVR). Потім відеореєстратор підключається до маршрутизатора, концентратора або сервера, а потім до брандмауера для підвищення безпеки даних зображення перед підключенням до Інтернету.

– Окрім цього налаштування, користувачі також можуть підключити

відеореєстратор безпосередньо до монітора або централізованої станції моніторингу через Інтернет або програмне забезпечення для керування.

Як переглядати потоки з камери та записані матеріали?

Наразі зображення та відео, зняті IP-камерами з відеореєстраторами (NVR), зазвичай зберігаються та переглядаються віддалено, онлайн або офлайн, за допомогою програмного забезпечення для керування. Це програмне забезпечення зазвичай постачається в комплекті з NVR, що дозволяє користувачам легко налаштовувати, підключатися та переглядати відеоматеріали. Однак, для розширених функцій, таких як сповіщення в режимі реального часу або інструменти пошуку, компаніям слід обрати систему керування відео (VMS).

Розширені системи керування відео (VMS) або інтелектуальні системи керування відео (IVMS) можуть бути або в комплекті з відеореєстратором, або служити як окремі інструменти. Обидві системи розроблені з розширеними, корисними функціями, які дозволяють користувачам повною мірою використовувати обладнання камери для моніторингу безпеки навколишньої території.

Поради щодо вибору правильного відеореєстратора

Щоб вибрати правильний відеореєстратор (NVR) для ваших потреб, врахуйте такі фактори:

– Кількість камер: Визначте, скільки камер ви плануєте використовувати у своїй системі безпеки. Виберіть відеореєстратор з достатньою кількістю каналів, щоб задовольнити як поточні, так і майбутні потреби.

– Роздільна здатність відео: Виберіть відеореєстратор, який може обробляти потрібну вам роздільну здатність відео.

– Місткість сховища: Якщо вам потрібне відео високої роздільної здатності, висока частота кадрів і точний запис, переконайтеся, що у вас достатньо місця для зберігання. Деякі відеореєстратори також пропонують хмарне сховище як опцію.

– Функції: Шукайте відеореєстратори з відеоаналітикою, віддаленим

доступом та можливостями інтеграції для ефективного управління та спостереження.

– Бюджет: Виберіть відеореєстратор відповідно до вашого бюджету, враховуючи бренд, функції та кількість каналів.

Примітки щодо систем відеореєстраторів (NVR)

Розуміння характеристик та принципів роботи систем відеореєстраторів (NVR) допоможе вам використовувати систему спостереження зручніше та ефективніше. Однак, щоб забезпечити максимальну потужність системи відеореєстраторів та комплексний моніторинг безпеки, користувачам слід звернути увагу на такі ключові моменти:

Як підключити IP-камери до відеореєстратора

Наразі існує два способи підключення IP-камер до відеореєстратора: за допомогою технології Power over Ethernet (PoE) або WiFi:

– Підключення PoE: Якщо ви використовуєте цю технологію, вам потрібно придбати відеореєстратор PoE з відповідним портом живлення. У цьому випадку просто підключіть IP-камеру до відеореєстратора за допомогою мережевого кабелю (Cat5e або Cat6).

– Бездротове підключення: Цей метод відносно простий; вам просто потрібно використовувати бездротовий відеореєстратор та IP-камери з підтримкою Wi-Fi.

Системи відеореєстраторів NVR не потребують більшої пропускної здатності, ніж відеореєстратори

Багато людей вважають, що системи NVR, оскільки вони зберігають дані через мережу, потребують більшої пропускної здатності. Однак це не так. Високе використання пропускної здатності мережі відбувається лише тоді, коли користувачі отримують доступ до пристрою віддалено через програмне забезпечення для керування або зберігають дані в хмарі.

Використовуйте дводіапазонні IP-камери під час вибору бездротового відеореєстратора

Фактично, IP-камери з підтримкою дводіапазонного режиму пропонують більшу дальність з'єднання з відеореєстратором. Це допомагає покращити моніторинг безпеки в межах допустимого діапазону сигналу WiFi, а також мінімізує ризик перешкод або відключення мережі.

3.2 Розробка структурної схеми

Керуйте безпекою за допомогою хмарного програмного забезпечення для відеоспостереження, яке доступне, коли вам це потрібно, безпечне та може розвиватися разом з вашим бізнесом.

Керування вашими відеосистемами на віддалених об'єктах має бути простим. Попрощайтеся з обслуговуванням ресурсоємних систем. Розроблюване у дані роботі програмне забезпечення – це послуга відеоспостереження як послуга (VSaaS) на основі підписки, яка спрощує роботу та максимізує ефективність вашого бізнесу. Легко масштабуйте систему на кількох локаціях та отримуйте доступ до свого відео з будь-якого місця. Хмарне програмне забезпечення дозволяє забезпечити централізовану та безпечну роботу.

Швидке масштабування, легка адаптація

Розширюйте та адаптуйте свої системи безпеки завдяки хмарній масштабованості, відкритим інтеграціям та гнучкості використання існуючих камер.

Спростіть та посиліть безпеку

Покращте операції за допомогою інтуїтивно зрозумілої платформи, яка є безпечною, простою у використанні та завжди оновленою, без потреби у складному обслуговуванні чи обладнанні.

Приймайте розумніші рішення

Використовуйте аналітику на базі штучного інтелекту, щоб перетворити ваші відеодані на чіткі та практичні висновки, які допоможуть вам приймати рішення на основі даних, а не здогадок.

Як підвищує вашу безпеку

Централізоване керування кількома сайтами

Слідкуйте за всіма своїми сайтами з одного вікна – на комп'ютері або мобільних пристроях. Сповіщення в режимі реального часу гарантують, що ви будете в курсі всього важливого.

– Керування кількома сайтами з єдиного інтерфейсу.

– Події тривоги з будь-якого сповіщення.

– Розгортання дозволяє зберігати ваші хмарні відеоканали в одному інтерфейсі.

Оптимізація операцій

Завдяки централізованому керуванню відео та інформаційним панелям для підвищення операційної ефективності, вашим командам безпеки та операцій ще ніколи не було так легко знаходити та ділитися потрібними відеоматеріалами. Крім того, ви знаєте, що все оновлено завдяки автоматичним хмарним оновленням.

– Налаштування «підключи та працюй».

– Віддалений доступ.

– 99,9% безвідмовної роботи сервісу.

Уникайте апаратної прив'язки

Немає потреби демонтувати та замінювати вашу існуючу систему. Насолоджуйтесь свободою використання старих камер або підключення до найширшого асортименту камер, датчиків та пристроїв Інтернету речей на ринку:

– Оберіть з понад 20 000 пристроїв ONVIF.

– Використовуйте вбудовану аналітику від низки основних брендів камер.

– Або ж додайте аналітику до існуючих камер.

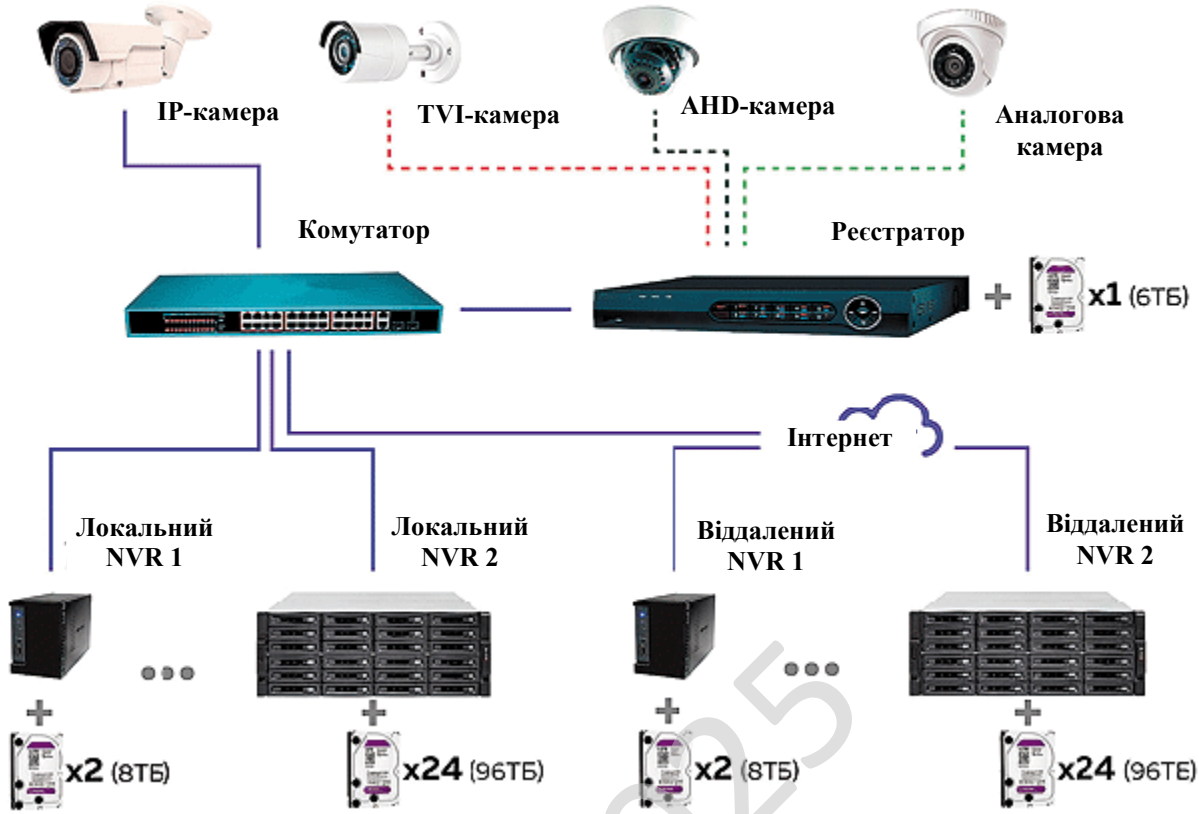

Рисунок 3.1 – Структурна схема системи

Робіть більше з аналітикою

Хмарні інтеграції включають контроль доступу, датчики Інтернету речей, судово-медичний пошук відео, теплові карти та аналітику зайнятості. Використовуйте аналітику штучного інтелекту, користувацькі правила та розширені інструменти пошуку для прийняття проактивних рішень.

– Аналітика ШІ для виявлення об'єктів, відстеження руху та проактивних сповіщень.

– Стан пристроїв та прогнозні тенденції оптимізують операції та зменшують витрати.

– Аналізуйте та діліться відеозаписами за лічені хвилини.

Задовольняйте потреби розташування, долайте обмеження

Не погоджуйтесь на універсальне рішення. Три варіанти розгортання дозволяють задовольнити ваші потреби та ефективно масштабуватися, незалежно від розміру та операційних вимог вашої організації.

Яка різниця між хмарою та локальною системою?

Фундаментальна різниця між традиційними та хмарними системами сягає корінням у віковий поділ IT на хмарні та локальні. Локальні рішення використовують локальні сервери для функціонування та зберігання даних, тоді як хмарні рішення використовують віддалені хмарні сервери для виконання того ж завдання.

Хмарні рішення стають дедалі більш помітними як у бізнес-середовищі, так і в споживчому середовищі. Електронна пошта вже перейшла від локального розміщення до хмари; спочатку електронна пошта була доступна лише через фізичні локальні сервери, а сьогодні домінують хмарні поштові платформи, такі як Gmail та Outlook.

У галузі відеобезпеки відбувається технологічний зсув, організації та постачальники усвідомлюють потенціал хмарних технологій.

Як працює хмарне відеоспостереження?

Усі системи камер безпеки фіксують записи спостереження. Основна відмінність між ними полягає в тому, що відбувається з відеоматеріалами після їх захоплення – як вони передаються та як зберігаються.

Донедавна всі системи відеоспостереження вимагали наявності локального відеореєстратора (NVR або DVR) для запису та зберігання відеоданих. Зазвичай це називають «локальною» або «традиційною» системою спостереження. Хмарні системи усувають потребу в локальних рекордерах та серверах – іншими словами, більше жодних NVR або DVR.

Системи хмарного відеоспостереження передають відеодані в хмару для хмарних обчислень та зберігання даних, що дозволяє отримати віддалений доступ до відео з будь-якого місця та з будь-якого пристрою.

Відеоспостереження без відеореєстраторів або цифрових відеореєстраторів

Хмарне відеоспостереження працює шляхом передачі відеоданих через Інтернет для зберігання відзнятого матеріалу в хмарі. «Хмара» – це просто віддалений сервер, де зберігаються відеодані, а не на локальному сервері чи жорсткому диску.

Для багатьох підприємств це бажана зміна. Традиційні системи з фізичними відеореєстраторами часто дорогі в обслуговуванні та складні у використанні. Хмарні системи легше обслуговувати IT-командам, а хмарне програмне забезпечення для керування відео відкриває цілу низку можливостей та функцій, які роблять управління безпекою набагато потужнішим.

Що таке хмарні IP-камери безпеки?

Хмарна IP-камера безпеки – це IP-камера, яка функціонує в хмарній системі відеоспостереження.

Як і традиційні IP-камери, хмарні камери передають відеодані за допомогою Інтернету – мережевого Ethernet-з'єднання, Wi-Fi, локальної мережі або стільникової мережі. Однак хмарні камери передають і зберігають відео на віддалений хмарний сервер, а не на локальний відеореєстратор (NVR або DVR).

Що таке програмне забезпечення для керування хмарним відео (VMS)?

Програмне забезпечення для керування відео (VMS) – це те, як ви використовуєте свою систему щодня. Щойно камери безпеки починають записувати відео, VMS – це інтерфейс, через який ви переглядаєте це відео та взаємодієте з ним.

За допомогою VMS ви можете:

– Переглянути відеоматеріали.

– Проводити розслідування.

– Зберігайте кліпи та діліться відеоматеріалами.

– Керування дозволами, камерами та системними налаштуваннями.

– Досягайте набагато більшого залежно від вашої системи, деякі з яких пропонують функції поглибленої аналітики.

Локальні системи VMS передбачають встановлення програмного забезпечення на локальні машини та часто вимагають від користувачів присутності на місці для доступу до відеозаписів або внесення змін до системи. Хмарна VMS має суттєву відмінність та перевагу: вбудований віддалений доступ.

Програмне забезпечення для керування хмарним відео не прив'язане до конкретних фізичних пристроїв. Зазвичай воно базується на додатках або браузерах і може бути завантажене на будь-який пристрій. Це надає користувачам набагато більше гнучкості в управлінні відеоспостереженням для своєї організації.

Що означає «VSaaS»?

«VSaaS» розшифровується як «Відеонагляд як послуга» і є ще одним позначенням хмарного відеоспостереження. Ця фраза стала стандартним галузевим терміном за останні кілька років. Як пропозиція «Програмне забезпечення як рішення» (SaaS), користувачі купують періодичні ліцензії на додаток до обладнання.

Хмарні ліцензії зазвичай охоплюють комплексне рішення, яке включає наступне:

– Програмне забезпечення для керування відео (VMS).

– Хмарне сховище та архівування.

– Локальне сховище на камері.

– Доступ до сервера.

– Автоматичні оновлення.

– Підтримка клієнтів.

– Розширені функції та аналітика штучного інтелекту (залежно від постачальника).

Розширені функції та аналітика хмарної системи керування відеоматеріалами (VMS)

Сьогодні багато хмарних систем відеоспостереження використовують можливості хмарних обчислень, щоб відкрити цілий світ нових функцій та можливостей. Сучасні системи можуть аналізувати величезні обсяги відео без будь-якої ручної роботи з боку користувача, що дозволяє швидко переміщатися по відеоматеріалах та отримувати проактивні сповіщення в режимі реального часу. Завдяки єдиній хмарній платформі ці функції легко використовувати в великих масштабах одночасно в кількох місцях – наприклад, ви можете одночасно шукати відеоматеріали з об'єкта в Кропивницькому та об'єкта в Києві.

Розширені системні функції можуть включати:

– Вбудований віддалений доступ: Віддалений доступ є основою будь-якої хмарної системи відеоспостереження. Переглядайте відеозаписи та використовуйте всю свою систему з будь-якої точки світу для більшої зручності, плавнішої роботи та економії часу.

– Інтелектуальний пошук для швидких розслідувань: Більше не потрібно переглядати відеоматеріали. Використовуйте інструменти на базі штучного інтелекту для легкої навігації відеопотоками та швидкого пошуку потрібної інформації.

– Легкий обмін прямими трансляціями та кліпами: завдяки хмарному віддаленому доступу ви можете легко ділитися відеоматеріалами дистанційно за допомогою таких методів, як URL-адреса, SMS та електронна пошта.

– Аналітика на основі штучного інтелекту: пошук по всіх камерах та місцях розташування за допомогою розпізнавання облич, розпізнавання транспортних засобів та номерних знаків, пошуку за кольором, підрахунку людей, виявлення шуму, аудіоаналітики, виявлення незвичайної поведінки тощо.

– Автоматичні оновлення: Підключення до хмари дозволяє постачальникам автоматично розгортати нові функції та оновлення безпеки, що спрощує обслуговування та гарантує відсутність прогалин у кібербезпеці.

– Сповіщення в режимі реального часу на будь-якому пристрої: отримуйте налаштовувані автоматичні сповіщення в дорозі, щоб зробити вашу безпеку проактивною та забезпечити душевний спокій у неробочий час.

– Мобільні додатки: Хмара дозволяє вам віддалено отримувати доступ до вашої системи з будь-якої точки світу. Мобільні додатки максимально використовують можливості віддаленого доступу та забезпечують повний контроль над ситуацією.

– Гнучкі дозволи користувачів: Більше жодних обмежень на основі пристроїв чи місцезнаходження. Хмарна система керування відеоматеріалами (VMS) дозволяє встановлювати власні, детальні дозволи користувачів, які відповідають унікальним потребам вашої організації.

3.3 Розробка функціональної схеми

Опишемо функції розробленої системи відеоспостереження, побудованої на базі мережевих сховищ на основі технології NVR.

Функціональна схема системи складається з наступних блоків:

– Блок функцій адміністратора.

– Блок функцій оператора.

– Блок функцій інженера.

Блок функцій адміністратора, який включає в себе наступні функції:

1. Додавання нових користувачів.

Система дозволяє не тільки контролювати обстановку по відеоряду, але й відслідковувати всі зміни на двомірному плані охоронюваного простору. На плані відображаються області видимості IP-камер і виявлені об'єкти, що полегшує сприйняття загальної обстановки.

Так, наприклад, на плані легко помітити області контрольованого простору не видимі ні однією IP-камерою.

Крім того система надає користувальницький інтерфейс для керування IP-камерами за допомогою кліків миші на компоненті головного вікна, із зображенням плану.

Блок функцій оператора, який включає в себе наступні функції:

1. Перегляд відео з IP-камер.
2. Оцінка ситуації по плану:

– Відображення областей спостереження IP-камер.

– Відображення доданих об’єктів.

– Відображення знайдених об’єктів.

3. Керування IP-камерами:

– Клік по відео.

– Обертання IP-камери.

– Клік по карті.

– Режим супроводу.

4. Зміна налаштувань системи.

Блок функцій інженера, який включає в себе наступні функції:

1. Завантаження плану контролюємого простору.
2. Додавання заборонених областей на план.
3. Додавання об’єктів на план.
4. Калібрування IP-камер.
5. Додавання та видалення IP-камер.
6. Калібрування плану.

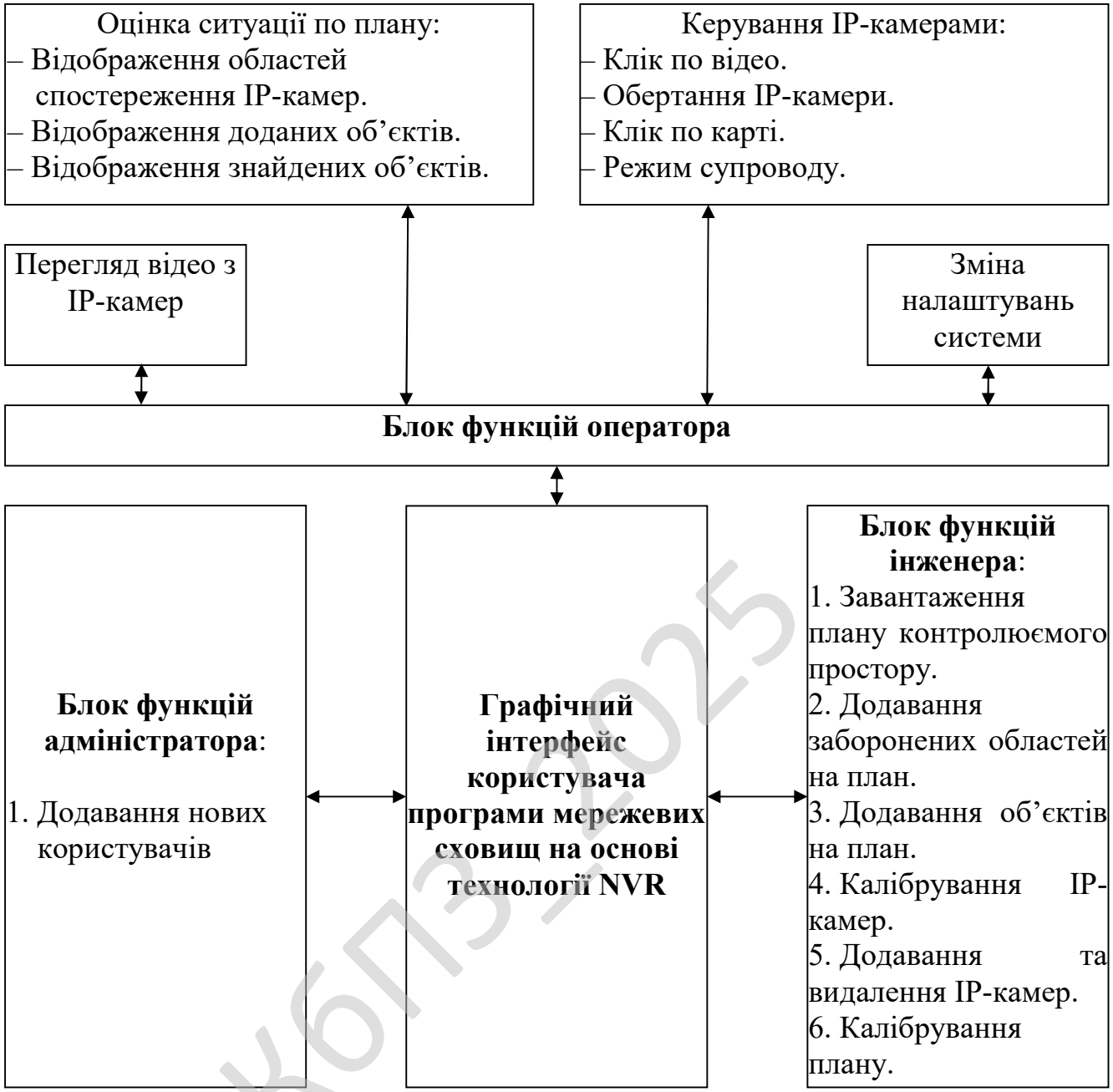

Рисунок 3.2 – Функціональна схема системи

Система відеоспостереження має розвитий графічний інтерфейс і істотну обчислювальну частину.

Система має більш багату функціональність ніж існуючі комерційні розробки. Використання поворотних IP-камер, розмаїтість можливостей по керуванню IP-камерами й спостереженню за обстановкою дозволяє ефективніше вирішувати завдання відеоспостереження, ніж при використанні традиційних систем відеоспостереження з нерухливими IP-камерами.

Таких можливостей удалося досягти завдяки використанню розроблених методів калібрування IP-камер і виявлення областей інтересу.

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

3.4 Розробка діаграми процесів

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

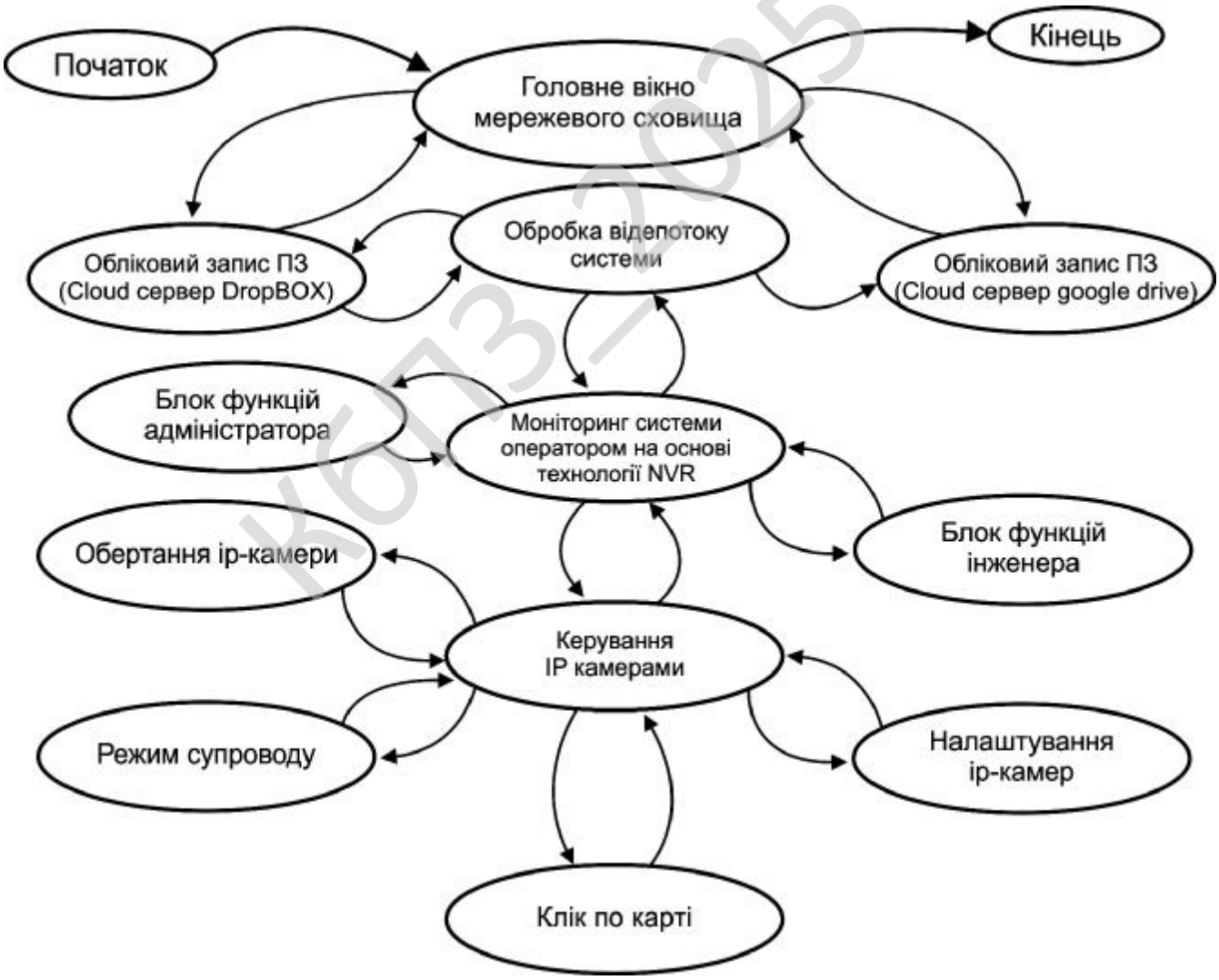

Рисунок 3.3 – Діаграма взаємодії процесів

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

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо Real Time Streaming Protocol який було використано у розробленій системі.

RTSP являє собою протокол керування відеопотоками для використання в розважальних і комунікаційних системах.

Протокол RTSP дає можливість клієнтові запитувати потоки з мультимедійних серверів, подібно тому, як HTTP дозволяє клієнтам видавати запити до Web-серверів.

Фактично RTSP перейняв більшу частину свого синтаксису й семантики від HTTP/1.1, оскільки формально обидва протоколи виконують схожі функції. Подібність підкреслює загальний характер багатьох реалізованих в HTTP концепцій.

Однак протоколи мають ряд ключових відмінностей, які пов'язані з унікальними вимогами для мультимедійних потоків і обмеженістю можливостей HTTP/1.1 по передачі мультимедійних даних.

Перед розглядом специфікація RTSP, необхідно знати що таке SDP. SDP (Session Description Protocol) мережний протокол, призначений для опису сесії передачі потокових даних мультимедіа.

Сесія SDP може реалізовувати кілька потоків даних. У протоколі SDP у цей час визначені відео дані.

Повідомлення SDP, передане від одного вузла іншому може вказувати на:

– адреси місця призначення, які можуть бути для медіа-потоків мультикастінг–адресами;

– номера UDP портів для відправника й одержувача;

– медіа-формати (наприклад кодек), які можуть застосовуватися під час сесії;

– часу старту й зупинки. Використовується у випадку широкомовних сесій, наприклад, телевізійних або радіопрограм.

Можна внести час початку, завершення й часи повторів сесії. Незважаючи на те, що Session Description Protocol надає можливість опису мультимедіаданих, у ньому не вистачає механізмів узгодження параметрів сесії. Розглянемо приклад надання моделі узгодження на основі механізму відгуку.

У прикладі вузли обмінюються SDP повідомленнями з метою дійти згоди щодо формату даних.

```
v=0
o=- 1815849 0 IN IP4 194.67.15.181
s=Cisco SDP 0
c=IN IP4 194.67.15.181
t=0 0
m=audio 20062 RTP/AVP 99 18 101 100
a=rtpmap:99 G.729b/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=rtpmap:100 X-NSE/8000
a=fmtp:100 200-202
a=sqn:0
a=cdsc: 1 audio RTP/AVP 99 18 101 100
a=cdsc: 5 image udptl t38
a=cpar: a=T38Faxversion:0
a=cpar: a=T38Faxratemanagement:transferredtcf
a=cpar: a=T38Faxmaxdatagram:160
a=X-sqn:0
a=X-cap: 1 image udptl t38
```

Специфікація RTSP значною мірою перегукується зі специфікацією HTTP/1.1. На верхньому рівні обидва протоколи переслідують одну мету — дати можливість клієнтові запитувати зміст із сервера на основі URI запиту.

Історично розробка RTSP почалася після того, як була виконана частина робіт над специфікацією HTTP.

Оскільки HTTP послужив основою для RTSP, мети й завдання цих протоколів багато в чому збігаються.

Доставка мультимедійних потоків використовує таку ж саму модель, що й HTTP: ресурси, URI що ідентифікуються, взаємодія запит/відповідь, а також можливість передачі даних через одне або декількох проміжних ланок на шляху між клієнтом і сервером.

Використання готових концепцій, включення спеціальних заголовків протоколу й кодів відповідей зменшило витрати на розробку й реалізацію RTSP. Крім того, обидва протоколи відіграють роль в ініціюванні передачі мультимедійних потоків через WEB.

Подібність між двома протоколами забезпечує значну гнучкість при ухваленні рішення, який протокол буде обслуговувати конкретну функцію.

Розглянемо запит на інформацію з описом сеансу (наприклад, http://www.foo.com/bar.sdp). Відповідь Web-сервера буде включати інформацію, у форматі SDP:

```
HTTP/1.1 200 OK Content-Type: application/sdp
v=0
o=- 2890844526 2890842807 IN IP4 192.16.24.202 s=RTSP Session
m=audio 0 RTP/AVP 0
a=control:
rtsp://foo.com/bar/audio
m=video 0 RTP/AVP 31
a=control:
rtsp://foo.com/bar/video
```

У цей момент Web-браузер уже може активізувати медіаплейер для виконання інших дій. Крім того, медіаплейер може надати користувачеві інтерфейс для вибору мультимедійних сеансів.

У цьому випадку медіаплейер може прямо взаємодіяти з Rtsp-сервером для уточнення описів без залучення Web-браузера.

Методи запитів RTSP. RTSP-сервери повинні підтримувати чотири основні методи, використовуваних клієнтами для добування мультимедійних сеансів: OPTIONS, SETUP, PLAY і TEARDOWN.

На верхньому рівні заголовок OPTIONS дозволяє клієнтові визначати функціональні можливості сервера, такі як номер версії RTSP і підтримувані методи; цей метод поводиться так само, як метод OPTIONS HTTP/1.1.

Інші три методи маніпулюють збереженої на сервері інформацією про стан для координації передачі мультимедійних даних. Клієнт використовує метод SETUP для встановлення транспортного з'єднання для кожного потоку в сеансі. Метод PLAY використовується для ініціювання передачі потоку (потоків).

Метод TEARDOWN використовується для завершення передачі. Більшість методів запитів пов'язане з підтримкою клієнта, який запитує відтворення потоку з мультимедійного сервера.

Однак RTSP – сервер може також допускати запис мультимедійного змісту за допомогою необов'язкових методів ANNOUNCE і RECORD.

Допустимо, користувач прагне записати телеконференцію на Rtsp-сервері для подальшого відтворення. У цьому випадку клієнт відправляє опис конференції серверу в повідомленні ANNOUNCE.

Потім клієнт повинен відправити повідомлення SETUP для кожного з потоків у сеансі, щоб проінформувати сервер про транспортні параметри, таких як Ip-адреси й номера портів. URI запиту в повідомленні SETUP указує ім'я, яке клієнт прагне асоціювати із записуваним змістом.

Після цього клієнтові слід відправити запит RECORD для початку запису потоків. Через якийсь час клієнт може зв'язатися із сервером і переглянути записану конференцію.

Обробка відео потоку і виведення на екран в середовищі Windows при застосуванні основних методів виведення відео інформації на екран лінійки операційних систем Windows.

Виникає гостра проблема в швидкості обробки потокового кадру, що приводить до уповільнення процесу виведення інформації на екран. Як відомо для перегляду відеопотоку необхідно не менше 24 кадрів в секунду.

Під час розробки ПЗ було прийнято рішення використовувати підходи P2P при організації мережевих сховищ.

Peer-to-peer (рівний до рівного) – варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів.

Комп'ютерні мережі типу peer-to-peer (або P2P) засновані на принципі рівноправності учасників і характеризуються тим, що їх елементи можуть зв'язуватися між собою, на відміну від традиційної архітектури, коли лише окрема категорія учасників, яка називається серверами може надавати певні сервіси іншим.

Фраза «peer-to-peer» була вперше використана у 1984 році Парбауелом Йохнухуйтсманом (Parbawell Yohnuhuitsman) при розробці архітектури Advanced Peer to Peer Networking фірми IBM.

В чистій «peer-to-peer» мережі не існує поняття клієнтів або серверів, лише рівні вузли, які одночасно функціонують як клієнти та сервери по відношенню до інших вузлів мережі. Ця модель мережевої взаємодії відрізняється від клієнт-серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером.

Така організація дозволяє зберігати працездатність мережі при будь-якій конфігурації доступних її учасників. Проте практикується використання P2P мереж які все ж таки мають сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі.

В P2P системі автономні вузли взаємодіють з іншими автономними вузлами. Вузли є автономними в тому сенсі, що не існує загальної влади, яка може контролювати їх. В результаті автономії вузлів, вони не можуть довіряти один одному та покладатися на поведінку інших вузлів, тому проблеми масштабування та надмірності стають важливішими ніж у випадку традиційної архітектури.

Сучасні P2P-мережі набули розвитку завдяки ідеям, пов'язаними з обміном інформацією, які формувалися у руслі того, кожен вузол може надавати та

отримувати ресурси які надаються будь-якими іншими учасниками. У випадку мережі Napster, це був обмін музикою, в інших випадках це може бути надання процесорного часу для пошуку інопланетних цивілізацій (SETI@home) або ліків від раку (Folding@home).

Переваги P2P

Розподіл/зменшення вартості. Централізовані системи, які обслуговують багато клієнтів, зазвичай складають більшість вартості системи. Коли, ця вартість стає дуже великою, архітектура P2P може допомогти розподілити вартість серед користувачів. Наприклад, серед систем файлообміну Napster дозволив розподілити вартість зберігання файлів і міг підтримувати індекс, потрібний для сумісного використання. Економія коштів, здійснюється за допомогою використання та об'єднання ресурсів, які в іншому випадку не використовуються (наприклад SETI@home). Оскільки вузли зазвичай є автономними, важливо розподіляти витрати справедливо.

Об'єднання ресурсів. Децентралізований підхід веде до об'єднання ресурсів. Кожен вузол в системі P2P приносить певні ресурси як наприклад обчислювальна потужність або пам'ять. У програмах, які потребують величезну кількість цих ресурсів, як наприклад intensive моделювання або розподілені файлові системи, природно використовувати P2P, щоб залучити ці ресурси. Розподілені обчислювальні системи, як наприклад SETI@Home, distributed.net, і Endeavours – очевидні приклади цього підходу. Об'єднуючи ресурси тисяч вузлів, вони можуть виконувати важкі з точки зору кількості обчислень функції. Файлобмінні системи, як наприклад Napster, Gnutella, і так далі, також об'єднують ресурси. У цих випадках, це дисковий простір, щоб зберігати дані, та пропускна спроможність, щоб їх передавати.

Вдосконалена масштабованість/надійність. З відсутністю сильної центральної влади по відношенню до автономних вузлів, важливою метою є покращення масштабованості і надійності. Масштабованість і надійність визначаються в традиційному для розподілених систем сенсі, як наприклад

використання пропускної спроможності – скільки вузлів можуть бути досягнуті від одного вузла, скільки вузлів може підтримуватися, скільки користувачів може підтримуватися. Розподілена природа peer-to-peer мереж також збільшує помилкостійкість у разі невдач, шляхом дублювання даних поміж багатьох вузлів, і – в чистих системах P2P – надаючи можливість вузлу знайти дані без залежності від єдиного централізованого індексного сервера. У останньому випадку, немає ніякої єдиної критичної точки в системі.

Збільшена автономія. У багатьох випадках, користувачі розподіленої системи не бажають залежати від будь-якого централізованого постачальника послуг. Натомість, вони воліють, щоб всі дані та призначена для них робота виконувалась локально. Системи P2P підтримують цей рівень автономії, тому що вони вимагають, щоб кожен вузол робив необхідну для нього частину праці.

Анонімність/конфіденційність. Пов'язаним із автономією є поняття анонімності і конфіденційності. Користувач, можливо, не хоче, щоб кого-небудь або будь-який постачальник послуг знав про нього або про його роль у системі. З центральним сервером, гарантувати анонімність важко, тому що сервер зазвичай зможе ідентифікувати клієнта, як мінімум через його адресу в Інтернет. Використовуючи структуру P2P, в якій дії виконуються локально, користувачі можуть уникати необхідності передавати будь-яку інформацію про себе до кого-небудь іншого. FreeNet – яскравий приклад того, як анонімність може вбудуватися в додаток P2P. Він пересилає повідомлення через інші вузли, щоб забезпечити неможливість вистежування початкового автора. Це збільшує анонімність, використовуючи ймовірнісні алгоритми таким чином, щоб походження не можливо було легко відстежити аналізуючи трафік у мережі.

Динамічність. Системи P2P припускають, що оточення надзвичайно динамічне. Тобто, ресурси, як наприклад вузли, з'являються та зникають із системи безперервно. У випадках комунікації, як наприклад мережі для обміну повідомленнями, використовуються так званий «список контактів», щоб інформувати користувачів, коли їхні друзі стають доступними. Без цього,

потрібно було би, щоб користувачі «опитували» партнерів, посилаючи періодичні повідомлення. У випадку розподілених обчислень, як наприклад distributed.net і SETI@home, система повинна пристосуватись до заміни учасників. Тому вони повинні повторно видавати завдання для обчислення іншим учасникам, щоб гарантувати, що робота не втрачена, якщо попередні учасники відпадають від мережі, поки вони виконували крок обчислення.

Класифікація P2P систем

За функціями:

1. Розподілені обчислення. Обчислювальна проблема розподіляються на невеликі незалежні частини. Обробка кожної з частин робиться на індивідуальному ПК і результати збираються на центральному сервері. Цей центральний сервер відповідальний за розподілення елементів роботи серед окремих комп'ютерів в Інтернеті. Кожен із зареєстрованих користувачів має клієнтське програмне забезпечення. Воно користується періодами бездіяльності в ПК (часто це характеризується часами активації скрінсейверів), щоб виконувати деяке обчислення, надане сервером. Після того, як обчислення закінчене, результат посилається назад до сервера, і нова робота передається для клієнта.

2. Файлообмін. Зберігання та обмін даними – це одна з областей, де технологія P2P була найуспішнішою. Мультимедійні дані, наприклад, вимагають великих файлів. Napster і Gnutella використовувались користувачами, щоб обійти обмеження пропускної спроможності, які роблять передачу великих файлів неприйнятними.

3. Співпраця. Природа технології P2P робить її добре придатною для забезпечення співпраці між користувачами. Це може бути обмін повідомленнями, онлайн ігри, сумісна робота над документами в бізнесі, освіті та дома.

За ступенем централізації:

1. Чисті peer-to-peer системи. Вузли є рівними, поєднуючи ролі серверу та клієнту. Не існує центрального сервера, що керує мережею. Прикладами таких систем є Gnutella та Freenet

2. Гібридні peer-to-peer системи. Мають центральний сервер, що зберігає інформацію про вузли та відповідає на запити відносно цієї інформації. Вузли займаються забезпеченням ресурсами (тому що центральний сервер їх не має), повідомленням сервера про наявність цих ресурсів надання ресурсів іншим вузлам, які бажають ними скористатися.

В залежності від того, як вузли з'єднуються один з одним можна поділити мережі на структуровані та неструктуровані:

1. Неструктурована мережа P2P формується, коли з'єднання встановлюються довільно. Такі мережі можуть бути легко сконструйовані, оскільки новий вузол, який хоче приєднатися до мережі, може скопіювати існуючі з'єднання іншого вузла, а вже потім почати формувати свої власні. У неструктурованій мережі P2P, якщо вузол бажає знайти певні дані в мережі, запит доведеться передати майже через всю мережу, щоб охопити так багато вузлів, як можливо. Головним недоліком таких мереж є те, що запити, можливо, не завжди вирішуються. Скоріш за все популярні дані будуть доступні в багатьох вузлів та пошук швидко знайде потрібне, але якщо вузол шукає рідкісні дані, наявні лише в декількох інших вузлів, то надзвичайно малоймовірно, що пошук буде успішним. Оскільки немає ніякої кореляції між вузлами та даними, що вони зберігають, немає ніякої гарантії, що запит знайде вузол, який має бажані дані.

2. Структурована мережа P2P використовує єдиний алгоритм, щоб гарантувати, що будь-який вузол може ефективно передати запит іншому вузлу, який має бажаний файл, навіть якщо файл надзвичайно рідкісний. Така гарантія потребує структуровану систему з'єднань. У наш час найпопулярнішим типом структурованої мережі P2P є розподілені хеш-таблиці, в яких хешування використовується для встановлення зв'язку між даними та конкретним вузлом, який за них відповідає.

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

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

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

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

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

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

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

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

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу.

Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

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

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

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

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю мережевих сховищ на основі технології NVR.

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 – перевірка поточного стану з завершенням

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

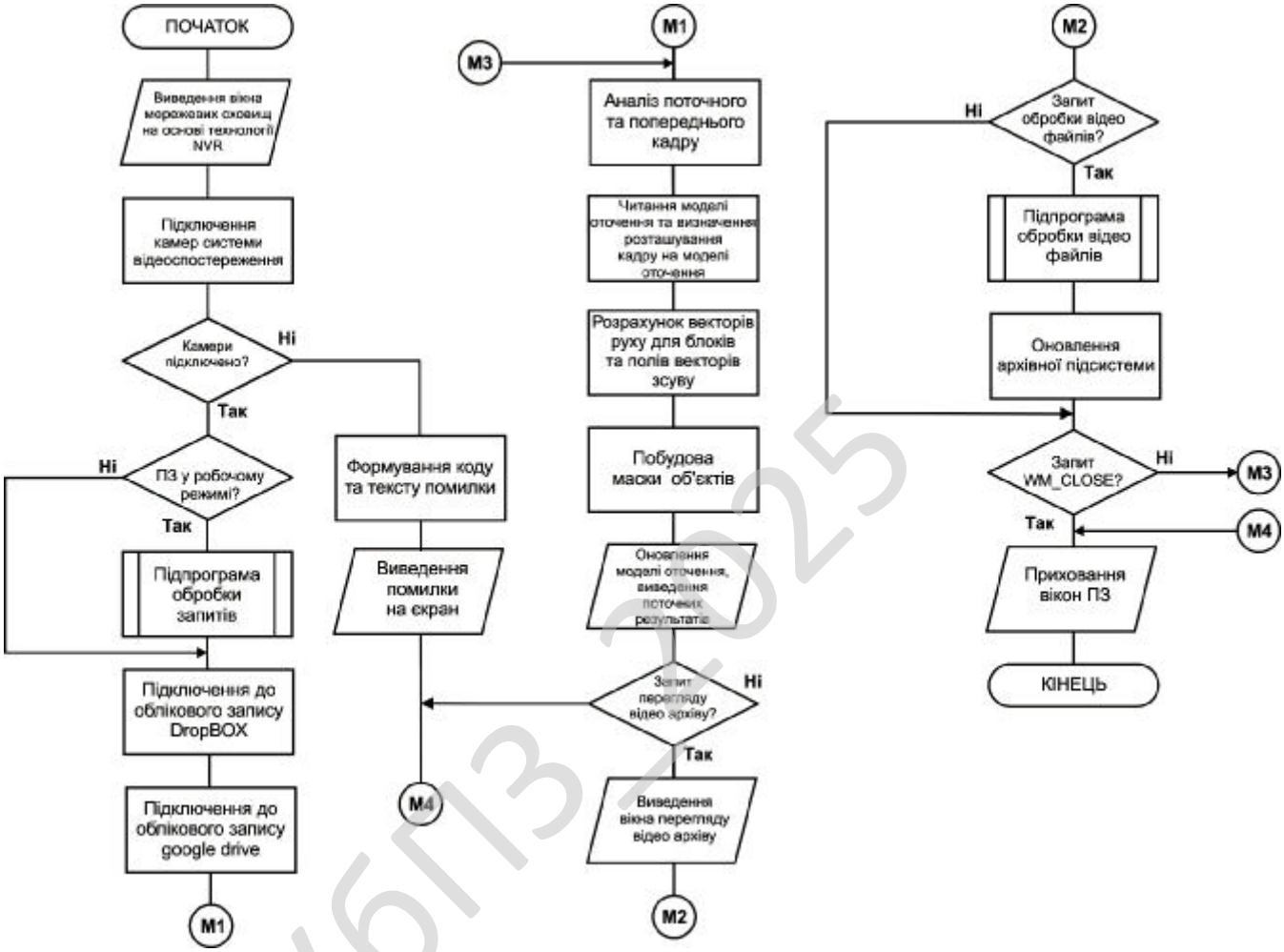

Рисунок 4.1 – Блок-схема основної програми

При розробці ПЗ було використано V-Model (або VEE модель) є моделлю розробки інформаційних систем (ІС), спрямованої на спрощення розуміння складнощів, пов'язаних з розробкою систем. Вона використовується для визначення єдиної процедури розробки програмного забезпечення, апаратного забезпечення та людино-машинного інтерфейсу.

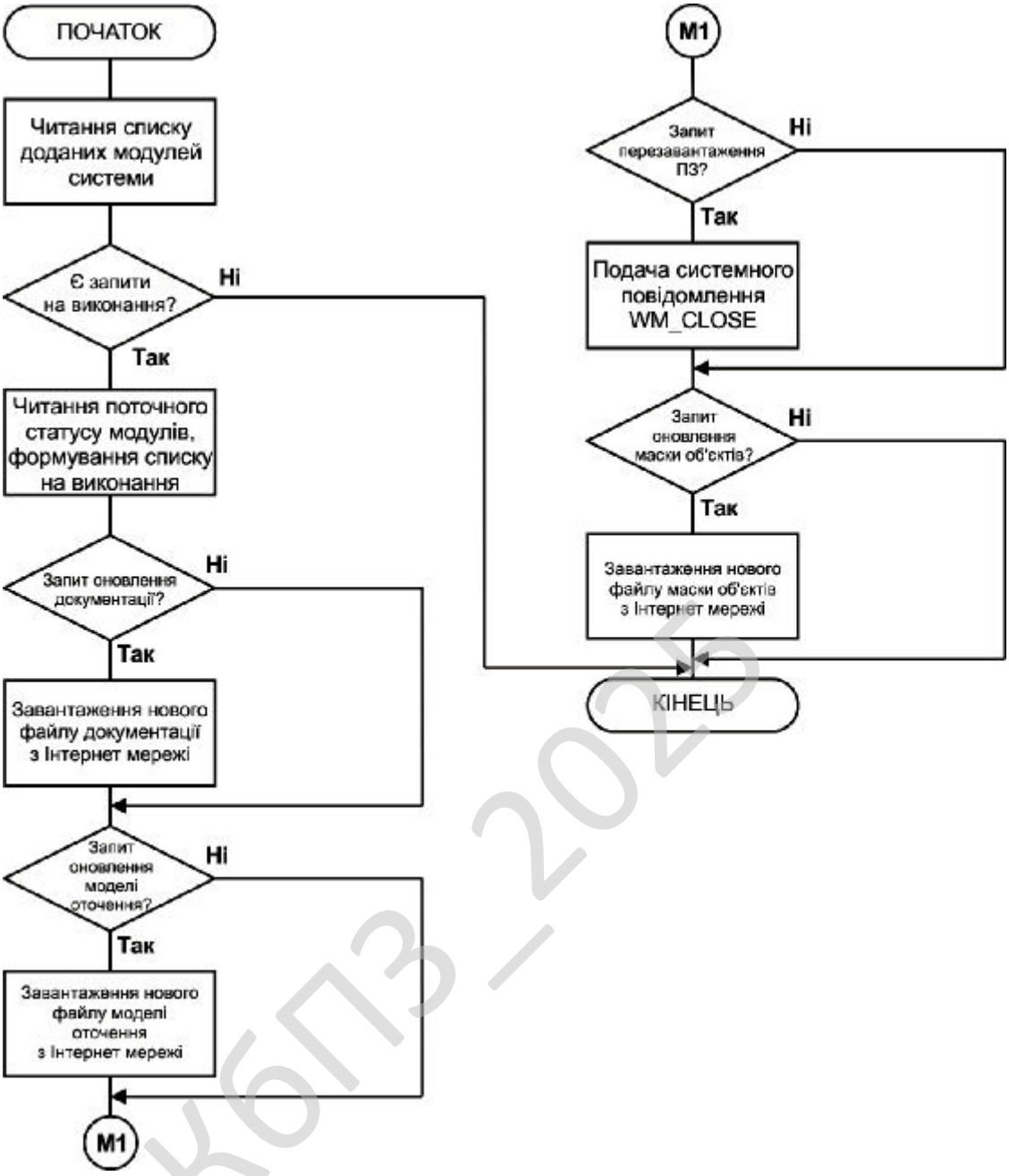

Рисунок 4.2 – Блок-схема роботи підпрограми

Концепція V-подібної моделі була розроблена Німеччиною та США в кінці 1980-х років незалежно один від одного:

– Німецька V-модель була розроблена аерокосмічної компанією IABG в Оттобрунні поряд з Мюнхеном у сприянні з Федеральним департаментом з закупівлі озброєнь в Кобленці, для Міністерства оборони Німеччини. Модель

була прийнята німецькою федеральною адміністрацією для цивільних потреб влітку 1992.

– Американська V-Model (VEE) була розроблена національною радою з системної інженерії (міжнародна – з 1995 року) для супутникових систем, включаючи обладнання, програмне забезпечення та взаємодію з користувачами.

Сучасною версією V-Model є V-Model XT, яка була затверджена в лютому 2005 року. V-модель використовується для управління процесом розробки програмного забезпечення для німецької федеральної адміністрації.

Зараз вона є стандартом для німецьких урядових і оборонних проектів, а також для виробників ПЗ в Німеччині. V-Model являє собою скоріше набір стандартів у галузі проектів, що стосуються розробки нових продуктів. Ця модель багато в чому схожа з Prince2 і описує методи як для проектного управління, так і для системного розвитку.

Основні принципи

Основний принцип V-подібної моделі полягає в тому, що деталізація проекту зростає при русі зліва направо, одночасно з плином часу, і ні те, ні інше не може повернути назад. Ітерації в проекті виробляються по горизонталі, між лівою і правою сторонами літери.

Стосовно до розробки інформаційних систем V-Model – варіація каскадної моделі, в якій завдання розробки йдуть зверху вниз по лівій стороні букви V, а завдання тестування – вгору по правій стороні букви V. Усередині V проводяться горизонтальні лінії, що показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній із фаз тестування.

Модель базується на тому, що приймально-здавальні випробування ґрунтуються, насамперед, на вимогах, системне тестування – на вимогах та архітектури, комплексне тестування – на вимогах, архітектурі та інтерфейсах, а компонентне тестування – на вимогах, архітектурі, інтерфейсах та алгоритмах.

Цілі

V-модель забезпечує підтримку у плануванні та реалізації проекту. В ході проекту ставляться такі завдання:

1. Мінімізація ризиків: V-подібна модель робить проект більш прозорим і підвищує якість контролю проекту шляхом стандартизації проміжних цілей і опису відповідних їм результатів та відповідальних осіб. Це дозволяє виявляти відхилення в проекті і ризики на ранніх стадіях і покращує якість управління проектом.

2. Підвищення та гарантії якості: V-Model – стандартизована модель розробки, що дозволяє домогтися від проекту результатів бажаної якості. Проміжні результати можуть бути перевірені на ранніх стадіях. Універсальне документування полегшує читаність, зрозумілість та контрольованість.

3. Зменшення загальної вартості проекту: Ресурси на розробку, виробництво, управління і підтримку можуть бути заздалегідь прораховані та проконтрольовані. Отримувані результати також універсальні і легко прогнозуються. Це зменшує витрати на подальші стадії та проекти.

4. Підвищення якості комунікації між учасниками проекту: Універсальний опис усіх елементів та умов полегшує взаєморозуміння всіх учасників проекту. Таким чином, зменшуються неточності у розумінні між користувачем, покупцем, постачальником і розробником.

Переваги:

– Користувачі V-Model беруть участь у розробці та підтримці V-моделі. Комітет з контролю за змінами підтримує проект і збирається раз на рік для обробки всіх отриманих запитів на внесення змін до V-Model.

– На старті будь-якого проекту V-подібна модель може бути адаптована під цей проект, так як ця модель не залежить від типів організацій та проектів.

– V-model дозволяє розбити діяльність на окремі кроки, кожен з яких буде включати в себе необхідні для нього дії, інструкції до них, рекомендації та докладне пояснення діяльності.

Обмеження. Наступні моменти не враховуються в V-моделі, але можуть бути розглянуті окремо, або можливо адаптувати модель під них:

– Не регулюється розміщення контрактів на обслуговування.

– Організація і виконання управління, обслуговування, ремонту та утилізації системи не враховуються в V-моделі. Однак, планування і підготовка до цих операцій моделлю розглядаються.

– V-подібна модель більше стосується розробки програмного забезпечення в проекті, ніж всієї організації процесу.

– У моделі особливе значення надається плануванню, спрямованому на верифікацію та атестацію розроблювального продукту на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації.

– У моделі передбачені атестація та верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту.

– У V-подібної моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПО – перед розробкою компонентів.

– Модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню.

– Завдяки моделі менеджери проекту можуть відслідковувати хід процесу розробки, так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою.

Недоліки:

– Модель не передбачає роботу з паралельними подіями.

– У моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу.

– Тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести змін, не вплинувши при цьому на графік виконання проекту.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою MARS, який є блочно-симетричним шифром з відкритим ключем. Розмір блоку при шифруванні 128 біта, розмір ключа може варіюватися від 128 до 448 біт включно (кратні 32 бітам). Творці прагнули поєднати в своєму алгоритмі швидкість кодування і стійкість шифру. В результаті вийшов один з самих криптостійкий алгоритм з алгоритмів, які брали участь в конкурсі AES.

Алгоритм унікальний тим, що використовував практично всі існуючі технології, застосовувані в криптоалгоритмах, а саме:

– Найпростіші операції (додавання, віднімання, виключаюче або).
– Підстановки з використанням таблиці замін.
– Фіксований циклічний зсув.
– Залежний від даних циклічний зсув.
– Множення за модулем 2^{32}.
– Ключове забілювання.

Використання подвійного перемішування представляє складність для криптоаналіз а, що деякі відносять до недоліків алгоритму. У той же час на даний момент не існує будь-яких ефективних атак на алгоритм, хоча деякі ключі можуть генерувати слабкі підключі.

Структура алгоритму

Автори шифру виходили з наступних припущень:

1. Вибір операцій. MARS був спроектований для використання на найсучасніших комп'ютерах того часу. Для досягнення найкращих захисних характеристик в нього були включені самі «сильні операції» підтримувані в них. Це дозволило добитися більшого відносини securityper-instruction для різних реалізації шифру.

2. Структура шифру. Двадцятирічний досвід роботи в області криптографії підштовхнув творців алгоритму до думки, що кожен раунд

шифрування грає свою роль в забезпеченні безпеки шифру. Зокрема, ми можемо бачити, що перший і останній раунди зазвичай сильно відрізняються від проміжних («центральних») раундів алгоритму в плані захисту від криптоаналітичних атак. Таким чином, при створенні MARSa використовувалася змішана структура, де перший і останній раунди шифрування істотно відрізняються від проміжних.

3. Аналіз. Швидше за все, алгоритм з гетерогенною структурою буде краще протистояти криптоаналітичних методам майбутнього, ніж алгоритм, всі раунди якого ідентичні. Розробники алгоритму MARS надали йому сильно гетерогенну структуру – раунди алгоритму дуже різняться між собою.

У шифрі MARS використовувалися такі методи шифрування:

1. Робота з 32-х бітними словами. Всі операції застосовуються до 32-бітовим словами. тобто вся початкова інформація розбивається на блоки по 32біта. (Якщо ж блок опинявся меншої довжини, то він доповнювався до 32біт)

2. Мережа Фейстеля. Творці шифру вважали, що це найкращий варіант поєднання швидкості шифрування і криптостійкості. В MARS використана мережа Фейстеля 3-го типу.

3. Симетричність алгоритму. Для стійкості шифру до різних атакам всі його раунди були зроблені повністю симетричними, тобто друга частина раунду є дзеркальне повторення першої його частини.

Структуру алгоритму MARS можна описати таким чином:

1. Попереднє накладення ключа: на 32-бітові субблоки A, B, C, D накладаються 4 фрагмента розширеного ключа k_0 … k_3 операцією складання за модулем 2^{32}.

2. Виконуються 8 раундів прямого перемішування (без участі ключа шифрування).

3. Виконуються 8 раундів прямого криптоперетворення.

4. Виконуються 8 раундів зворотного криптоперетворення. [2]

5. Виконуються 8 раундів зворотного перемішування, також без участі ключа шифрування.

6. Фінальне накладення фрагментів розширеного ключа k_{36} … k_{39} операцією віднімання за модулем 2^{32}.

Пряме перемішування

У першій фазі на кожне слово даних накладається слово ключа, а потім відбувається вісім раундів змішування згідно з мережею Фейстеля третього типу спільно з деякими додатковими змішування. У кожному раунді ми використовуємо одне слово даних (зване, вихідним словом) для модифікації трьох інших слів (звані, цільовими словами). Ми розглядаємо чотири байта вихідного слова як індексів на двох S-блоків, S_0 і S_1 , кожен, що складається з 256 32-розрядних слів, а далі проводимо операції XOR або додавання даних відповідного S-блоку в три інших слова.

Якщо чотири байти вихідного слова b_0 , b_1 , b_2, b_3 (де b_0 є першим байтом, а b_3 є старшим байтом), то ми використовуємо b_0 , b_2, як індекси в блоку S_0 і байти b_1 , b_3 , як індекси в S-блоці S_1 . Спочатку зробимо XOR S_0 до першого цільовим речі, а потім додамо S_1 до того ж слова. Ми також додаємо S_0 до другого цільовим слову і XOR блоку-S_1 до третього цільовим слову. У висновку, ми обертаємо вихідне слово на 24 біта вправо.

У наступному раунді ми обертаємо наявні у нас чотири слова: таким чином, нинішні перші цільове слово стає наступним вихідним словом, поточним другий цільове слово стає новим першим цільовим словом, третє цільове слово стає наступний другий цільовим словом, і поточне вихідне слово стає третім цільовим словом.

Більш того, після кожного з чотирьох раундів ми додаємо одне з цільових слів назад у вихідне слово. Зокрема, після першого і п'ятого раундів ми додав третю цільове слово назад у вихідне слово, а після другого і шостого раунду ми додаємо першої цільової слово назад у вихідне слово. Причиною цих додаткових операцій змішування, є ліквідація декількох простих диференціальних

криптоатаки в фазі перемішування, щоб порушити симетрію у фазі змішування та отримати швидкий потік.

Криптографічне ядро

Криптографічне ядро MARS – мережа Фейстеля 3-го типу, що містить в собі 16 раундів. У кожному раунді ми використовуємо ключову E-функцію, яка є комбінацією множень, обертань, а також звернень до S-блоків. Функція приймає на вхід слово даних, а повертає три слова, з якими згодом буде здійснена операція додавання або XOR до інших трьох слів даними. У доповненні вихідне слово обертається на 13 біт вліво.

Для забезпечення, серйозного опору до криптоатаки, три вихідних значення E-функції (O_1 , O_2, O_3) використовуються в перших восьми раундах і в останніх восьми раундах в різних порядках. У перші вісім раундів ми додаємо O_1 і O_2до першого і другого цільовим речі, відповідно, і XOR O_3 у третьому цільовим слову. За останні вісім раундів, ми додаємо O_1 і O_2 до третього і другого цільовим речі, відповідно, і XOR O_3 до першого цільовим слову.

E-функція

E-функція приймає як вхідні дані одне слово даних і використовує ще два ключових слова, виробляючи на виході три слова. У цій функції ми використовуємо три тимчасові змінні, що позначаються L, M і R (для лівої, середньої та правої).

Спочатку ми встановлюємо в R значення вихідного слова зміщеного на 13 біт вліво, а в M – сума вихідних слів і першого ключового слова. Потім ми використовуємо перші дев'ять бітів M як індекс до однієї з 512S-блоків (яке виходить суміщенням S_0 і S_1 змішуванням фази), і зберігаємо в L значення відповідного S-блоку.

Потім помножимо друге ключове слово на R, зберігши значення в R. Потім обертаємо R на 5 позицій вліво (так, 5 старших бітів стають 5 нижніми бітами R після обертання). Тоді ми робимо XOR R в L, а також переглядаємо п'ять нижніх біт R для визначення величини зсуву (від 0 до 31), і обертаємо M

вліво на цю величину. Далі ми обертаємо R ще на 5 позицій вліво і робимо XOR в L. У висновку, ми знову дивимося на 5 молодших бітів R, як на величину обертання і обертаємо L на цю величину вліво. Таким чином результат роботи E-функції – 3 слова (по порядку): L, M, R.

Зворотне перемішування

Зворотне перемішування практично збігається з прямим перемішуванням, за винятком того факту, що дані обробляються в зворотному порядку. Тобто, якби ми поєднали пряме і зворотне перемішування так, щоб їх виходи і входи були б з'єднані в зворотному порядку (D [0] прямого і D [3] зворотного, D [1] прямого і D [2] зворотного), то не побачили б результату перемішування. Як і в прямому змішування, тут ми теж використовуємо одне вихідне слово і три цільових. Розглянемо чотири перших байта вихідного слова: b_0 , b_1 , b_2, b_3 . Будемо використовувати b_0 , b_2як індекс до S-блоку – S_1 , а b_1 b_3 для S_0 . Зробимо XOR S_1 [b_0] в перше цільове слово, віднімемо S_0 [b_3] з другого слова, віднімемо S_1 [b_2] з третього цільового слів і потім проробимо XOR S_0 [b_1] також до третього цільового слова. Нарешті, ми повертаємо початкове слово на 24 позицій вліво. Для наступного раунду ми обертаємо наявні слова так, щоб нинішнє перше цільове слово стало наступним вихідним словом, поточне друге цільове слово стало першим цільовим словом, поточне третє цільове слово стало другим цільовим словом, і поточне вихідне слово стало третім цільовим словом. Крім того, перед одним з чотирьох «особливих» раундів ми віднімаємо одне з цільових слів з вихідного слова: перед четвертим і восьмим раундами ми віднімаємо першої цільової слово, перед третьому і сьомим раундами ми віднімемо третя цільова слово з вихідного.

Дешифрування

Процес декодування обернений процесу кодування. Код дешифрування схожий (але не ідентичний) на код шифрування.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ мережевих сховищ на основі технології NVR яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Cloud параметри; Інстументарій користувача; Архівні дані; Довідка.

– Функції представлені у графічному вигляді – блоку керування камерою.

– Розділу виведення результату роботи системи – вікно камери.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

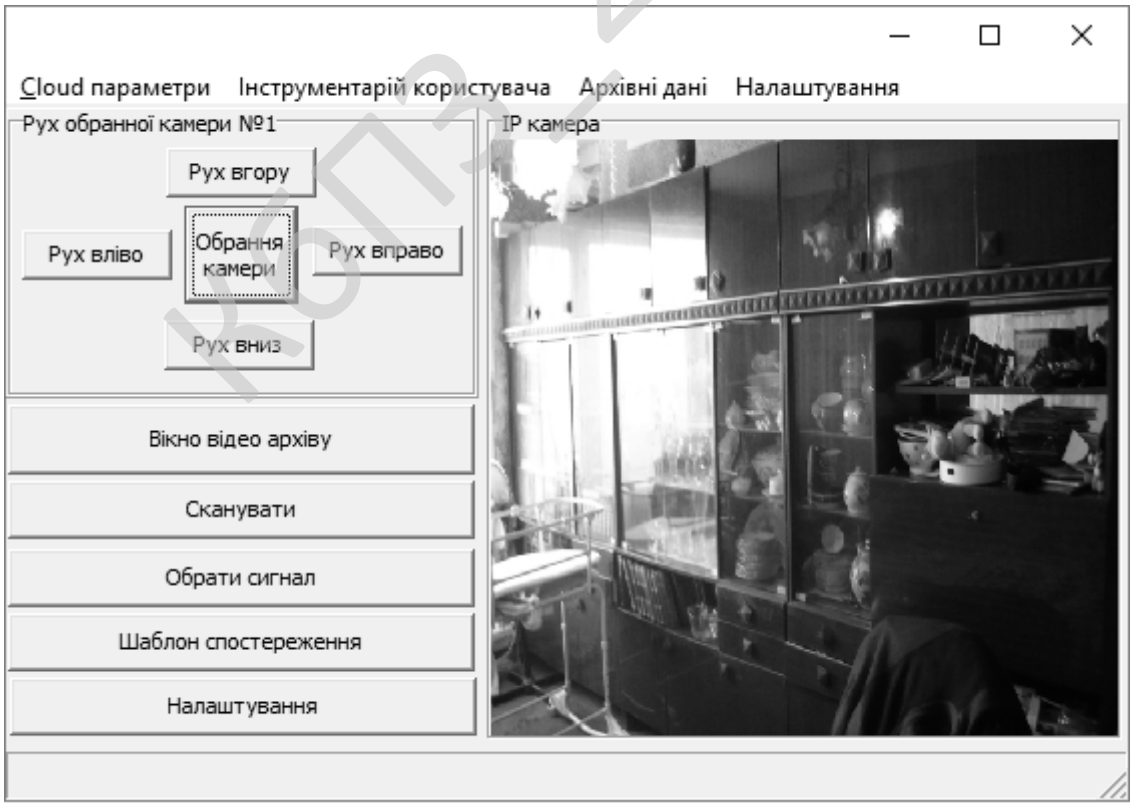

Рисунок 5.1 – Головне вікно ПЗ

Система призначена для реалізації мережних сховищ на основі технології NVR. ПЗ мережевих сховищ на основі технології NVR досить просто вирішують ряд завдань, пов'язаних зі зберіганням записів, при роботі з відеореєстраторами. За допомогою NAS можна розширити ємність відеоархіву реєстратора, а також підвищити його надійність шляхом резервування даних.

Система NAS + IP-камера має місце бути тільки тоді, коли сховище вже є й використовується не тільки для зберігання записів з камер, але й для зберігання іншої інформації. Тобто якщо вибирати між покупкою спеціалізованого реєстратора й NAS тільки під систему відеоспостереження, те мережні сховища програють реєстраторам як за ціною, так і по функціоналу.

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

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

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

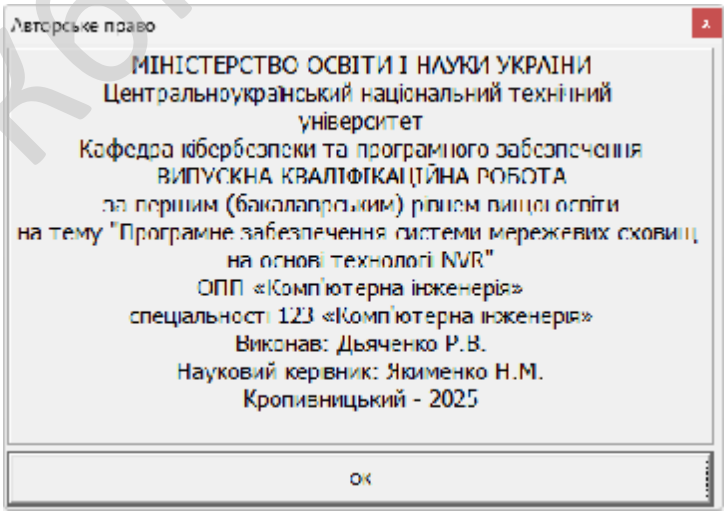

Рисунок 5.2 – Авторське право

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

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Видалення 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

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.

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до

кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

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

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

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

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

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи мережевих сховищ на основі технології NVR.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем мережевих сховищ на основі технології NVR.

– Досліджена система мережевих сховищ на основі технології NVR.

– На основі отриманих результатів досліджень створена програмна реалізація системи мережевих сховищ на основі технології NVR.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання мережевих сховищ на основі технології NVR.

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 мережевих сховищ на основі технології NVR. 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 MARS.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

2. 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.

3. 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.

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

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

6. 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.

7. 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

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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,

15. 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.

16. 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

17. 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.

18. 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.

19. 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.

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

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

39. 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.

40. 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.

41. 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.

42. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

43. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

44. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

45. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

46. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для

моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

47. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

48. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

49. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

50. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". - Випуск 2 (118). т.2. - Х.: ХУПС - 2014. - С. 64-67

51. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції "Проблеми та перспективи розвитку ІТ-індустрії". м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. - 2014. - С. 240.

52. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник. – Кіровоград: КНТУ 2013. – 257с.

53. Смірнов О.А., Дрєєв О.М., Доренський О.П. «Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі. Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи мережевих сховищ на основі технології NVR.

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. № 48-02 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 мережевих сховищ на основі технології NVR.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи мережевих сховищ на основі технології NVR;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

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 – 81 аркуш.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 9.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи мережевих сховищ на основі технології NVR

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 23

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
import os
import threading
import sqlite3
import time
import json
import socket
import http.server
import socketserver
import hashlib
import shutil
import io
import base64
from pathlib import Path
from datetime import datetime
import urllib.parse

class ConfigManager:
    def __init__(self, path):
        self.path=path
        self.config={}
        self.load()
    def load(self):
        if os.path.exists(self.path):
            with open(self.path,"r") as f:
                self.config=json.loads(f.read())
        else:
            default={
                "storage_path":"./data",
                "db_path":"./nvr.db",
                "host":"0.0.0.0",
                "port":8000,
                "roles":{

"admin":["upload","list","play","search","delete","stream","download_fec"],
                    "user":["upload","list","play","search","stream"]
                },
                "fec_block_size":1024,
                "smtp":{
                    "host":"smtp.example.com",
                    "port":587,
                    "tls":True,
                    "user":"user@example.com",
                    "password":"password",
                    "from":"noreply@example.com"
                },
                "twilio":{
                    "sid":"YOUR_TWILIO_SID",
                    "token":"YOUR_TWILIO_TOKEN",
                    "from":"+1234567890"
                },
                "recipients":{
                    "email":["admin@example.com"],
                    "sms":["+19876543210"]
                }
            }
            self.config=default
            with open(self.path,"w") as f:
                f.write(json.dumps(self.config,indent=4))

class DBManager:
```

```
    def __init__(self,path):
        self.path=path
        self.conn=sqlite3.connect(self.path,check_same_thread=False)
        self.cursor=self.conn.cursor()
        self.init_tables()
    def init_tables(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS videos(id INTEGER
PRIMARY KEY AUTOINCREMENT,filename TEXT,path TEXT,timestamp INTEGER)")
        self.cursor.execute("CREATE TABLE IF NOT EXISTS users(id INTEGER PRIMARY
KEY AUTOINCREMENT,username TEXT UNIQUE,password_hash TEXT,role TEXT)")
        self.conn.commit()
    def add_user(self,username,password_hash,role):
        try:
            self.cursor.execute("INSERT INTO users(username,password_hash,role)
VALUES(?,?,?)",(username,password_hash,role))
            self.conn.commit()
            return True
        except sqlite3.IntegrityError:
            return False
    def get_user(self,username):
        self.cursor.execute("SELECT password_hash,role FROM users WHERE
username=?",(username,))
        row=self.cursor.fetchone()
        return row if row else (None,None)
    def add_video(self,filename,path,timestamp):
        self.cursor.execute("INSERT INTO videos(filename,path,timestamp)
VALUES(?,?,?)",(filename,path,timestamp))
        self.conn.commit()
    def list_videos(self):
        self.cursor.execute("SELECT id,filename,path,timestamp FROM videos")
        return self.cursor.fetchall()
    def get_video(self,id):
        self.cursor.execute("SELECT filename,path FROM videos WHERE id=?",(id,))
        return self.cursor.fetchone()
    def delete_video(self,id):
        self.cursor.execute("SELECT path FROM videos WHERE id=?",(id,))
        row=self.cursor.fetchone()
        if row:
            os.remove(row[0])
            self.cursor.execute("DELETE FROM videos WHERE id=?",(id,))
            self.conn.commit()
            return True
        return False

class AuthManager:
    def __init__(self,db,roles):
        self.db=db
        self.roles=roles
    def register(self,username,password,role):
        ph=hashlib.sha256(password.encode()).hexdigest()
        return self.db.add_user(username,ph,role)
    def authenticate(self,header):
        if not header or not header.startswith("Basic "):
            return (None,None)
        token=header.split(" ",1)[1]
        data=base64.b64decode(token).decode()
        if ":" not in data:
            return (None,None)
        username,password=data.split(":",1)
        stored,role=self.db.get_user(username)
        if stored and hashlib.sha256(password.encode()).hexdigest()==stored:
            return (username,role)
        return (None,None)
```

```
    def has_permission(self,role,op):
        return op in self.roles.get(role,[])

class FECManager:
    def __init__(self,block_size):
        self.block=block_size
    def encode(self,path):
        data=open(path,"rb").read()
        blocks=[data[i:i+self.block] for i in range(0,len(data),self.block)]
        parity=bytearray(self.block)
        for b in blocks:
            for i in range(len(b)):
                parity[i]^=b[i]
        base=path+".fec"
        with open(base,"wb") as f:
            f.write(len(blocks).to_bytes(4,"big"))
            for b in blocks:
                f.write(len(b).to_bytes(4,"big"))
                f.write(b)
            f.write(parity)
        return base
    def decode(self,path):
        raw=open(path,"rb").read()
        count=int.from_bytes(raw[0:4],"big")
        idx=4
        blocks=[]
        for _ in range(count):
            size=int.from_bytes(raw[idx:idx+4],"big");idx+=4
            blocks.append(raw[idx:idx+size]);idx+=size
        parity=raw[idx:]
        missing=None
        for i,b in enumerate(blocks):
            if len(b)<self.block:
                missing=i
        if missing is not None:
            rec=bytearray(self.block)
            for i in range(self.block):
                xor=parity[i]
                for j,b in enumerate(blocks):
                    if j!=missing and i<len(b):
                        xor^=b[i]
                rec[i]=xor
            blocks[missing]=bytes(rec)
        data=b"".join(blocks)
        out=path.replace(".fec","")
        open(out,"wb").write(data)
        return out

class StorageManager:
    def __init__(self,path):
        self.path=path
        Path(self.path).mkdir(parents=True,exist_ok=True)
    def save(self,filename,data):
        p=os.path.join(self.path,filename)
        open(p,"wb").write(data)
        return p

class NotificationManager:
    def __init__(self,smtp,twilio,recipients):
        self.smtp=smtp
        self.twilio=twilio
        self.recipients=recipients
    def notify(self,event,message):
```

```
        import smtplib
        from email.mime.text import MIMEText
        from twilio.rest import Client
        sub=f"Event {event}"
        msg=MIMEText(message)
        msg['Subject']=sub
        msg['From']=self.smtp['from']
        msg['To']=",".join(self.recipients['email'])
        s=smtplib.SMTP(self.smtp['host'],self.smtp['port'])
        if self.smtp.get('tls'):s.starttls()
        s.login(self.smtp['user'],self.smtp['password'])
        s.sendmail(self.smtp['from'],self.recipients['email'],msg.as_string())
        client=Client(self.twilio['sid'],self.twilio['token'])
        for num in self.recipients['sms']:

client.messages.create(body=f"{sub}:{message}",from_=self.twilio['from'],to=num)

class VideoRecorder:
    def __init__(self,storage,db,fec,notifier):
        self.storage=storage
        self.db=db
        self.fec=fec
        self.notifier=notifier
    def record(self,stream_id):
        try:
            name=f"rec_{stream_id}_{int(time.time())}.dat"
            data=os.urandom(1024*512)
            path=self.storage.save(name,data)
            self.db.add_video(name,path,int(time.time()))
            self.fec.encode(path)
            self.notifier.notify("record_started",name)
        except Exception as e:
            self.notifier.notify("record_error",str(e))

class NVRRequestHandler(http.server.BaseHTTPRequestHandler):
    def do_POST(self):

user,role=self.server.auth.authenticate(self.headers.get("Authorization"))
        if not user: return self._una()
        parsed=urllib.parse.urlparse(self.path)
        if parsed.path=="/upload" and
self.server.auth.has_permission(role,"upload"):
            length=int(self.headers.get("Content-Length",0))
            data=self.rfile.read(length)
            name=f"upl_{int(time.time())}.dat"
            path=self.server.storage.save(name,data)
            self.server.db.add_video(name,path,int(time.time()))
            return self._res(200,{"status":"ok"})
        if parsed.path=="/delete" and
self.server.auth.has_permission(role,"delete"):
            q=urllib.parse.parse_qs(parsed.query)
            vid=int(q.get("id",[0])[0])
            ok=self.server.db.delete_video(vid)
            return self._res(200,{"deleted":ok})
        return self._una()
    def do_GET(self):

user,role=self.server.auth.authenticate(self.headers.get("Authorization"))
        if not user: return self._una()
        parsed=urllib.parse.urlparse(self.path)
        if parsed.path=="/videos" and
self.server.auth.has_permission(role,"list"):
            vids=self.server.db.list_videos()
```

```
            data=[{"id":v[0],"filename":v[1],"timestamp":v[3]} for v in vids]
            return self._res(200,{"videos":data})
        if parsed.path=="/play" and
self.server.auth.has_permission(role,"play"):
            q=urllib.parse.parse_qs(parsed.query)
            vid=int(q.get("id",[0])[0])
            v=self.server.db.get_video(vid)
            if not v: return self._res(404,{"error":"not found"})
            html=f"""<html><body><video width="640" height="480"
controls><source src="/stream?id={vid}"
type="video/mp4"></video></body></html>"""
            self.send_response(200);self.send_header("Content-
Type","text/html");self.send_header("Content-
Length",str(len(html)));self.end_headers();self.wfile.write(html.encode());retur
n
        if parsed.path=="/stream" and
self.server.auth.has_permission(role,"stream"):
            q=urllib.parse.parse_qs(parsed.query)
            vid=int(q.get("id",[0])[0])
            v=self.server.db.get_video(vid)
            if not v: return self._res(404,{"error":"not found"})
            data=open(v[1],"rb").read()
            self.send_response(200);self.send_header("Content-
Type","application/octet-stream");self.send_header("Content-
Length",str(len(data)));self.end_headers();self.wfile.write(data);return
        if parsed.path=="/download-fec" and
self.server.auth.has_permission(role,"download_fec"):
            q=urllib.parse.parse_qs(parsed.query)
            vid=int(q.get("id",[0])[0])
            v=self.server.db.get_video(vid)
            if not v: return self._res(404,{"error":"not found"})
            eb=self.server.fec.encode(v[1])
            data=open(eb,"rb").read()
            self.send_response(200);self.send_header("Content-
Type","application/octet-stream");self.send_header("Content-
Disposition",f"attachment;filename={os.path.basename(eb)}");self.send_header("Co
ntent-Length",str(len(data)));self.end_headers();self.wfile.write(data);return
        return self._una()
    def _res(self,code,obj):
        b=json.dumps(obj).encode()
        self.send_response(code);self.send_header("Content-
Type","application/json");self.send_header("Content-
Length",str(len(b)));self.end_headers();self.wfile.write(b)
    def _una(self):
        self.send_response(401);self.end_headers()

class ThreadedServer(socketserver.ThreadingMixIn,http.server.HTTPServer):
    daemon_threads=True
    def __init__(self,addr,handler,auth,storage,db,fec):
        super().__init__(addr,handler)
        self.auth=auth;self.storage=storage;self.db=db;self.fec=fec

def main():
    cfg=ConfigManager("config.json")
    storage=StorageManager(cfg.config["storage_path"])
    db=DBManager(cfg.config["db_path"])
    auth=AuthManager(db,cfg.config["roles"])
    fec=FECManager(cfg.config["fec_block_size"])

notifier=NotificationManager(cfg.config["smtp"],cfg.config["twilio"],cfg.config[
"recipients"])
    recorder=VideoRecorder(storage,db,fec,notifier)
    threading.Thread(target=lambda:recorder.record("main"),daemon=True).start()
```

```
server=ThreadedServer((cfg.config["host"],cfg.config["port"]),NVRRequestHandler,
auth,storage,db,fec)
    server.serve_forever()

if __name__=="__main__":
    main()

import requests
import base64
BASE="http://localhost:8000"
def auth_header(u,p):
    token=base64.b64encode(f"{u}:{p}".encode()).decode()
    return {"Authorization":f"Basic {token}"}
def list_videos(u,p):
    r=requests.get(BASE+"/videos",headers=auth_header(u,p))
    return r.json()
def upload_video(u,p,path):
    data=open(path,"rb").read()
    return
requests.post(BASE+"/upload",headers=auth_header(u,p),data=data).json()
def delete_video(u,p,vid):
    return
requests.post(f"{BASE}/delete?id={vid}",headers=auth_header(u,p)).json()
def play_link(vid):
    return f"{BASE}/play?id={vid}"
def download_stream(u,p,vid,path):
    r=requests.get(f"{BASE}/stream?id={vid}",headers=auth_header(u,p))
    open(path,"wb").write(r.content)
def download_fec(u,p,vid,path):
    r=requests.get(f"{BASE}/download-fec?id={vid}",headers=auth_header(u,p))
    open(path,"wb").write(r.content)
if __name__=="__main__":
    user="admin";password="admin"
    vids=list_videos(user,password)
    if vids.get("videos"):
        vid=vids["videos"][0]["id"]
        download_stream(user,password,vid,"out.dat")
        download_fec(user,password,vid,"out.dat.fec")
```

Файл camera_s.py

```
import datetime as dts
from system.shared import LastErrorHolder
import time
import cv2 as cv


class CameraConnectionSupport(LastErrorHolder):
    def __init__(self, camConnectionString, logger):
        LastErrorHolder.__init__(self)
        self.logger = logger

        # date & time when class instance created
        self.started = dts.datetime.utcnow()

        # camera
        self.cap = None

        # frame height
        self.frameHeight = None

        # frame width
        self.frameWidth = None

        # total count of pixels
        self.nb_pixels = None

        self.camConnectionString = camConnectionString

    def utcNow(self):
        return dts.datetime.utcnow()

    def onFrameSizeUpdate(self, frameWidth, frameHeight):
        """
        Will be called when frame size will be updated or initialized

        :param frameWidth: new width
        :param frameHeight: new height
        :return:
        """

        pass

    def _initCamera(self, callSleep = True):
        """
        Initializes camera. If can't establish connection will write error
message to log file and sleep for some
        interval.

        :return: True when camera successfully open, otherwise False
        """
        self.cap = cv.VideoCapture(self.camConnectionString)

        if self.cap is None:
            self.setError("can't connect to camera")
            if callSleep:
                time.sleep(5)
            return None

        if not self.cap.isOpened():  # did we get a connection at all ?
            self.setError("can't connect to camera")
            if callSleep:
```

```
                time.sleep(5)

            return None

        return self.cap
```

Файл log_support.py

```
import config
from system.shared import mkdir_p

import logging
from logging.handlers import RotatingFileHandler
import os
import sys


def init_logger(mainLoggerName = __name__):
    logger = logging.getLogger(mainLoggerName)

    logsDirPath = os.path.dirname(config.LOG_FILE_PATH)
    if not os.path.exists(logsDirPath):
        if not mkdir_p(logsDirPath):
            print("ERROR INITIALIZING! Can't create directory for logs:
'{}'".format(logsDirPath))
            exit(-1)

    # create file handler
    handler = RotatingFileHandler(
        config.LOG_FILE_PATH,
        encoding="utf-8",
        maxBytes=config.MAIN_LOG_FILE_MAX_SIZE,
        backupCount=config.LOG_BACKUPS_COUNT
    )
    handler.setLevel(logging.DEBUG)

    # create formatter
    formatter = logging.Formatter(config.LOG_FORMAT)
    handler.setFormatter(formatter)
    logger.addHandler(handler)

    if config.LOG_TO_CONSOLE:
        consoleHandler = logging.StreamHandler(sys.stdout)
        consoleHandler.setLevel(config.APP_LOG_LEVEL)
        consoleHandler.setFormatter(formatter)
        logger.addHandler(consoleHandler)

    logger.setLevel(config.APP_LOG_LEVEL)
    return logger
```

Файл MotionD.py

```
import cv2 as cv
from system.shared import LastErrorHolder
import imutils
import numpy as np
import datetime


class MotionDetectorBase(LastErrorHolder):
    """
    Base class for motion detection support
    """
    def __init__(self):
        LastErrorHolder.__init__(self)
        # previous frame
        self.prevFrame = None

        # DTS (date & time) of moment when last motion was detected
        self.motionDetectionDts = None

        self.resizeBeforeDetect = True

        self.multiFrameDetection = False

    def preprocessInputFrame(self, newFrame):
        if self.resizeBeforeDetect:
            return imutils.resize(newFrame, width=500, height=500)

        return newFrame.copy()

    def checkMotionDetected(self, frame):
        """
        Checks that motion detected.

        :param frame: new frame from camera
        :return: True when motion detected, otherwise False
        """
        return False

    def updateMotionDetectionDts(self):
        self.motionDetectionDts = datetime.datetime.utcnow()


class MotionDetectorV1(MotionDetectorBase):
    def __init__(self):
        MotionDetectorBase.__init__(self)
        self.threshold = 8

    def motionDetected(self, new_frame):
        frame = self.preprocessInputFrame(new_frame)

        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        gray = cv.GaussianBlur(gray, (21, 21), 0)

        if self.prevFrame is None:
            self.prevFrame = gray
            return False

        # compute the absolute difference between the current frame and
        # first frame
        frameDelta = cv.absdiff(gray, self.prevFrame)
        thresh = cv.threshold(frameDelta, 25, 255, cv.THRESH_BINARY)[1]
```

```
        # dilate the thresholded image to fill in holes, then find contours
        # on thresholded image
        thresh = cv.dilate(thresh, None, iterations=2)
        (cnts,  _,  _)  =  cv.findContours(thresh.copy(),  cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)

        height = np.size(gray, 0)
        width = np.size(gray, 1)
        nb = height * width

        qty = 0
        for c in cnts:
            a = cv.boundingRect(c)

            (x, y, w, h) = a
            s = w * h

            pcs = (float(s) / float(nb)) * 100

            if pcs < self.threshold:
                continue

            # cv.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

            qty += 1
            break

        # cv.imshow("frame", frame)
        self.prevFrame = gray

        ret = (qty > 0)
        if ret:
            self.updateMotionDetectionDts()

        return ret


class MotionDetectorV2(MotionDetectorBase):
    def __init__(self):
        MotionDetectorBase.__init__(self)
        self.threshold = 1

    def motionDetected(self, new_frame):
        frame = self.preprocessInputFrame(new_frame)

        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        gray = cv.GaussianBlur(gray, (21, 21), 0)

        if self.prevFrame is None:
            self.prevFrame = gray
            return False

        frameDiff = cv.absdiff(gray, self.prevFrame)

        # kernel = np.ones((5, 5), np.uint8)

        opening = cv.morphologyEx(frameDiff, cv.MORPH_OPEN, None)  # noqa
        closing = cv.morphologyEx(frameDiff, cv.MORPH_CLOSE, None)  # noqa

        ret1, th1 = cv.threshold(frameDiff, 10, 255, cv.THRESH_BINARY)

        height = np.size(th1, 0)
```

```
        width = np.size(th1, 1)

        nb = cv.countNonZero(th1)

        avg = (nb * 100) / (height * width)  # Calculate the average of black
pixel in the image

        self.prevFrame = gray

        # cv.DrawContours(currentframe, self.currentcontours, (0, 0, 255), (0,
255, 0), 1, 2, cv.CV_FILLED)
        # cv.imshow("frame", current_frame)

        ret = avg > self.threshold   # If over the ceiling trigger the alarm

        if ret:
            self.updateMotionDetectionDts()

        return ret


class MotionDetectorV3(MotionDetectorBase):
    def __init__(self):
        MotionDetectorBase.__init__(self)

        self.threshold = 1000
        self.prevPrevFrame = None

    def diffImg(self, t0, t1, t2):
        d1 = cv.absdiff(t2, t1)
        d2 = cv.absdiff(t1, t0)
        return cv.bitwise_and(d1, d2)

    def motionDetected(self, new_frame):
        frame = self.preprocessInputFrame(new_frame)

        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        gray = cv.GaussianBlur(gray, (11, 11), 0)

        if self.prevPrevFrame is None:
            self.prevPrevFrame = gray
            return False

        if self.prevFrame is None:
            self.prevFrame = gray
            return False

        cv.normalize(gray, gray, 0, 255, cv.NORM_MINMAX)

        frameDiff = self.diffImg(self.prevPrevFrame, self.prevFrame, gray)
        ret1, th1 = cv.threshold(frameDiff, 10, 255, cv.THRESH_BINARY)

        cv.dilate(th1, None, iterations=15)
        cv.erode(th1, None, iterations=1)

        delta_count = cv.countNonZero(th1)

        cv.imshow("frame_th1", th1)

        self.prevPrevFrame = self.prevFrame
        self.prevFrame = gray

        ret = delta_count > self.threshold
```

```
        if ret:
            self.updateMotionDetectionDts()

        return ret


class MotionDetector(MotionDetectorBase):
    def __init__(self):
        MotionDetectorBase.__init__(self)

        self.threshold = 1500
        self.prevPrevFrame = None

    def diffImg(self, t0, t1, t2):
        if not self.multiFrameDetection:
            return cv.absdiff(t2, t1)

        d1 = cv.absdiff(t2, t1)
        d2 = cv.absdiff(t1, t2)
        return cv.bitwise_and(d1, d2)

    def motionDetected(self, new_frame):
        frame = self.preprocessInputFrame(new_frame)

        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        gray = cv.GaussianBlur(gray, (11, 11), 0)

        if (self.multiFrameDetection) and (self.prevPrevFrame is None):
            self.prevPrevFrame = gray
            return False

        if self.prevFrame is None:
            self.prevFrame = gray
            return False

        cv.normalize(gray, gray, 0, 255, cv.NORM_MINMAX)

        frameDiff = self.diffImg(self.prevPrevFrame, self.prevFrame, gray)
        ret1, th1 = cv.threshold(frameDiff, 10, 255, cv.THRESH_BINARY)

        th1 = cv.dilate(th1, None, iterations=8)
        th1 = cv.erode(th1, None, iterations=4)

        delta_count = cv.countNonZero(th1)

        if self.multiFrameDetection:
            self.prevPrevFrame = self.prevFrame

        self.prevFrame = gray
        if delta_count < self.threshold:
            return False

        if self.multiFrameDetection:
            self.prevPrevFrame = self.prevFrame

        self.prevFrame = gray
        self.updateMotionDetectionDts()
        return True


class MotionDetectorV3Traced(MotionDetectorBase):
    def __init__(self):
```

```
        MotionDetectorBase.__init__(self)

        self.threshold = 1500
        self.prevPrevFrame = None

        self.produceContoursFrame = False
        self.contoursFrame = None

        self.productDiffFrame1 = False
        self.diffFrame1 = None

        self.productDiffFrame2 = False
        self.diffFrame2 = None

    def diffImg(self, t0, t1, t2):
        if not self.multiFrameDetection:
            return cv.absdiff(t2, t1)

        d1 = cv.absdiff(t2, t1)
        d2 = cv.absdiff(t1, t2)
        return cv.bitwise_and(d1, d2)

    def motionDetected(self, new_frame):
        frame = self.preprocessInputFrame(new_frame)

        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        gray = cv.GaussianBlur(gray, (11, 11), 0)

        if (self.multiFrameDetection) and (self.prevPrevFrame is None):
            self.prevPrevFrame = gray
            return False

        if self.prevFrame is None:
            self.prevFrame = gray
            return False

        cv.normalize(gray, gray, 0, 255, cv.NORM_MINMAX)

        frameDiff = self.diffImg(self.prevPrevFrame, self.prevFrame, gray)
        ret1, th1 = cv.threshold(frameDiff, 10, 255, cv.THRESH_BINARY)

        if self.productDiffFrame1:
            self.diffFrame1 = th1.copy()

        th1 = cv.dilate(th1, None, iterations=8)
        th1 = cv.erode(th1, None, iterations=4)

        if self.productDiffFrame2:
            self.diffFrame2 = th1.copy()

        delta_count = cv.countNonZero(th1)

        if self.multiFrameDetection:
            self.prevPrevFrame = self.prevFrame

        self.prevFrame = gray
        if delta_count < self.threshold:
            return False

        if self.produceContoursFrame:
            self.contoursFrame = frame.copy()
```

```
            im2, contours, hierarchy = cv.findContours(th1, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)
            for c in contours:
                cv.drawContours(self.contoursFrame, [c], 0, (0, 0, 255), 2)

            # (x, y, w, h) = cv.boundingRect(c)
            # cv.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        if self.multiFrameDetection:
            self.prevPrevFrame = self.prevFrame

        self.prevFrame = gray

        self.updateMotionDetectionDts()
        return True


class MotionDetectorV4(MotionDetectorBase):
    def __init__(self):
        MotionDetectorBase.__init__(self)

        self.threshold = 10
        self.prevPrevFrame = None

    def diffImg(self, t0, t1, t2):
        d1 = cv.absdiff(t2, t1)
        d2 = cv.absdiff(t1, t0)
        return cv.bitwise_and(d1, d2)

    def motionDetected(self, new_frame):
        frame = self.preprocessInputFrame(new_frame)

        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        gray = cv.GaussianBlur(gray, (11, 11), 0)

        if self.prevPrevFrame is None:
            self.prevPrevFrame = gray
            return False

        if self.prevFrame is None:
            self.prevFrame = gray
            return False

        cv.normalize(gray, gray, 0, 255, cv.NORM_MINMAX)

        frameDiff = self.diffImg(self.prevPrevFrame, self.prevFrame, gray)
        ret1, th1 = cv.threshold(frameDiff, 10, 255, cv.THRESH_BINARY)

        cv.dilate(th1, None, iterations=4)
        cv.erode(th1, None, iterations=2)

        totalArea = 0
        im2, contours, hierarchy = cv.findContours(th1, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)
        for c in contours:
            totalArea += cv.contourArea(c)
            cv.drawContours(frame, [c], 0, (0, 0, 255), 2)

        if totalArea < self.threshold:
            return False

        cv.imshow("frame_th1", frame)
```

```
        self.prevPrevFrame = self.prevFrame
        self.prevFrame = gray
        self.updateMotionDetectionDts()

        return True
```

Файл Xshared.py

```
import os
import errno
from config import APP_ROOT


class LastErrorHolder:
    """
    Support holding of last error
    """
    def __init__(self):
        self.errorText  = ""
        self.__hasError = False

    def clearError(self):
        """
        Clears last error
        :return: None
        """
        self.errorText  = ""
        self.__hasError = False

    def setError(self, errorText):
        """
        Sets new error
        :param errorText: new error text
        :return: False
        """
        self.errorText   = errorText
        self.__hasError  = True

        return False

    @property
    def hasError(self):
        """
        Holds error flag.
        :return: True when have error, otherwise False
        """
        return self.__hasError


def mkdir_p_ex(path):
    try:
        if os.path.exists(path) and os.path.isdir(path):
            return (True, None)

        os.makedirs(path)
        return (True, None)
    except OSError as exc:  # Python >2.5
        if exc.errno == errno.EEXIST and os.path.isdir(path):
            return (True, None)
        else:
            return (False, str(exc))
    except Exception as e:
```

```
        return (False, str(e))


def mkdir_p(path):
    (ok, stub) = mkdir_p_ex(path)
    return ok


def makeAbsoluteAppPath(path, basePath = None):
    """
    Converts relative path to absolute.

    :param path: relative path
    :param basePath: base path (by default - current app directory)
    :return: absolute path
    """
    if os.path.isabs(path):
        return path

    if basePath is None:
        basePath = APP_ROOT

    path = os.path.join(basePath, path)
    return os.path.abspath(os.path.normpath(path))


def subFolderNameForDtsGeneratorFunc(dts):
    return "{:04}/{:02}/{:02}".format(dts.year, dts.month, dts.day)
```

Файл recorder.py

```
import os

import time
import cv2 as cv
from system.motion_detection import MotionDetector
import imutils
import datetime as dts
import numpy as np
from system.camera_support import CameraConnectionSupport
import config
from system.shared import mkdir_p
import Queue
import uuid


class QueueCommand:
    CMD_QUIT_THREAD = "quit"

    def __init__(self, cmd):
        self.cmd = cmd
        self.uid = str(uuid.uuid4())


class MotionDrivenRecorder(CameraConnectionSupport):
    def __init__(self, camConnectionString, logger):
        CameraConnectionSupport.__init__(self, camConnectionString, logger)

        # initializing motion detector
        self.detector = MotionDetector()
        self.detector.resizeBeforeDetect = False
        self.detector.multiFrameDetection = False

        self.inMotionDetectedState = False

        self._camConnectionDts = None
        self._canDetectMotion = False
        self.camFps = None

        self.preAlarmRecordingSecondsQty = 0

        self._savedFrames = []

        self._isRecording = False

        # output writer
        self.outputDirectory = None
        self._output = None

        self.subFolderNameGeneratorFunc = None
        self._prevSubFolder = None
        self.scaleFrameTo = None


        self._messages_queue = Queue.Queue()

        self._quit = False

    def add_stop_request(self):
        cmd = QueueCommand(QueueCommand.CMD_QUIT_THREAD)
        self.logger.info("adding quit command with uid = {}".format(cmd.uid))
        self._messages_queue.put(cmd)
```

```
    def _addPreAlarmFrame(self, frame):
        if self.preAlarmRecordingSecondsQty == 0:
            return

        if self.camFps is None:
            return

        totalQty = int(self.preAlarmRecordingSecondsQty * self.camFps)
        if len(self._savedFrames) < totalQty:
            self._savedFrames.append(frame)
            return

        if len(self._savedFrames) > 0:
            self._savedFrames.pop(0)

        self._savedFrames.append(frame)

    def canDetectMotion(self):
        if self._canDetectMotion:
            return True

        if self._camConnectionDts is None:
            return False

        minDts = self._camConnectionDts +
dts.timedelta(seconds=config.INITIAL_WAIT_INTERVAL_BEFORE_MOTION_DETECTION_SECS)

        if minDts > self.utcNow():
            return False

        self._canDetectMotion = True
        return True

    def setError(self, errorText):
        self.logger.error(errorText)
        return CameraConnectionSupport.setError(self, errorText)

    def _writeOutFrame(self, frame):
        assert self._output is not None
        self._output.write(frame)

    def _stopRecording(self):
        if self._output is None:
            return

        self._output.release()
        self._output = None

        self._isRecording = False

    def _getSubFolderName(self, dts):
        if self.subFolderNameGeneratorFunc is None:
            return None

        return self.subFolderNameGeneratorFunc(dts)

    def _startRecording(self):
        if self.outputDirectory is None:
            return self.setError("output directory is not specified")

        if None in [self.frameWidth, self.frameHeight]:
            return self.setError("resolution is't specified")
```

```
        fourcc = cv.VideoWriter_fourcc(*config.FOURCC_CODEC)
        videoSize = (self.frameWidth, self.frameHeight)

        # calculation output filename
        now = dts.datetime.utcnow()
        fileName = "video_{}{}".format(now.strftime("%Y%m%dT%H%M%S"),
config.OUTPUT_FILES_EXTENSION)

        subFolder = self._getSubFolderName(now)
        if subFolder is not None:
            needCreate = ((self._prevSubFolder is not None) or (subFolder !=
self._prevSubFolder))

            dirName = os.path.join(self.outputDirectory, subFolder)
            dirName = os.path.normpath(dirName)

            if (needCreate) and (not os.path.exists(dirName)):
                self.logger.info("adding new directory: {}".format(dirName))
                if not mkdir_p(dirName):
                    return self.setError("can't create sub-directory:
{}".format(dirName))

            fileName = os.path.join(dirName, fileName)
        else:
            fileName = os.path.join(self.outputDirectory, fileName)

        self._output = cv.VideoWriter(fileName, fourcc,
config.OUTPUT_FRAME_RATE, videoSize)

        self._isRecording = True
        return True

    def _flushPreRecordingFrames(self):
        """
        Writes pre-alarm frames to output file
        :return:
        """
        assert self._output is not None

        for frame in self._savedFrames:
            self._output.write(frame)

        self._savedFrames = []
        return True

    def _detect_motion(self, current_frame, instant):
        # detection motion if can do it now
        if not self.canDetectMotion():
            return False

        if not self.detector.motionDetected(current_frame):
            return False

        self.trigger_time = instant  # Update the trigger_time

        if not self.inMotionDetectedState:
            self.logger.info("something moved!")

        self.inMotionDetectedState = True
        return True

    def _process_queue_commands(self):
```

```
        if self._messages_queue.empty():
            return

        cmd = self._messages_queue.get()
        self.logger.info("got new command - {} [{}]".format(cmd.cmd, cmd.uid))

        if cmd.cmd == QueueCommand.CMD_QUIT_THREAD:
            self._quit = True
            return

        self.logger.error("unknown command: {} [{}]".format(cmd.cmd, cmd.uid))
        raise Exception("Unknown command")

    def start(self):  # noqa
        """
        Main loop for motion detection tester
        :return:
        """
        self.logger.info("main loop started")

        emptyFrame = None

        prev_logged_left_seconds = None

        bad_frames = 0

        while not self._quit:
            self._process_queue_commands()
            if self._quit:
                break

            if bad_frames > 100:
                if self.cap is not None:
                    self.cap.release()
                    self.cap = None
                    bad_frames = 0

            # initializing connection to camera
            if self.cap is None:
                self.logger.info("initializing connection to camera")

                if self._initCamera() is None:
                    self.logger.error("can't initialize connection to camera")
                    continue

                self._camConnectionDts = self.utcNow()

            ret, current_frame = self.cap.read()

            # if can't read current frame - going to the next loop
            if (not ret) or (current_frame is None):  # the connection broke, or
the stream came to an end
                self.logger.warning("bad frame")
                bad_frames += 1
                continue
            else:
                bad_frames = 0

            if self.scaleFrameTo is not None:
                current_frame = imutils.resize(current_frame,
width=self.scaleFrameTo[0], height=self.scaleFrameTo[1])

            # get timestamp of the frame
```

```
            instant = time.time()

            frameHeight = np.size(current_frame, 0)
            frameWidth = np.size(current_frame, 1)

            if self.camFps is None:
                self.camFps = self.cap.get(cv.CAP_PROP_FPS)
                self.logger.info("FPS = {}".format(self.camFps))

            # adding frame to pre-recording buffer
            if self.preAlarmRecordingSecondsQty > 0:
                self._addPreAlarmFrame(current_frame)

            if emptyFrame is None:
                emptyFrame = np.zeros((frameHeight, frameWidth, 3), np.uint8)

            resolutionChanged = False
            if None in [self.frameWidth, self.frameHeight]:
                self.frameWidth = frameWidth
                self.frameHeight = frameHeight
                self.nb_pixels = self.frameWidth * self.frameHeight

                self.logger.info("self.width = {}".format(self.frameWidth))
                self.logger.info("self.height = {}".format(self.frameHeight))

                resolutionChanged = True
            else:
                resolutionChanged = ((self.frameWidth != frameWidth) or
(self.frameHeight != frameHeight))

            if resolutionChanged:
                # TODO: need process when recording now, or will be exception!
                self.onFrameSizeUpdate(frameWidth, frameHeight)

            # detecting motion
            motionDetected = self._detect_motion(current_frame, instant)

            now = self.utcNow()
            # prolongating motion for minimal motion duration
            if (not motionDetected) and (self.detector.motionDetectionDts is not
None):
                minDuration = self.detector.motionDetectionDts +
dts.timedelta(seconds=config.MINIMAL_MOTION_DURATION)
                if minDuration > now:
                    motionDetected = True

            # clearing motion detection flag when needed
            if not motionDetected:
                self.inMotionDetectedState = False

                if self._isRecording:
                    self.logger.info("stopping recording...")
                    self._stopRecording()

            elif not self._isRecording:
                self.logger.info("starting recording...")
                self._startRecording()
                self._flushPreRecordingFrames()

            # calculating left seconds for motion (for further use in label)
            dx = 0
            if motionDetected:
                dx = now - self.detector.motionDetectionDts
```

```
                dx = config.MINIMAL_MOTION_DURATION - dx.seconds

                if (prev_logged_left_seconds != dx):
                    self.logger.info("left seconds for motion recording:
{}".format(dx))
                    prev_logged_left_seconds = dx

            # adding label for frame with detected motion
            if motionDetected:
                text = "MOTION DETECTED [{}]".format(dx)
                cv.putText(
                    current_frame,
                    text,
                    (10, 20),
                    cv.FONT_HERSHEY_SIMPLEX,
                    0.5,
                    (0, 0, 255),  # b g r
                    2
                )

            if self._isRecording:
                self._writeOutFrame(current_frame)

        # stop recording if now recording
        if self._isRecording:
            self._stopRecording()

        if self.cap is not None:
            self.cap.release()

        cv.destroyAllWindows()
        self.logger.info("main loop finished")
```