ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

Пояснювальна записка

до кваліфікаційної роботи

бакалавра
(освітньо-кваліфікаційний рівень)

на тему *Розробка комп'ютерної системи діагностики твердотільного накопичувача Kingston SSD A400*

Development of a computer diagnostic system for Kingston A400 240 GB solid-state drive

Виконав: студент *4* курсу, групи *4КСМ*
напряму підготовки (спеціальності)
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)
Жук К.О.
(прізвище та ініціали)
Керівник *Дроздова Є.А.*
(прізвище та ініціали)
Рецензент *Захарченко Р.М.*
(прізвище та ініціали)

Хмельницький – 2026 року

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Інститут, факультет, відділення *інформаційних технологій та дизайну*
Кафедра, циклова комісія *комп'ютерних систем та мереж*
Освітньо-кваліфікаційний рівень *бакалавр*
Напрям підготовки *12 Інформаційні технології*
(шифр і назва)
Спеціальність *123 «Комп'ютерна інженерія»*
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних систем та мереж*
Анжела ГРИГОРОВА _
«___» _____ 202__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Жук Кирило Олександрович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Розробка комп'ютерної системи діагностики твердотільного накопичувача Kingston SSD A400*
Development of a computer diagnostic system for Kingston A400 240 GB solid-state drive

керівник проекту (роботи) *Дроздова Є.А. старший викладач*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «13» січня 2026 року №48-с

2. Строк подання студентом проекту (роботи) 15 червня 2026 року

3. Вихідні дані до проекту (роботи) *Завдання на кваліфікаційну роботу, завдання і матеріали переддипломної практики*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Пристрій що діагностується, 2 Апаратна складова комп'ютерної системи діагностики, 3 Програмна складова комп'ютерної системи діагностики, 4 Проектування комп'ютерної мережі, 5 Проектування серверної кімнати, 6 кодування інформації

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Комп'ютерна мережа, блок-схема алгоритму шифрування, схема підключення комп'ютерної системи діагностики, схема алгоритму роботи програми системи діагностики

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Аналіз актуальності питання по темі кваліфікаційної роботи*	***01.05.2026***	
2	*Пошук та вивчення теоретичного матеріалу стосовно кваліфікаційної роботи*	***05.05.2026***	
3	*Проектування локальної комп'ютерної мережі*	***07.05.2026***	
4	*Розробка програми кодування інформації*	***10.05.2026***	
5	*Проектування серверної кімнати*	***15.05.2026***	
6	*Розробка пристрою діагностики*	***21.05.2026***	
7	*Розробка програмного забезпечення для мікроконтролеру*	***27.05.2026***	
8	*Розробка і оформлення креслень*	***03.06.2026***	
9	*Оформлення кваліфікаційної роботи*	***13.06.2026***	

Студент _____ Жук К.О.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Дроздова Є.А.*
(підпис) (прізвище та ініціали)

ВІДОМІСТЬ ОБСЯГУ КВАЛІФІКАЦІЙНОЇ РОБОТИ

№ п/п	Формат	Позначення	Найменування	Кількість	Шифр док-та	Примітки
1.	А4	123.22011	Реферат	1	РФ	
2.	А4	123.22011	Пояснювальна записка	92	КРПЗ	
3.	А1	123.22011	Схема електрична принципова	1	Е3	
4.	А1	123.22011	Кодування. Алгоритм	1	АГ	
5.	А1	123.22011	ЛКМ. Структурна схема мережі	1	Е1	
6.	А1	123.22011	ЛКМ. Адресний простір	1		
7.						
8.						
9.						
10.						
11.						
12.						
13.						

РЕФЕРАТ

Кваліфікаційна робота містить: 96 сторінок, 30 рисунків, 21 таблиць, 34 посилання, 2 додатки.

Об'єкт дослідження – процес діагностики технічного стану твердотільних накопичувачів.

Предмет дослідження – методи та засоби побудови комп'ютерної системи діагностики SSD-накопичувачів.

Метою роботи є розробка комп'ютерної системи діагностики твердотільного накопичувача Kingston SSD A400, яка забезпечує контроль параметрів його роботи, аналіз технічного стану та інформування користувача про результати діагностики.

У кваліфікаційній роботі виконано аналіз конструкції та технічних характеристик твердотільних накопичувачів, досліджено особливості роботи накопичувача Kingston SSD A400 та визначено основні параметри його діагностики.

Розроблено апаратну складову комп'ютерної системи діагностики на базі мікроконтролера ESP32-WROOM-32 із використанням OLED-дисплея та звукового модуля оповіщення. Виконано розробку структурної схеми системи та алгоритму її функціонування.

SSD, KINGSTON SSD A400, ДІАГНОСТИКА НАКОПИЧУВАЧІВ, ESP32, МІКРОКОНТРОЛЕР, OLED-ДИСПЛЕЙ, КОМП'ЮТЕРНА СИСТЕМА ДІАГНОСТИКИ, SMART-ПАРАМЕТРИ, СЕРВЕРНА КІМНАТА, КОМП'ЮТЕРНА МЕРЕЖА, PYTHON, КОД ХЕМІНГА, КОДУВАННЯ ІНФОРМАЦІЇ

ABSTRACT

Explanatory note to the qualification work: 96 pages, 30 figures, 21 tables, 34 references, 2 appendices.

Object of research – the process of diagnosing the technical condition of solid-state drives.

Subject of research – methods and tools for building a computer-based SSD diagnostic system.

The purpose of the work is to develop a computer diagnostic system for the Kingston SSD A400 solid-state drive capable of monitoring operating parameters, analyzing technical condition and informing the user about diagnostic results.

The qualification work includes an analysis of the design and technical characteristics of solid-state drives, investigation of the Kingston SSD A400 features and determination of the main diagnostic parameters used for condition assessment.

A hardware diagnostic subsystem based on the ESP32-WROOM-32 microcontroller with an OLED display and an audible notification module was developed. A structural diagram of the system and an operating algorithm were designed.

SSD, KINGSTON SSD A400, SSD DIAGNOSTICS, ESP32, MICROCONTROLLER, OLED DISPLAY, COMPUTER DIAGNOSTIC SYSTEM, SMART PARAMETERS, SERVER ROOM, COMPUTER NETWORK, PYTHON, HAMMING CODE, INFORMATION CODING

ЗМІСТ

ВСТУП

На сучасному етапі розвитку інформаційних технологій твердотільні накопичувачі (SSD) стали основним засобом довготривалого зберігання даних у персональних комп’ютерах, ноутбуках, серверних системах та вбудованих пристроях. Завдяки високій швидкодії, низькому енергоспоживанню та відсутності механічних елементів SSD практично витіснили традиційні жорсткі диски з більшості сфер застосування. Разом із тим зростання обсягів інформації та підвищення вимог до надійності зберігання даних обумовлюють необхідність постійного контролю технічного стану накопичувачів і своєчасного виявлення ознак їх деградації.

На сьогодні для оцінювання стану твердотільних накопичувачів використовуються спеціалізовані програмні засоби, зокрема CrystalDiskInfo, Smartmontools, Kingston SSD Manager та інші. Вони дозволяють отримувати службову інформацію про накопичувачі, аналізувати параметри SMART та виконувати моніторинг основних експлуатаційних характеристик. Разом із тим більшість існуючих рішень орієнтована виключно на програмну реалізацію та не передбачає використання спеціалізованих апаратних засобів індикації й сповіщення користувача.

Провідними виробниками твердотільних накопичувачів є Kingston Technology, Samsung Electronics, Western Digital, Micron Technology, SK Hynix та Seagate Technology. Значний внесок у розвиток архітектури комп’ютерних систем, систем зберігання даних та комп’ютерних мереж зробили такі вчені та фахівці, як Вільям Столлінгс, Девід Паттерсон, Джон Хеннессі, Ендрю Таненбаум та інші дослідники у галузі комп’ютерної інженерії.

Світовими тенденціями розвитку систем діагностики є автоматизація процесів моніторингу технічного стану обладнання, використання вбудованих мікроконтролерних систем, віддалений контроль параметрів через мережу Internet, застосування технологій прогнозування відмов на основі аналізу експлуатаційних даних та використання інтелектуальних алгоритмів обробки інформації. Одним із перспективних напрямів є створення спеціалізованих апаратно-програмних комплексів, які поєднують функції збору, аналізу та відображення діагностичної інформації.

Актуальність даної роботи обумовлена необхідністю підвищення надійності експлуатації твердотільних накопичувачів шляхом створення комп'ютерної системи діагностики, яка дозволяє контролювати основні параметри роботи накопичувача, своєчасно виявляти критичні зміни його технічного стану та інформувати користувача про можливі несправності. Використання додаткових апаратних засобів індикації та звукового сповіщення дозволяє підвищити ефективність моніторингу накопичувача та зменшити ризик втрати важливої інформації.

Метою роботи є розробка комп'ютерної системи діагностики твердотільного накопичувача Kingston SSD A400, яка забезпечує контроль основних параметрів його роботи, аналіз технічного стану та інформування користувача про результати діагностики.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– виконати аналіз конструкції та технічних характеристик накопичувача Kingston SSD A400;

– визначити основні параметри, що підлягають контролю під час діагностики;

– розробити апаратну складову комп'ютерної системи діагностики;

– обґрунтувати вибір мікроконтролера, дисплея та звукового модуля;

– розробити алгоритм роботи системи;

– створити програмне забезпечення для аналізу діагностичних параметрів накопичувача;

– виконати проєктування серверної кімнати для забезпечення функціонування інформаційної інфраструктури;

– реалізувати програму кодування інформації методом Хемінга;

– провести тестування розроблених програмних засобів.

Об’єктом дослідження є процес діагностики технічного стану твердотільних накопичувачів.

Предметом дослідження є методи та засоби побудови апаратно-програмних систем діагностики SSD-накопичувачів.

Практичне значення роботи полягає у розробці концепції комп’ютерної системи діагностики Kingston SSD A400, яка може бути використана як основа для створення реального пристрою контролю стану твердотільних накопичувачів. Отримані результати також можуть бути використані під час вивчення дисциплін, пов’язаних із мікроконтролерними системами, комп’ютерною інженерією, програмуванням та технічною діагностикою комп’ютерного обладнання.

1 ПРИСТРІЙ, ЩО ДІАГНОСТУЄТЬСЯ

1.1 Загальні відомості про твердотільні накопичувачі

Твердотільні накопичувачі (Solid State Drive, SSD) є сучасними пристроями довготривалого зберігання інформації, які використовують мікросхеми флеш-пам’яті замість механічних компонентів, характерних для традиційних жорстких дисків (HDD). Завдяки відсутності рухомих елементів SSD забезпечують високу швидкодію, низьке енергоспоживання, стійкість до механічних впливів та безшумність роботи [1, 3].

З розвитком комп’ютерних технологій твердотільні накопичувачі практично витіснили жорсткі диски з ринку персональних комп’ютерів та ноутбуків. Сучасні SSD використовуються не лише у домашніх комп’ютерах, а й у серверних системах, центрах обробки даних, промислових контролерах та вбудованих системах [3, 4].

Основними компонентами SSD є контролер, мікросхеми флеш-пам’яті NAND Flash, буферна пам’ять та інтерфейс підключення до комп’ютерної системи. Контролер виконує функції управління записом і читанням даних, розподілом навантаження між комірками пам’яті, корекцією помилок та моніторингом стану накопичувача за допомогою технології SMART (Self-Monitoring, Analysis and Reporting Technology) [16, 24].

На сьогоднішній день найбільш поширеними є два типи SSD-накопичувачів: пристрої з інтерфейсом SATA та накопичувачі NVMe. SATA SSD використовують інтерфейс Serial ATA, який був розроблений для жорстких дисків, тому їхня максимальна швидкість передачі даних обмежується пропускною здатністю шини SATA III та становить близько 600 МБ/с [18].

На рисунку 1.1 представлено твердотільний накопичувач з інтерфейсом SATA.

Рисунок 1.1 – Твердотільний накопичувач (SSD) з інтерфейсом SATA

Накопичувачі NVMe використовують інтерфейс PCI Express та спеціалізований протокол Non-Volatile Memory Express, який був розроблений саме для роботи з флеш-пам’яттю. Завдяки цьому забезпечується значно вища швидкість обміну даними та менші затримки доступу до інформації порівняно з SATA SSD [19].

На рисунку 1.2 представлені твердотільні накопичувачі стандарту NVMe у форм-факторі M.2.

Рисунок 1.2 – SSD NVMe

Серед великої кількості моделей твердотільних накопичувачів особливий інтерес для задач діагностики становлять SATA SSD, які широко використовуються у персональних комп’ютерах та ноутбуках [24–27].

1.2 Конструкція та технічні характеристики Kingston SSD A400

Об'єктом дослідження даної роботи є твердотільний накопичувач Kingston SSD A400, який належить до класу накопичувачів з інтерфейсом SATA III та призначений для використання у персональних комп’ютерах і ноутбуках. Дана модель є однією з найбільш поширених серед бюджетних SSD завдяки поєднанню доступної вартості, достатньої продуктивності та високої надійності [16, 17].

На рисунку 1.3 представлено зовнішній вигляд твердотільного накопичувача Kingston SSD A400.

Рисунок 1.3 – Kingston SSD A400

Накопичувач виконаний у форм-факторі 2,5 дюйма та має товщину корпусу 7 мм, що дозволяє використовувати його як у настільних комп’ютерах, так і в сучасних ноутбуках. Для підключення до материнської плати використовується інтерфейс SATA Revision 3.0 зі швидкістю передачі даних до 6 Гбіт/с [17, 18].

Основу накопичувача становлять мікросхеми флеш-пам’яті NAND та спеціалізований контролер, який забезпечує керування процесами запису,

читання, адресації комірок пам’яті та корекції помилок. Контролер також виконує функції моніторингу технічного стану накопичувача та підтримує технологію SMART, що дозволяє отримувати інформацію про основні експлуатаційні параметри пристрою [17, 24].

Серед основних технічних характеристик накопичувача Kingston SSD A400 можна виділити:

– інтерфейс підключення SATA III (6 Гбіт/с);

– форм-фактор 2,5";

– тип пам’яті NAND Flash;

– швидкість послідовного читання даних до 500 МБ/с;

– швидкість послідовного запису даних до 450 МБ/с;

– підтримку технології SMART;

– низьке енергоспоживання та безшумну роботу [16, 17].

Порівняно з традиційними жорсткими дисками Kingston SSD A400 забезпечує значно менший час доступу до даних, швидше завантаження операційної системи та прикладного програмного забезпечення, а також підвищує загальну швидкодію комп’ютерної системи. Відсутність механічних вузлів позитивно впливає на надійність накопичувача та стійкість до вібрацій і ударів [3, 16].

1.3 Параметри та методи діагностики Kingston SSD A400

Основним засобом контролю стану сучасних накопичувачів є технологія SMART. Вона дозволяє отримувати службову інформацію про стан пристрою, накопичувати статистику експлуатації та прогнозувати можливі відмови обладнання [24].

Для накопичувача Kingston SSD A400 найбільш важливими параметрами діагностики є:

– температура накопичувача;

– загальний час роботи пристрою;

– кількість ввімкнень;

– обсяг записаної інформації;

– залишковий ресурс флеш-пам’яті;

– кількість помилок передачі даних;

– поточний стан SMART;

– швидкість читання та запису інформації [16, 24].

Одним із найбільш популярних програмних засобів діагностики SSD є утиліта CrystalDiskInfo. Дана програма забезпечує зчитування SMART-параметрів накопичувача та відображає основні показники його технічного стану у зручному для користувача вигляді.

На рисунку 1.4 представлено вікно програми CrystalDiskInfo під час діагностики накопичувача Kingston SSD A400.

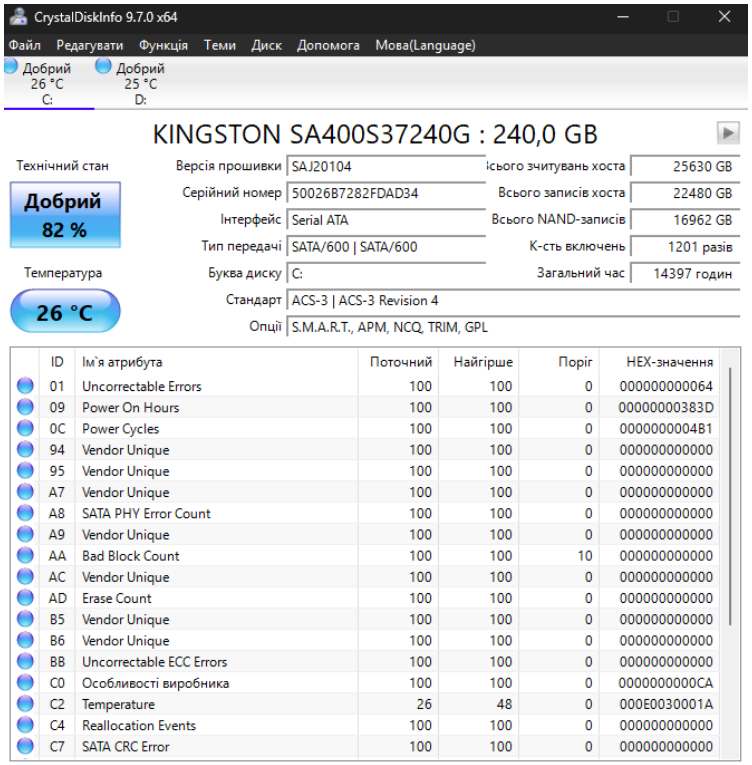


Рисунок 1.4 – Утиліта CrystalDiskInfo для діагностики пристрою

За допомогою утиліти CrystalDiskInfo користувач може отримати інформацію про температуру накопичувача, кількість годин напрацювання, обсяг переданих даних, версію прошивки та загальну оцінку технічного стану пристрою. Програма також дозволяє відстежувати зміни параметрів у процесі

експлуатації та своєчасно реагувати на появу попереджень або критичних значень [24].

Окрім аналізу SMART-показників, важливим методом діагностики є тестування швидкісних характеристик накопичувача. Для цього використовуються спеціалізовані програми, які визначають швидкість послідовного та випадкового читання і запису даних. Зниження продуктивності накопичувача може свідчити про погіршення стану флеш-пам’яті або виникнення проблем у роботі контролера [25, 27].

Ще одним важливим параметром є температура накопичувача. Тривала робота SSD при підвищених температурах негативно впливає на ресурс комірок пам’яті та може призвести до прискореного зношування пристрою. Тому контроль температурного режиму є обов’язковою складовою системи діагностики [25].

На основі проведеного аналізу можна зробити висновок, що для оцінювання технічного стану Kingston SSD A400 доцільно контролювати такі параметри: температуру накопичувача, поточний стан SMART, відсоток залишкового ресурсу пам’яті, кількість помилок передачі даних та швидкість роботи пристрою. Саме ці показники будуть використовуватися у подальшому під час розробки комп’ютерної системи діагностики накопичувача.

Висновки до розділу 1

У даному розділі було розглянуто особливості твердотільних накопичувачів як сучасних засобів зберігання інформації. Проведено аналіз принципів їх роботи, основних переваг над традиційними жорсткими дисками та сучасних інтерфейсів підключення SATA і NVMe.

Досліджено конструкцію та технічні характеристики накопичувача Kingston SSD A400, який обрано як об’єкт діагностики. Встановлено, що даний накопичувач забезпечує достатню продуктивність для використання у персональних комп’ютерах та підтримує засоби контролю власного технічного стану.

Також було проаналізовано основні параметри, які характеризують працездатність накопичувача та можуть бути використані для оцінювання його технічного стану. До найбільш інформативних параметрів належать температура пристрою, швидкість обміну даними, час напрацювання, кількість увімкнень, обсяг записаної інформації та залишковий ресурс накопичувача.

Розглянуто сучасні програмні засоби діагностики SSD, зокрема утиліту CrystalDiskInfo, яка дозволяє здійснювати моніторинг експлуатаційних характеристик накопичувача та своєчасно виявляти ознаки погіршення його стану.

На основі проведеного аналізу визначено перелік параметрів, контроль яких доцільно реалізувати у розроблюваній комп’ютерній системі діагностики Kingston SSD A400. Отримані результати є підґрунтям для вибору апаратних і програмних засобів, що будуть розглянуті в наступних розділах роботи.

2 АПАРАТНА СКЛАДОВА КОМП'ЮТЕРНОЇ СИСТЕМИ ДІАГНОСТИКИ

2.1 Обґрунтування вибору мікроконтролера

Одним із ключових елементів розроблюваної комп’ютерної системи діагностики накопичувача Kingston SSD A400 є мікроконтролер, який забезпечує приймання, обробку та відображення діагностичної інформації. Саме мікроконтролер виконує функції взаємодії з персональним комп’ютером, керування дисплеєм та звуковою сигналізацією.

Для реалізації поставленого завдання мікроконтролер повинен відповідати таким вимогам:

– забезпечувати обмін даними з персональним комп’ютером через USB-підключення;

– підтримувати роботу з графічним дисплеєм;

– мати достатній обсяг оперативної пам’яті для обробки інформації;

– підтримувати сучасні інтерфейси зв’язку;

– мати низьке енергоспоживання;

– бути доступним для програмування та подальшої модернізації системи [6, 8].

Для вибору найбільш придатного рішення було проведено порівняння декількох популярних мікроконтролерних платформ, які широко використовуються під час розробки вбудованих систем.

Arduino Uno є однією з найпоширеніших платформ для навчання та створення простих електронних пристроїв. Її перевагами є простота використання та велика кількість готових програмних бібліотек. Проте невеликий обсяг оперативної пам’яті та низька продуктивність обмежують можливості використання даного мікроконтролера у складніших системах моніторингу та діагностики [8–11].

Мікроконтролер ESP8266 має вбудований модуль бездротового зв’язку Wi-Fi та значно більшу продуктивність порівняно з Arduino Uno. Разом із тим він має

обмежені апаратні ресурси та не підтримує Bluetooth, що звужує можливості його подальшого використання в перспективних проєктах [13].

Таблиця 2.1 – Порівняння характеристик мікроконтролерів

Характеристика	Arduino Uno	ESP8266	ESP32-WROOM-32
Тактова частота	16 МГц	80 МГц	240 МГц
Оперативна пам’ять	2 КБ	160 КБ	520 КБ
Flash-пам’ять	32 КБ	4 МБ	4 МБ
Wi-Fi	Немає	Є	Є
Bluetooth	Немає	Немає	Є
Кількість GPIO	14	17	До 32
Інтерфейси SPI, I2C, UART	Є	Є	Є
Продуктивність	Низька	Середня	Висока

Найбільш функціональним серед розглянутих варіантів є ESP32-WROOM-32. Даний мікроконтролер побудований на базі двоядерного процесора Tensilica Xtensa LX6 з тактовою частотою до 240 МГц та має 520 КБ оперативної пам’яті. Крім того, він оснащений вбудованими модулями Wi-Fi та Bluetooth, підтримує велику кількість периферійних інтерфейсів і характеризується високою продуктивністю при невеликому енергоспоживанні [22, 23].

Для розроблюваної комп’ютерної системи діагностики використання ESP32-WROOM-32 (рисунок 2.1) дозволяє забезпечити відображення інформації на дисплеї, реалізацію звукового сповіщення та можливість подальшого розширення функціоналу системи без заміни апаратної платформи. Значний обсяг пам’яті та висока швидкодія також створюють запас продуктивності для реалізації додаткових функцій аналізу та обробки діагностичних даних.

Рисунок 2.1 – Зовнішній вигляд мікроконтролера ESP32-WROOM-32

2.2 Обґрунтування вибору дисплея

Однією з функцій розроблюваної комп’ютерної системи діагностики є відображення інформації про технічний стан накопичувача Kingston SSD A400. Для забезпечення зручної взаємодії користувача із системою необхідно використати дисплей, на якому будуть відображатися основні параметри діагностики, зокрема температура накопичувача, загальний стан пристрою, швидкість передачі даних та повідомлення про можливі несправності.

Під час вибору дисплея для проєктованої системи необхідно враховувати такі критерії:

– сумісність із мікроконтролером ESP32-WROOM-32;

– простота підключення;

– низьке енергоспоживання;

– достатня інформативність відображення;

– компактні габаритні розміри;

– доступність програмних бібліотек для розробки [10, 11, 21].

Для порівняння було розглянуто два найбільш поширені типи дисплеїв, які використовуються у мікроконтролерних системах: символьний LCD-дисплей 1602 та графічний OLED-дисплей SSD1306.

Таблиця 2.2 – Порівняння характеристик дисплеїв

Характеристика	LCD 1602	OLED SSD1306
Тип дисплея	Символьний	Графічний
Роздільна здатність	16×2 символи	128×64 пікселі
Підсвічування	Потрібне	Самосвітні пікселі
Споживання енергії	Середнє	Низьке
Інтерфейс	I2C, паралельний	I2C, SPI
Якість відображення	Середня	Висока
Відображення графіки	Ні	Так
Габарити	Більші	Компактні

Символьний дисплей LCD 1602 є одним із найпоширеніших засобів відображення інформації у вбудованих системах. Його перевагами є низька вартість та простота використання. Однак він має обмежені можливості щодо відображення інформації, оскільки дозволяє виводити лише текстові дані у двох рядках по 16 символів [10].

Графічний OLED-дисплей SSD1306 забезпечує значно кращу якість відображення завдяки роздільній здатності 128×64 пікселі. На такому дисплеї можливо одночасно відображати декілька параметрів діагностики, текстові повідомлення, графічні символи та індикатори стану системи. Додатковою перевагою є висока контрастність зображення та відсутність потреби у додатковому підсвічуванні [11, 21].

На рисунку 2.2 представлено зовнішній вигляд OLED-дисплея SSD1306.

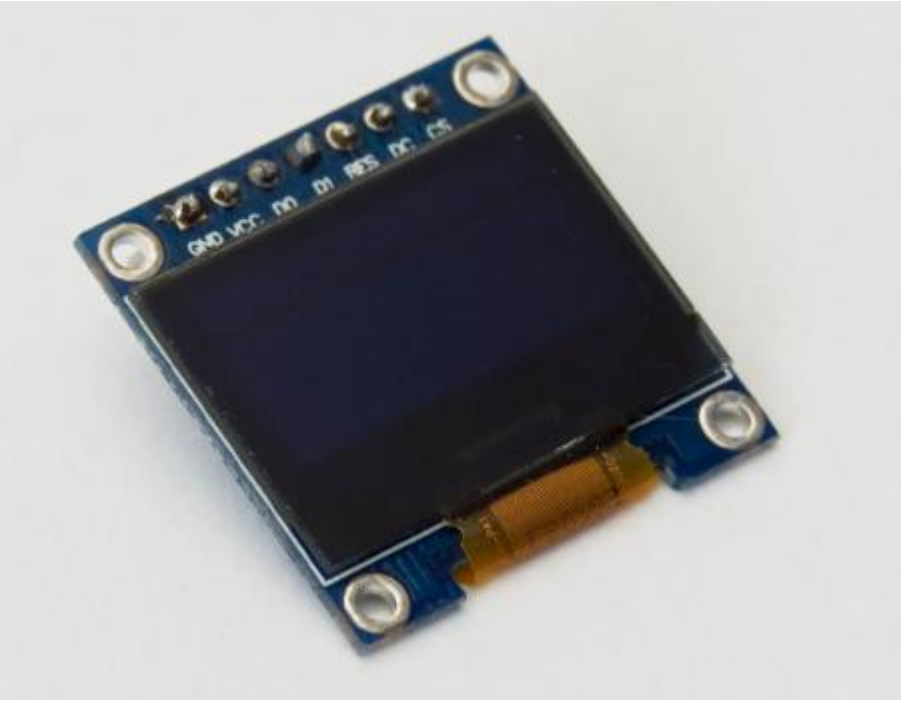


Рисунок 2.2 – OLED-дисплей SSD1306

Підключення дисплея до мікроконтролера ESP32 здійснюється за допомогою інтерфейсу I2C, що потребує використання лише двох сигнальних ліній передачі даних. Це дозволяє зменшити кількість задіяних контактів мікроконтролера та спростити схему пристрою [21].

У розроблюваній системі діагностики дисплей буде використовуватися для відображення таких даних:

– назви накопичувача;

– температури SSD;

– поточного технічного стану накопичувача;

– попереджувальних повідомлень;

– результатів виконання діагностики.

2.3 Обґрунтування вибору звукового модуля

Для підвищення зручності використання комп’ютерної системи діагностики доцільно передбачити не лише візуальне відображення інформації на дисплеї, а й звукове сповіщення користувача про виникнення небезпечних або аварійних ситуацій. Такий підхід дозволяє оперативно привертати увагу

користувача до проблем із накопичувачем навіть у випадках, коли він не контролює показники на дисплеї.

У розроблюваній системі звукове сповіщення планується використовувати у таких випадках:

– перевищення допустимої температури накопичувача;

– виявлення критичного стану SSD;

– завершення процесу діагностики;

– виникнення помилок під час обміну даними між персональним комп’ютером та мікроконтролером.

Для реалізації звукової індикації можуть використовуватися різні пристрої, серед яких найбільш поширеними є активні та пасивні п’єзоелектричні динаміки (buzzer). Активний динамік має вбудований генератор звукової частоти та потребує лише подачі напруги живлення для формування звукового сигналу. Пасивний динамік не містить генератора і потребує формування сигналу безпосередньо мікроконтролером [8, 11].

Таблиця 2.3 – Порівняння звукових модулів

Характеристика	Активний buzzer	Пасивний buzzer
Простота підключення	Висока	Середня
Необхідність генерації сигналу	Ні	Так
Можливість зміни тональності	Обмежена	Висока
Навантаження на мікроконтролер	Мінімальне	Підвищене
Складність програмування	Низька	Вища

Оскільки розроблювана система не потребує відтворення складних звукових сигналів або мелодій, використання пасивного динаміка є недоцільним. Для задач діагностики достатньо короткого звукового сигналу, який

інформуватиме користувача про наявність несправності або перевищення допустимих параметрів роботи накопичувача.

На рисунку 2.3 представлено активний п’єзоелектричний динамік, який пропонується використовувати у складі комп’ютерної системи діагностики.

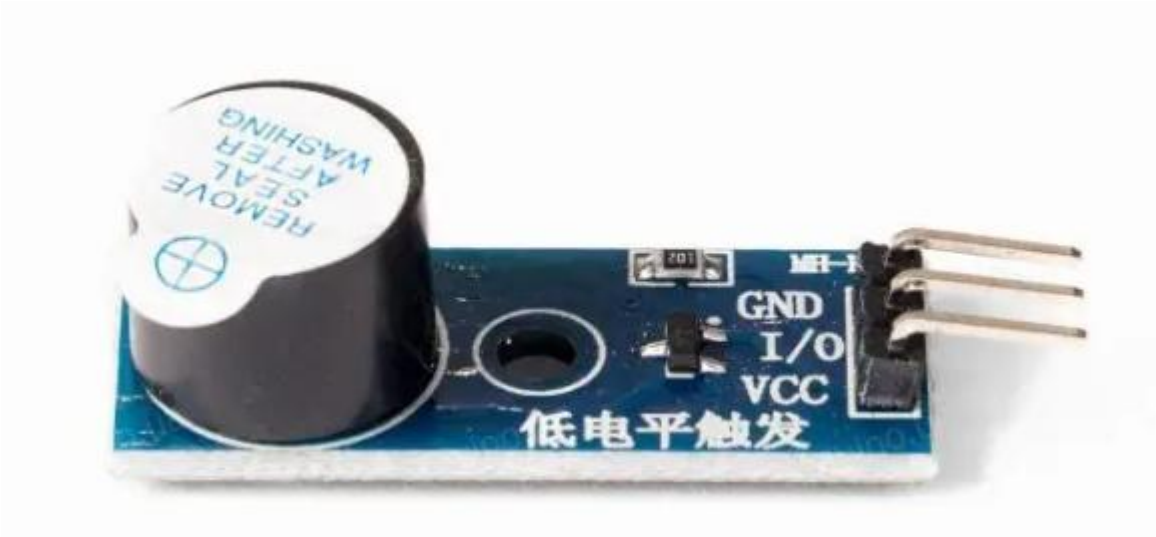


Рисунок 2.3 – Активний п’єзоелектричний динамік (buzzer)

Перевагами використання активного звукового модуля є простота підключення до мікроконтролера ESP32, низьке енергоспоживання, невисока вартість та мінімальні вимоги до програмного забезпечення. Для формування звукового сигналу достатньо подати логічний рівень на відповідний вихід мікроконтролера, що значно спрощує реалізацію системи [13, 22].

У розроблюваній комп’ютерній системі діагностики звуковий модуль буде використовуватися як додатковий засіб інформування користувача. При виявленні критичних параметрів накопичувача, наприклад перевищення температури або погіршення технічного стану SSD, мікроконтролер формуватиме звуковий сигнал для привернення уваги користувача до проблеми.

2.4 Схема під'єднання пристроїв комп'ютерної системи діагностики

Після вибору основних компонентів системи було розроблено схему їх взаємного підключення. До складу системи входять персональний комп'ютер з

установленим накопичувачем Kingston SSD A400, мікроконтролер ESP32-WROOM-32, OLED-дисплей SSD1306 та звуковий модуль типу buzzer.

На рисунку 2.4 наведено схему під'єднання пристроїв комп'ютерної системи діагностики.

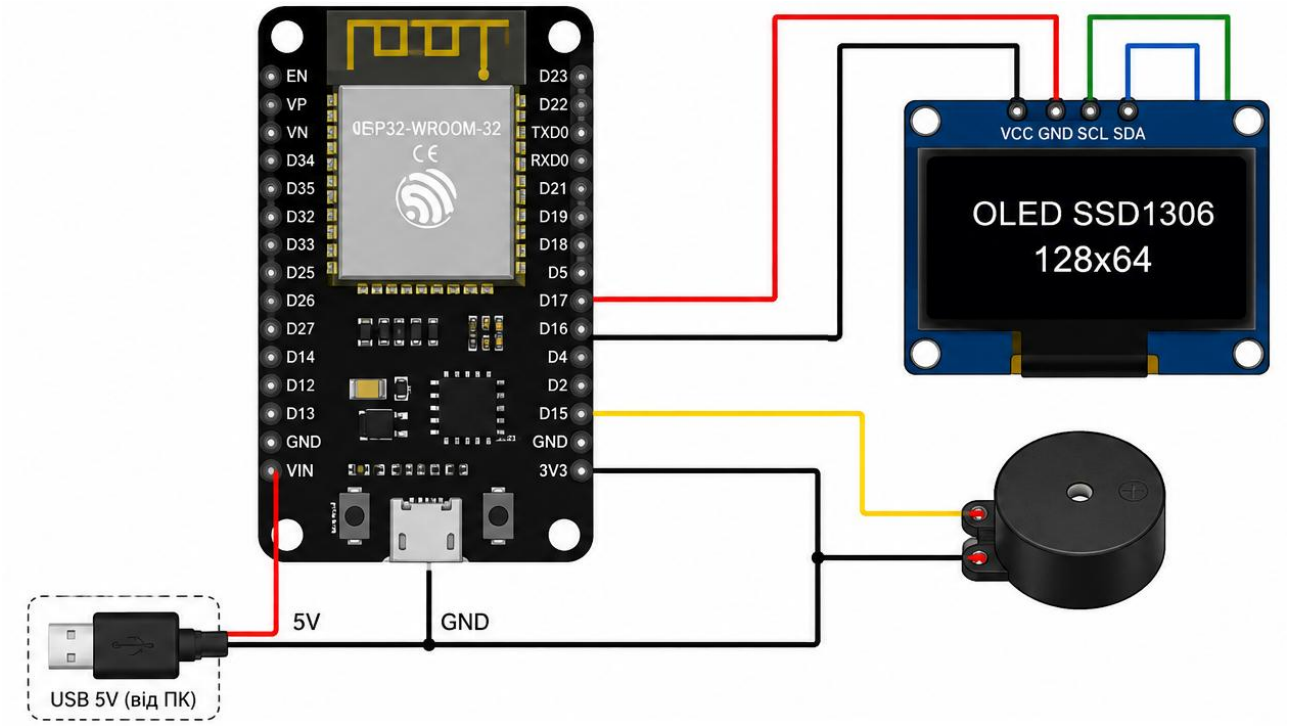


Рисунок 2.4 – Схема під'єднання пристроїв комп'ютерної системи діагностики

Персональний комп'ютер здійснює зчитування діагностичних параметрів накопичувача Kingston SSD A400 за допомогою спеціалізованого програмного забезпечення. Після аналізу отримана інформація передається через USB-інтерфейс до мікроконтролера ESP32-WROOM-32.

OLED-дисплей SSD1306 підключається до мікроконтролера через інтерфейс I2C та використовується для відображення поточних параметрів діагностики накопичувача. Для передачі даних використовуються лінії SDA та SCL, а також контакти живлення VCC і GND.

Звуковий модуль підключається до одного з цифрових виходів ESP32 і використовується для формування попереджувальних сигналів у разі виявлення критичних параметрів роботи накопичувача.

Висновки до розділу 2

У даному розділі було розроблено апаратну складову комп’ютерної системи діагностики твердотільного накопичувача Kingston SSD A400.

Проведено аналіз вимог до апаратної платформи системи та виконано порівняння сучасних мікроконтролерів. За результатами аналізу для реалізації проєкту обрано мікроконтролер ESP32-WROOM-32, який характеризується високою продуктивністю, значним обсягом пам’яті, підтримкою сучасних інтерфейсів зв’язку та можливістю подальшого розширення функціональних можливостей системи.

Для відображення результатів діагностики обрано графічний OLED-дисплей SSD1306. Даний дисплей забезпечує високу інформативність, компактні розміри та простоту підключення до мікроконтролера за допомогою інтерфейсу I2C.

З метою підвищення зручності використання системи передбачено застосування активного п’єзоелектричного динаміка, який забезпечує звукове інформування користувача про виникнення критичних ситуацій під час роботи накопичувача.

На основі обраних компонентів розроблено схему під’єднання пристроїв комп’ютерної системи діагностики, яка визначає взаємозв’язки між мікроконтролером, засобами індикації та персональним комп’ютером.

Отримані результати створюють необхідну апаратну основу для реалізації програмної складової комп’ютерної системи діагностики, яка буде розглянута у наступному розділі.

3 ПРОГРАМНА СКЛАДОВА КОМП'ЮТЕРНОЇ СИСТЕМИ ДІАГНОСТИКИ

3.1 Обґрунтування вибору мови програмування та середовища розробки

Програмна складова комп’ютерної системи діагностики Kingston SSD A400 призначена для забезпечення роботи мікроконтролера ESP32-WROOM-32, приймання діагностичних даних від персонального комп’ютера, їх обробки, виведення результатів на OLED-дисплей SSD1306 та формування звукового попередження у разі виявлення критичного стану накопичувача.

На відміну від звичайних програмних утиліт діагностики SSD, розроблене програмне забезпечення орієнтоване саме на мікроконтролерну реалізацію. Персональний комп’ютер у цій системі виконує роль джерела діагностичних даних, а ESP32 — роль автономного модуля індикації та оповіщення. Такий підхід відповідає структурі апаратної частини, розробленої у другому розділі, де для системи було обрано ESP32-WROOM-32, OLED-дисплей SSD1306 та звуковий модуль.

Для програмування мікроконтролера доцільно використати мову MicroPython. Вона є спрощеною реалізацією Python для мікроконтролерів і дозволяє створювати програмне забезпечення для ESP32 без складної низькорівневої конфігурації. Основними перевагами MicroPython є простота синтаксису, підтримка роботи з GPIO, I2C, UART, таймерами, файлами та периферійними пристроями.

Як середовище розробки обрано Thonny IDE. Це середовище підтримує роботу з MicroPython, дозволяє завантажувати прошивку на ESP32, передавати файли до пам’яті мікроконтролера, запускати код та виконувати налагодження через USB-з’єднання. У прикріпленому прикладі також використовується підхід, за якого мікроконтролерна частина працює на MicroPython, а хост-система передає їй діагностичні дані для локального відображення .

Для реалізації програмної складової використовуються такі програмні модулі:

– machine — для роботи з контактами GPIO, I2C та UART;

– ssd1306 — для керування OLED-дисплеєм;

– time — для організації затримок і циклічного оновлення даних;

– json — для розбору діагностичних повідомлень, отриманих від ПК;

– sys — для приймання даних через послідовний інтерфейс USB-UART.

Таким чином, вибір MicroPython і Thonny IDE є обґрунтованим, оскільки ці засоби дозволяють швидко реалізувати програму для ESP32, організувати приймання параметрів SSD, виведення інформації на дисплей та керування звуковим сигналом.

3.2 Алгорим роботи системи

Алгоритм роботи програмного забезпечення ESP32 побудований за циклічним принципом. Після подачі живлення мікроконтролер виконує ініціалізацію периферійних пристроїв, перевіряє готовність OLED-дисплея, налаштовує вихід для звукового модуля та переходить у режим очікування діагностичних даних.

Дані про стан Kingston SSD A400 формуються на стороні персонального комп’ютера. До них належать температура накопичувача, загальна оцінка стану, залишковий ресурс, кількість помилок, обсяг записаних даних та статус SMART. Після цього дані передаються до ESP32 через USB-послідовний інтерфейс у вигляді текстового повідомлення формату JSON.

Загальна послідовність роботи програми така:

1. Увімкнення живлення ESP32.
2. Ініціалізація OLED-дисплея SSD1306 через інтерфейс I2C.
3. Ініціалізація звукового модуля через цифровий вихід GPIO.
4. Виведення стартового повідомлення на дисплей.
5. Очікування надходження діагностичних даних від ПК.
6. Приймання рядка з параметрами SSD.
7. Перевірка правильності формату отриманих даних.

8. Розбір отриманого повідомлення.

9. Порівняння параметрів із допустимими межами.

10.Визначення стану накопичувача: нормальний, попереджувальний або критичний.

11.Виведення результатів на OLED-дисплей.

12.Увімкнення звукового сигналу у разі критичного стану.

13.Повернення до очікування нового повідомлення.

Нормальний стан визначається тоді, коли температура накопичувача не перевищує допустимого значення, SMART-статус не містить помилок, а залишковий ресурс знаходиться у безпечному діапазоні. Попереджувальний стан встановлюється, якщо температура підвищена або ресурс SSD зменшився до граничного рівня. Критичний стан фіксується у випадку перегріву, низького залишкового ресурсу або наявності помилок SMART.

3.3 Опис реалізації

Програмне забезпечення мікроконтролера складається з кількох логічних частин:

– блоку ініціалізації;

– блоку приймання даних;

– блоку аналізу параметрів;

– блоку виведення інформації на дисплей;

– блоку керування звуковим оповіщенням.

На початку програми підключаються необхідні бібліотеки та створюються об’єкти для роботи з дисплеєм і звуковим модулем.

```
from machine import Pin, I2C
import ssd1306
import time
import sys
import json
```

```
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
display = ssd1306.SSD1306_I2C(128, 64, i2c)


buzzer = Pin(23, Pin.OUT)
buzzer.value(0)
```

У цьому фрагменті коду задаються контакти ESP32 для роботи з OLED-дисплеєм. Лінія SDA підключена до GPIO21, а SCL — до GPIO22. Звуковий модуль підключений до GPIO23.

Для зручного оновлення дисплея створено окрему функцію:

```
def show_message(line1, line2="", line3="", line4=""):
    display.fill(0)
    display.text(line1, 0, 0)
    display.text(line2, 0, 16)
    display.text(line3, 0, 32)
    display.text(line4, 0, 48)
    display.show()
```

Ця функція очищує екран і виводить до чотирьох рядків тексту. Вона використовується для стартового повідомлення, відображення поточних параметрів SSD та повідомлень про помилки.

Після запуску програми на дисплей виводиться повідомлення про готовність системи:

```
show_message("SSD DIAG", "ESP32 READY", "Waiting data...")
```

Основний цикл програми постійно очікує надходження рядка з даними:

```
while True:
    try:
        line = sys.stdin.readline()
        if line:
            data = json.loads(line)
            analyze_data(data)
```

```
except Exception:
    show_message("DATA ERROR", "Wrong format")
    buzzer.value(1)
    time.sleep(0.2)
    buzzer.value(0)
```

Отримане повідомлення перетворюється з формату JSON у набір параметрів. Якщо формат неправильний, на дисплей виводиться повідомлення про помилку, а звуковий модуль подає короткий сигнал.

Для аналізу параметрів використовується функція analyze_data():

```
def analyze_data(data):
    temp = data.get("temp", 0)
    health = data.get("health", 0)
    errors = data.get("errors", 0)
    smart = data.get("smart", "OK")

    status = "NORMAL"

    if temp >= 60 or health <= 20 or errors > 0 or smart != "OK":
        status = "CRITICAL"
        alarm()
    elif temp >= 50 or health <= 40:
        status = "WARNING"
    else:
        buzzer.value(0)

    show_message(
        "Kingston A400",
        "Temp: " + str(temp) + " C",
        "Health: " + str(health) + " %",
        status
```

)

У цій функції виконується перевірка температури, залишкового ресурсу, кількості помилок і SMART-статусу. Якщо хоча б один параметр переходить у критичну область, система формує стан CRITICAL і вмикає звуковий сигнал.

Функція звукового оповіщення має такий вигляд:

```
def alarm():
    for i in range(3):
        buzzer.value(1)
        time.sleep(0.2)
        buzzer.value(0)
        time.sleep(0.2)
```

Звуковий сигнал складається з трьох коротких імпульсів. Такий режим дозволяє користувачу швидко звернути увагу на проблему з накопичувачем.

Приклад повідомлення, яке надходить від персонального комп'ютера до ESP32:

```
{"temp": 42, "health": 86, "errors": 0, "smart": "OK"}
```

Якщо накопичувач працює нормально, на OLED-дисплеї відображається приблизно така інформація:

Kingston A400

Temp: 42 C

Health: 86 %

NORMAL

У разі критичного стану, наприклад при температурі 64 °C або наявності помилок SMART, дисплей відображає статус CRITICAL, а звуковий модуль подає попереджувальний сигнал.

3.5 Тестування роботи

Оскільки ESP32 не має прямого доступу до SMART-параметрів SSD через SATA-інтерфейс, зчитування діагностичної інформації виконується на стороні

персонального комп’ютера. Саме ПК отримує параметри Kingston SSD A400 за допомогою спеціалізованого програмного забезпечення, після чого передає їх мікроконтролеру через USB.

Такий принцип відповідає логіці апаратної схеми: персональний комп’ютер є джерелом даних, ESP32 виконує їх відображення й аналіз, OLED-дисплей показує результат, а звуковий модуль повідомляє про небезпечний стан. У другому розділі вже зазначено, що ПК зчитує параметри SSD, а потім передає інформацію до ESP32 через USB-інтерфейс .

Для передавання даних можна використати допоміжний скрипт на Python:

```
import serial
import json
import time

port = serial.Serial("COM5", 115200, timeout=1)

while True:
    data = {
        "temp": 42,
        "health": 86,
        "errors": 0,
        "smart": "OK"
    }

    message = json.dumps(data) + "\n"
    port.write(message.encode("utf-8"))

    time.sleep(2)
```

У реальній системі замість тестових значень у змінну data підставляються параметри, отримані від SMART-утиліти або іншої програми діагностики SSD.

Передавання виконується з певним інтервалом, наприклад один раз на 2 секунди. Завдяки цьому мікроконтролер регулярно оновлює інформацію на дисплеї.

Перевірка програмного забезпечення мікроконтролера виконувалася шляхом передавання до ESP32 діагностичних повідомлень із різними значеннями параметрів. Метою перевірки було підтвердження правильності приймання даних, їх аналізу, виведення інформації на OLED-дисплей та роботи звукового модуля.

Було розглянуто три основні режими:

– нормальний стан SSD;

– попереджувальний стан SSD;

– критичний стан SSD.

Для нормального стану використовувалися такі параметри:

{"temp": 42, "health": 86, "errors": 0, "smart": "OK"}

У цьому випадку ESP32 виводить на дисплей температуру, залишковий ресурс і статус NORMAL. Звуковий сигнал не активується.

Для попереджувального стану використовувалися такі параметри:

{"temp": 52, "health": 35, "errors": 0, "smart": "OK"}

У цьому випадку система визначає підвищену температуру або зниження ресурсу накопичувача та виводить статус WARNING. Звуковий сигнал може не вмикатися, оскільки стан ще не є критичним, але користувач отримує візуальне попередження.

Для критичного стану використовувалися такі параметри:

{"temp": 64, "health": 15, "errors": 2, "smart": "BAD"}

У цьому випадку ESP32 визначає критичний стан, виводить на дисплей статус CRITICAL та активує звукове оповіщення. Це дозволяє користувачу своєчасно звернути увагу на можливу несправність SSD.

Результати перевірки наведено в таблиці 3.1.

Таблиця 3.1 – Результати перевірки роботи ПЗ мікроконтролера

№	Температура, °C	Ресурс, %	Помилки	SMART	Результат на дисплеї	Звуковий сигнал
1	42	86	0	OK	NORMAL	Ні
2	52	35	0	OK	WARNING	Ні
3	64	15	2	BAD	CRITICAL	Так

Отримані результати підтверджують, що програмне забезпечення мікроконтролера правильно обробляє діагностичні параметри, визначає стан накопичувача та забезпечує індикацію результатів. На відміну від попереднього варіанта, де основна увага приділялася програмі з графічним інтерфейсом на ПК, у цьому розділі розглянуто саме програмну реалізацію для ESP32.

Висновки до розділу 3

У третьому розділі було розроблено програмну складову комп’ютерної системи діагностики Kingston SSD A400, орієнтовану на роботу мікроконтролера ESP32-WROOM-32.

Обґрунтовано вибір мови програмування MicroPython та середовища Thonny IDE, які забезпечують зручну розробку, завантаження та налагодження програмного забезпечення для ESP32. Розроблено алгоритм роботи мікроконтролера, що передбачає ініціалізацію OLED-дисплея, приймання діагностичних даних від персонального комп’ютера, аналіз параметрів SSD, виведення результатів на дисплей та керування звуковим модулем.

Описано реалізацію основних програмних блоків: приймання даних у форматі JSON, перевірку температури, залишкового ресурсу, кількості помилок і SMART-статусу. Передбачено три режими роботи системи: нормальний, попереджувальний і критичний.

Проведена перевірка показала, що програмне забезпечення ESP32 коректно реагує на зміну діагностичних параметрів накопичувача, відображає стан

Kingston SSD A400 на OLED-дисплеї та активує звукове оповіщення у випадку критичних значень. Таким чином, розроблена програмна складова забезпечує практичну реалізацію функцій мікроконтролерного модуля системи діагностики.

4 ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ МЕРЕЖІ

4.1 Вихідні данні для проектування корпоративної КМ

Для проєктування корпоративної КМ необхідно об'єднати 8 ЛКМ. Конфігурація такої КМ має відповідати «дереву». «Дерево» вибирається з графа представленому на рисунку 4.1. Номери гілки графа відповідають номеру ЛКМ. Гілки графа з'єднані за допомогою маршрутизаторів A, B, C.

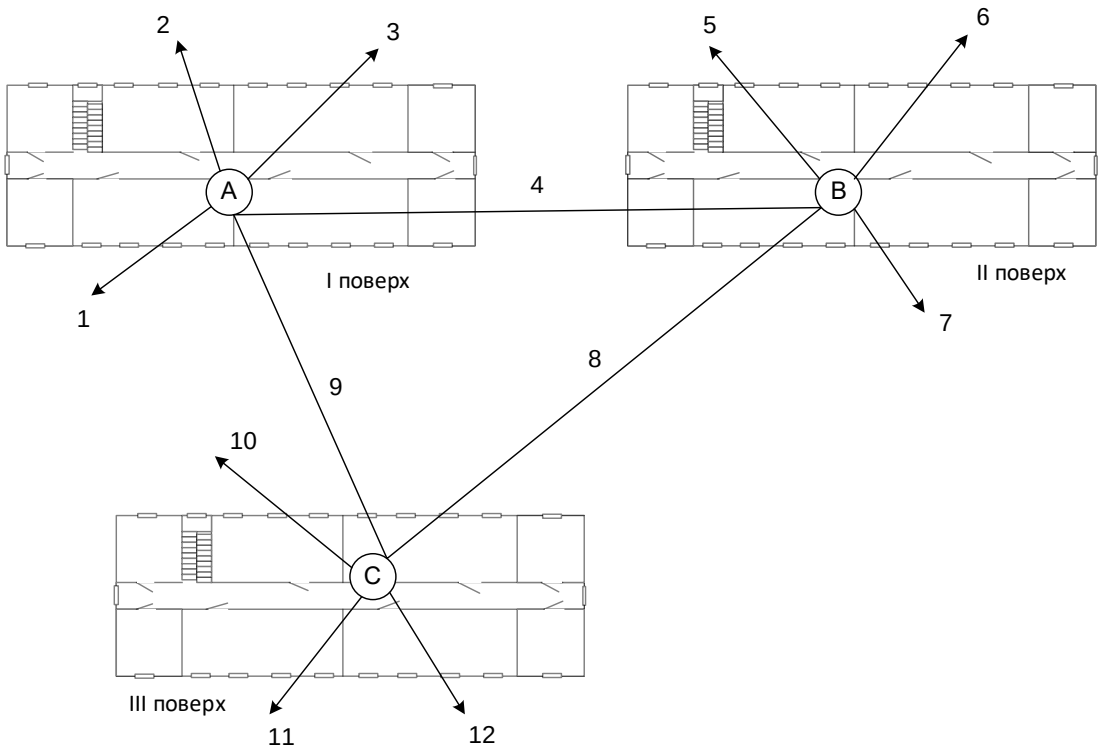


Рисунок 4.1 – Загальний граф корпоративної КМ

У відповідності з вхідними даними кваліфікаційної роботи бакалавра, а саме варіанту №3, поданими у таблиці 4.1 було складено граф проєктованої комп'ютерної мережі та представлено на рисунку 4.2.

Таблиця 4.1 – Вхідні дані графа корпоративної КМ

Номер варіанту	Номера гілок «дерева» які входять до корпоративної КМ								Інтеграція в Internet
									Маршрутизатор
3	1	2V	4	5S	6	7	10	11P	C

Для проєктування КМ вибираються наступні номери гілок, а саме: 1, 2, 4, 5, 6, 7, 10 та 11. Всі інші гілки графа прибираються.

Один з маршрутизаторів КМ, а саме маршрутизатор С має безпосередньо інтегруватися в глобальну КМ – Internet. Для реалізації цього зв'язку мають використовуватися орендовані канали або лінії зв'язку, а саме через оптоволоконний зв'язок.

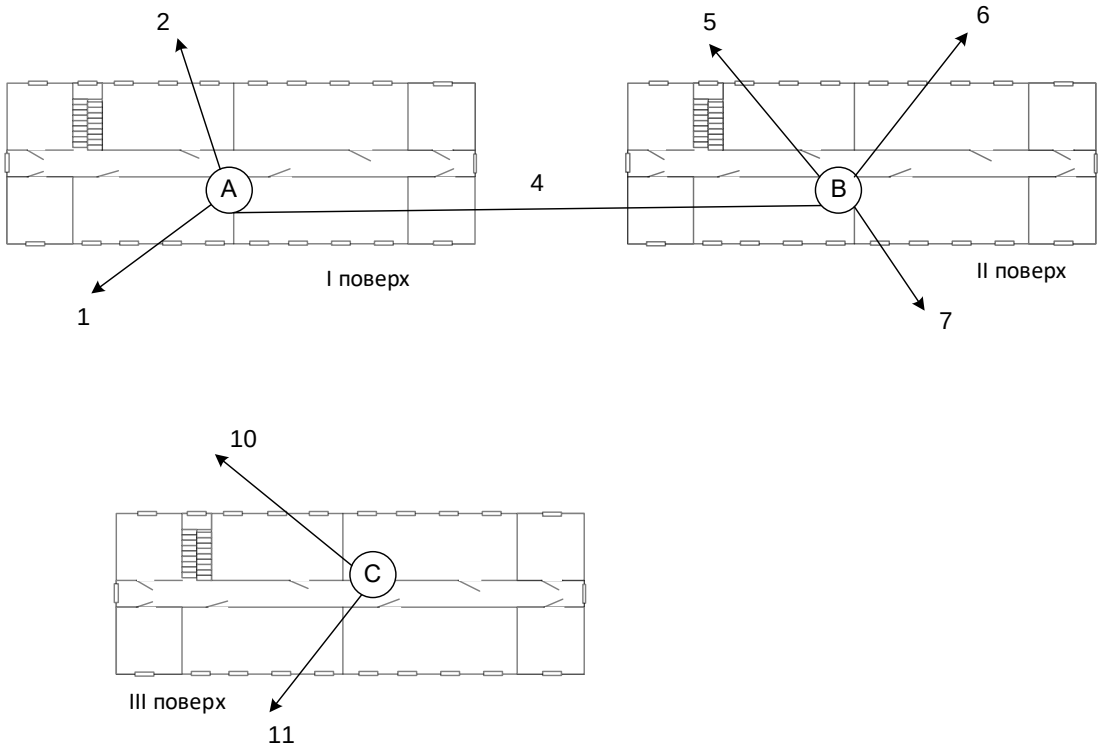


Рисунок 4.2 – Граф корпоративної КМ

Як можна побачити з рисунку графа, утворюється дві корпоративні комп'ютерні мережі, які не з'єднанні між собою і можуть працювати окремо одна від одної, це може вплинути на подальше проектування КМ та її маршрутизацію. Також, маршрутизатори А та В не будуть з'єднані з ГКМ Internet.

4.2 Технічні параметри КМ

Корпоративна комп’ютерна система – це масштабна IT-інфраструктура окремого підприємства, яка об'єднує комп'ютери, сервери, бази даних та програмне забезпечення для спільної роботи всіх підрозділів і філій як єдиного цілого.

Зазвичай, корпоративні комп’ютерні системи складаються з декількох ЛКМ.

Локальні комп’ютерні мережі (ЛКМ) повинні проєктуватися відповідно до вимог чинних міжнародних стандартів, що регламентують їхню архітектуру та принципи функціонування. Дотримання цих стандартів забезпечує сумісність обладнання різних виробників, стабільність роботи мережі, можливість масштабування та спрощення технічного обслуговування.

Стандарти визначають основні параметри побудови ЛКМ, зокрема топологію мережі, методи доступу до середовища передачі даних, фізичні характеристики кабельних систем, формати кадрів, структуру та довжину сегментів, а також правила об’єднання окремих сегментів у єдину мережеву інфраструктуру.

У таблиці 2.2 номера ЛКМ з визначеними стандартами за якими вони мають проєктуватися. Стандарти подані у виді скорочень, які мають наступні значення:

BT – Стандарт IEEE 802.3, специфікація 10Base-T;

BF – Стандарт IEEE 802.3, специфікація 10Base-FL;

B2 – Стандарт IEEE 802.3, специфікація 10Base-2;

B5 – Стандарт IEEE 802.3, специфікація 10Base-5;

TR – Стандарт IEEE 802.5, специфікація Token Ring.

Крім стандартів, в таблиці 4.2 для кожної гілки корпоративної КМ вказано наступні параметри: проєктна кількість робочих станцій (PC) N та проєктна протяжність L. Протяжність L визначається як частка від довжини l, де l – це максимальна допустима довжина сегмента ЛКМ для мереж типу 10Base-2 і 10Base-5 або середня довжина відвідного кабелю в мережах 10Base-T і Token Ring

на ділянці між концентратором і робочою станцією, а також у мережі 10Base-FL між концентратором і трансивером.

Таблиця 4.2 – Неповна таблиця параметрів ЛКМ.

Номер варіанту	Параметри ЛКМ	Номери гілки графа (ЛКМ)							
		1	2	4	5	6	7	10	11
3	Стандарт	BT+B2	BT	BT	B5	B2	-	TR	-
	Кількість PC, N	25+(n/2	30-n	10+n	90+n	30+n	40-n	15+n	20+n
	Протяжність, L	0.6l	0.85l	0.9l	Nl/60	Nl/30	0.8l	0.9l	Nl/50

Як можна побачити з таблиці 4.2, то у деяких гілках не задано стандарти, а саме у ЛКМ №1, ЛКМ №7 та ЛКМ №11, також, не розраховано кількість PC та протяжність мережі у всіх ЛКМ корпоративної мережі.

Для того, щоб визначити стандарт для цих гілок, пропонується порівняти стандарти між собою, а також, провести порівняння Ethernet та Token Ring.

Ethernet – це найпоширеніша у світі технологія побудови локальних обчислювальних мереж. Вона застосовується як у невеликих офісах і домашніх мережах, так і в масштабних корпоративних інфраструктурах та дата-центрах. Популярність Ethernet зумовлена простотою реалізації, високою надійністю, гнучкістю масштабування та економічною доступністю обладнання.

У широкому розумінні Ethernet – це не одна технологія, а ціле сімейство мережевих рішень, які підтримують різні швидкості передавання даних і типи фізичних середовищ. Базовим нормативним документом є стандарт IEEE 802.3, що регламентує функціонування канального і фізичного рівнів моделі OSI.[7]

Залежно від середовища передавання та швидкості обміну даними, стандарт IEEE 802.3 передбачає низку специфікацій, зокрема:

10Base-5 (Thick Ethernet) – використовує товстий коаксіальний кабель та може забезпечувати передачу даних на відстань до 500 метрів без підсилювача сигналу.

10Base-2 (Thin Ethernet) – удосконалена версія з тоншим коаксіальним кабелем, довжина сегмента якого обмежена 185 метрами.

10Base-T – найпоширеніший варіант, який застосовує виту пару як середовище передавання даних та з’єднує пристрої через комутатор або концентратор.

10Base-FL використовує оптичне волокно, що дозволяє передавати дані на значні відстані та забезпечує стійкість до електромагнітних завад.

Ethernet розвивався протягом кількох десятиліть, поступово збільшуючи швидкість передачі даних – від 10 Мбіт/с до 100 Гбіт/с і більше. Простота побудови мереж, низька вартість обладнання та зворотна сумісність із попередніми стандартами зробили Ethernet основною технологією для локальних мереж у всьому світі.

Архітектура Token Ring була розроблена компанією IBM та стандартизована як IEEE 802.5. Її створення мало на меті забезпечити стабільну й передбачувану роботу локальної мережі з детермінованим доступом до середовища передавання даних.

У цій технології всі вузли об’єднуються в логічне кільце, яким безперервно циркулює спеціальний службовий кадр – маркер (token). Передавати дані може лише той пристрій, який отримав маркер. Після завершення передачі вузол передає токен наступному учаснику кільця. Такий механізм (Token Passing) повністю усуває колізії та забезпечує гарантований час доступу до мережі для кожного вузла.

Token Ring вважається складнішою технологією порівняно з Ethernet, оскільки потребує спеціального обладнання – MAU (Multistation Access Unit), що фізично може мати вигляд «зірки», але логічно формує кільце. Перевагами є стабільна пропускна здатність, контрольований трафік і передбачувані затримки. Проте висока вартість обладнання та складність обслуговування призвели до поступового витіснення цієї технології Ethernet-рішеннями.[8]

На рисунку 4.3 наведено принцип роботи Token Ring

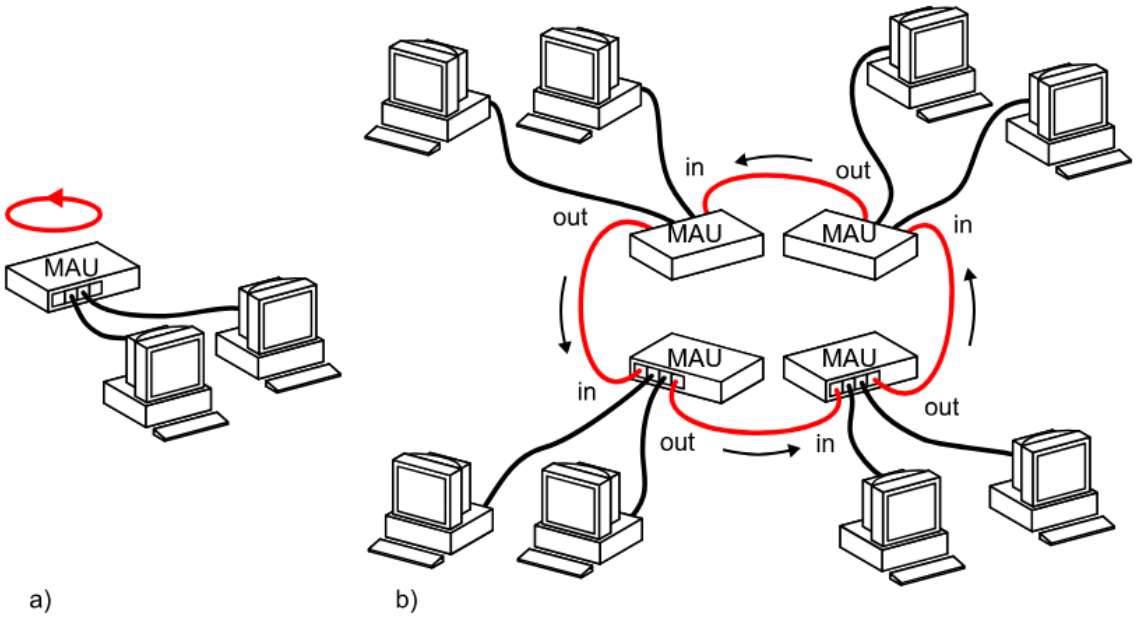


Рисунок 4.3 – Схема роботи технології Token Ring: a) з використанням однієї MAU; b) з використанням декількох MAU, підключених одна до одної

4.2.1 Порівняння топології Ethernet та Token Ring

У сучасних реалізаціях Ethernet застосовується топологія «зірка», де всі пристрої підключаються до центрального комутатора. Це спрощує монтаж, діагностику та масштабування мережі. У разі відмови одного вузла інші залишаються працездатними. Топологію «зірка» наведено на рисунку 4.4.

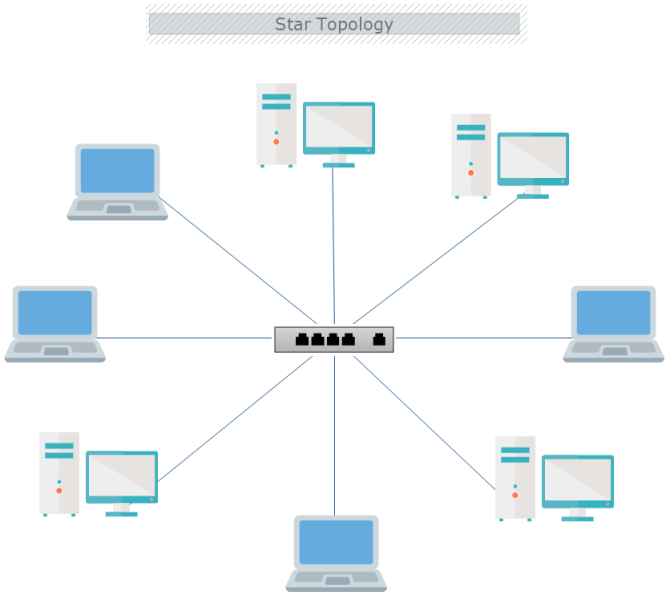


Рисунок 4.4 – Топологія «зірка»

Ранні версії Ethernet використовували топологію «шина», у якій усі пристрої підключалися до спільного коаксіального кабелю. Пошкодження кабелю або неправильне термінування могло призвести до зупинки всієї мережі. Топологію «шина» наведено на рисунку 4.5

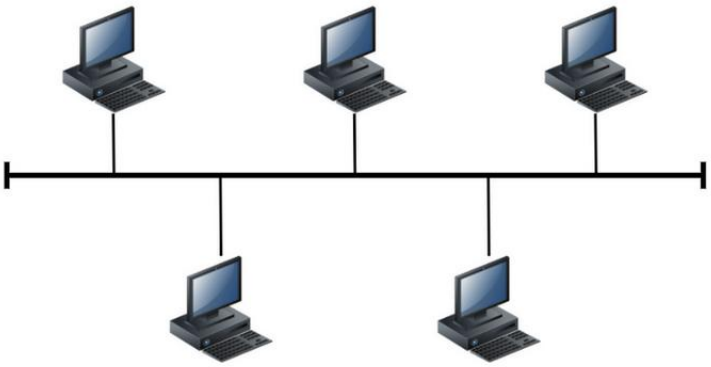


Рисунок 4.5 – Топологія «шина»

Token Ring логічно використовує топологію «кільце», де кожен вузол пов’язаний із двома сусідніми. Дані проходять послідовно через усі пристрої. Порушення фізичної цілісності кільця могло зупинити роботу мережі, якщо не застосовувалися додаткові механізми резервування. Топологію «кільце» було наведено на рисунку 4.3.

Отже, топологія «зірка» в Ethernet є більш гнучкою та стійкою до локальних пошкоджень, тоді як «кільце» у Token Ring створює більшу залежність від цілісності всієї структури.[9]

4.2.2 Порівняння надійності

Надійність сучасного Ethernet значно зросла завдяки використанню комутаторів, які ізолюють трафік між портами. Відмова одного пристрою або кабелю зазвичай не впливає на інші сегменти мережі. Додатково можуть застосовуватися технології резервування каналів і обладнання.

Token Ring мав перевагу у відсутності колізій, оскільки передача відбувалася лише при наявності токена. Проте його залежність від безперервності кільця робила мережу вразливою до фізичних пошкоджень. Для підвищення

надійності впроваджувалися складніші рішення, що збільшувало вартість системи.

З практичної точки зору Ethernet є більш відмовостійким та простішим в експлуатації, але це залежить від його стандарту. Наприклад, стандарти 10Base-5 та 10Base-2 будуть менш надійні, ніж стандарт 10Base-T, але все одно надійніше, ніж Token Ring.

4.2.3 Порівняння швидкості передавання даних

Token Ring підтримував швидкості 4 або 16 Мбіт/с, причому всі пристрої в кільці повинні були працювати на однаковій швидкості.

Ethernet еволюціонував від 10 Мбіт/с до 100 Мбіт/с (Fast Ethernet), 1 Гбіт/с (Gigabit Ethernet) і значно вищих швидкостей. Це забезпечило йому перевагу в продуктивності та масштабованості.

4.2.4 Порівняння методів доступу

Ethernet у класичному варіанті використовував метод CSMA/CD – вузол перевіряє зайнятість каналу перед передачею. У разі одночасної передачі виникає колізія, після чого пристрої повторюють спробу через випадковий інтервал часу. У сучасних комутованих мережах із повнодуплексним режимом колізії практично відсутні.[10]

Token Ring застосовує метод Token Passing. У цій технології передача даних можлива тільки тоді, коли пристрій отримує спеціальний токен, який циркулює по мережі. Тільки той пристрій, який володіє маркером, може передавати дані, що виключає можливість зіткнень. Після завершення передачі маркер передається наступному пристрою в кільці. Цей метод забезпечує організований доступ до мережі, знижуючи ймовірність зіткнень, але вимагає додаткового часу на передачу маркера і може бути менш ефективним при малому навантаженні мережі.[11]

4.2.5 Обгрунтування побудови ЛКМ

Оскільки для окремих ЛКМ стандрат не задано, необхідно обрати оптимальний варіант серед наданих специфікацій Ethernet.

Існує кілька варіантів, серед яких 10Base-2, 10Base-5, 10Base-T та 10Base-FL та інші. Кожен з них має свої переваги та недоліки, залежно від умов експлуатації, вартості, простоти монтажу та технічних характеристик.

10Base-2 (Thin Ethernet) – використовує тонкий коаксіальний кабель типу RG-58 та забезпечує швидкість передавання даних 10 Мбіт/с. Максимальна довжина одного сегмента становить 185 м, при цьому до одного сегмента може бути підключено не більше 30 вузлів. Підключення здійснюється за шинною топологією із застосуванням BNC-конекторів і термінаторів на кінцях кабелю. Перевагами стандарту є відносно низька вартість і простота розгортання невеликих мереж. Водночас істотним недоліком є залежність усієї мережі від цілісності кабелю: будь-яке пошкодження або неправильне термінування призводить до повної втрати працездатності сегмента.[12]

10Base-5 (Thick Ethernet) – базується на використанні товстого коаксіального кабелю типу RG-11. Довжина сегмента може досягати 500 м, а кількість підключених пристроїв до 100. Підключення вузлів здійснюється через спеціальні відгалужувачі (трансивери). Порівняно з 10Base-2 цей стандарт забезпечує більшу дальність і кращу стійкість сигналу, однак монтаж є складнішим, а кабель громіздким і дорогим. Через високу вартість та незручність інсталяції 10Base-5 сьогодні практично не використовується.[13]

10Base-T – передбачає застосування витої пари (UTP або STP) зі швидкістю 10 Мбіт/с. Максимальна довжина кабелю між пристроєм і концентратором або комутатором становить 100 м. Мережа будується за топологією «зірка», де кожен вузол має окремий кабель до центрального пристрою. Це значно спрощує пошук несправностей, модернізацію та розширення мережі. Пошкодження одного кабелю не впливає на роботу інших сегментів. Саме 10Base-T став основою для подальшого розвитку Fast Ethernet і Gigabit Ethernet.[14]

10Base-FL – використовує волоконно-оптичний кабель і забезпечує передавання даних на відстань до 2 км. Оптичне середовище гарантує високу завадостійкість, електробезпеку та мінімальні втрати сигналу. Цей стандарт доцільний для об’єднання будівель або роботи в умовах сильних електромагнітних перешкод. Недоліком є вища вартість кабелю та активного обладнання.

Вибір стандарту 10Base-T для побудови локальної комп’ютерної мережі є найбільш раціональним з огляду на баланс вартості, надійності та зручності експлуатації. Використання витої пари спрощує прокладання кабельної інфраструктури, а зіркоподібна топологія забезпечує ізоляцію несправностей. Обладнання для 10Base-T є доступним, недорогим і сумісним із сучасними рішеннями Ethernet, що дозволяє без суттєвих витрат модернізувати мережу в майбутньому.

Таким чином, 10Base-T є оптимальним рішенням для більшості локальних мереж невеликого та середнього масштабу, особливо там, де важливі простота адміністрування та можливість подальшого розвитку інфраструктури.

Таблицю зі вказаними стандартами ЛКМ та розрахованими значеннями РС та протяжності наведено нижче (див. табл4 2.3)

Таблиця 4.3 – Характеристики гілок корпоративної КМ.

Варіант	Параметр ЛКМ	Номер гілки графа (ЛКМ)							
		1	2	4	5	6	7	10	11
3	Стандарт	BT	BT	BT	B5	B2	BT	TR	BT
	Кількість PC, N	27	27	13	93	33	37	18	23
	Протяжність, L	60 м	85 м	90 м	775 м	204 м	80 м	41 м	46 м

Корпоративна мережа буде налічувати 271 РС.

4.2.6 Обґрунтування будови модемної лінії

Для організації зв’язку між маршрутизаторами корпоративної комп’ютерної мережі та глобальної мережі Internet доцільно використовувати оптоволоконну лінію зв’язку.

Вибір оптичного середовища передачі даних обумовлений низкою технічних та експлуатаційних переваг.

По-перше, волоконно-оптичний кабель забезпечує значну пропускну здатність, що дозволяє передавати великі обсяги даних на високих швидкостях (від сотень Мбіт/с до десятків Гбіт/с і більше). Це особливо важливо для корпоративних мереж, де одночасно працює велика кількість користувачів, серверів, хмарних сервісів та систем відеоконференцій.

По-друге, оптоволокно характеризується мінімальними втратами сигналу та можливістю передачі даних на великі відстані без суттєвого погіршення якості. Це дозволяє забезпечити стабільний канал зв’язку між об’єктами, навіть якщо вони територіально віддалені.

По-третє, оптичний кабель є повністю нечутливим до електромагнітних завад. На відміну від мідних ліній, він не піддається впливу електричних наводок, імпульсних перешкод та грозових розрядів. Це підвищує надійність роботи мережі та зменшує ризик пошкодження активного обладнання.

Також важливим фактором є перспективність технології. Оптоволоконна інфраструктура дозволяє в подальшому збільшувати швидкість передачі даних без повної заміни кабельної системи – достатньо модернізувати лише активне обладнання.

З урахуванням високої пропускної здатності, завадостійкості, надійності, безпеки та можливості масштабування, використання волоконно-оптичної лінії зв’язку є технічно обґрунтованим і найбільш доцільним рішенням для підключення корпоративної мережі до глобальної мережі Internet.

Технологія PON (Passive Optical Network)

PON – це пасивна оптична мережа, яка забезпечує передавання даних від одного центрального вузла до багатьох кінцевих користувачів через оптичне волокно без активних електронних компонентів між ними. «Пасивна» означає, що на проміжних ділянках мережі немає підсилювачів, повторювачів або комутаторів, які потребують живлення.[15]

Принцип роботи PON

Центральний вузол OLT (Optical Line Terminal)

В OLT знаходиться активне обладнання, яке формує оптичний сигнал для передачі даних у мережу та обробляє отриманий сигнал від користувачів. OLT зазвичай розташовується в дата-центрі або центральному серверному приміщенні.

Оптичне волокно магістралі

Основна оптична магістраль йде від OLT і передає сигнал до пасивних розгалужувачів. Волокно не потребує електроживлення і не піддається впливу електромагнітних завад.

Пасивний спліттер (Splitter)

Спліттер розділяє один оптичний сигнал на декілька (наприклад, 1:16 або 1:32), забезпечуючи підключення кількох кінцевих точок до однієї магістралі. Спліттер не має живлення і працює повністю пасивно.

Кінцеві пристрої (ONT/ONU, Optical Network Terminal/Unit)

Кожен офіс, робоче місце або клієнт підключається через ONT/ONU. Ці пристрої перетворюють оптичний сигнал у електричний для локальної мережі (Ethernet, Wi-Fi тощо).

У підсумку, PON – це ефективне, економічне і надійне рішення для побудови корпоративних та широкомасштабних оптичних мереж, яке поєднує високу пропускну здатність з простотою експлуатації.

2.3 Перелік обладнання та матеріалів

Одним з найважливіших етапів є побудова кабельної схеми корпоративної КМ, що наводиться у додатку А. За поданою кабельною схемою будується таблиця 4.4 в якій розписана кількість необхідного обладнання.

Згідно з розрахунками (30+n)%, середня протяжність AUI кабелю становить 16.5 метрів.

Таблиця 4.4 – Перелік необхідного обладнання для будівництва корпоративної КМ

№	Назва обладнання	Одиниця вимірювання	Кількість одиниць обладнання								
			ЛКМ 1	ЛКМ 2	ЛКМ 4	ЛКМ 5	ЛКМ 6	ЛКМ 7	ЛКМ 10	ЛКМ 11	Всього по КМ
А. Мережні пристрої											
1	Трансивер 10Base-5	Шт				96					96
2	Мережевий комутатор з 24 портами RJ-45	Шт	2	2	1			2		1	8
3	Міст Ethernet з портами DB-15	Шт				1					1
4	Повторювач Ethernet з портами BNC	Шт					1				1
5	NIC Ethernet з портом DB-15	Шт				94					94
6	NIC Ethernet з портом RJ-45	Шт	28	28	15			38	19	24	152
7	N-термінатор	Шт				4					4
8	BNC термінатор	Шт					4				4

Продовження таблиці 2.4

№	Назва обладнання	Одиниця вимірювання	Кількість одиниць обладнання								
			ЛКМ 1	ЛКМ 2	ЛКМ 4	ЛКМ 5	ЛКМ 6	ЛКМ 7	ЛКМ 10	ЛКМ 11	Всього по КМ
9	NIC з портом BNC	Шт					34				34
10	MSAU з 8 портами RJ-45	Шт							3		3
Б. З'єднувачі											
11	N-Connectors	Шт				4					4
12	N Barrel Connector	Шт				4					4
13	BNC	Шт					68				68
14	BNC-T	Шт					36				36
15	RJ-45	Шт	58	58	30			78	38	48	314
16	DB-15	Шт				192					192
В. Кабелі											
17	Кабель AUI	М				96 * 16.5 = 1584					1584
18	RG-11	М				775					775
19	RG-58	М					204				204
20	Неекранований кабель Тип 3	М							19 * 41 = 779		779
21	Кабель 2 пари UTP 3-категорії	М	29 * 60 = 1740	29 * 85 = 2465	15 * 90 = 1350			39 * 80 = 3120		24 * 46 = 1104	9779
Г. Комп'ютери											
22	ПК Pentium 200MHz RAM 16MB	Шт	27	27	13	93	33	37	18	23	271
Д. Мережні ОС											
23	Windows 95	Шт	27	27	13	93	33	37	18	23	271

4.4 Просторові показники сигналу в ЛКМ

Згідно стандарту, найбільша протяжність мережі між двома крайніми станціями не повинна перевищувати 2500 метрів. При не виконанні даної

рекомендації мережа буде робочою, якщо значення часу подвійного оберту сигналу та скорочення між кадрового інтервалу не перевищує норми.

У побудованій схемі корпоративної КМ найвіддаленішими вузлами між собою є РС 48-93 в ЛКМ №5 та РС 16-27 в ЛКМ №2, відстань між ними займає 1276 метрів. Задовольняє стандартам та не перевищує допустимі 2500 метрів

(16.5 * 4 + 775) + (90 * 2) + (85 * 3) = 1276 метрів

Для розрахунку затримок часу мережевого обладнання використовується таблиця 4.5

Таблиця 4.5 – Дані для розрахунку часу подвійного оберту

Тип сегменту	База лівого сегменту	База проміжного сегменту	База правого сегменту	Затримка середовища на 1 м	Максимальна довжина сегменту
10Base-5	11.8	46.5	169.5	0.0866	500
10Base-2	11.8	46.5	169.5	0.1026	185
10Base-T	15.3	42.0	165.0	0.113	100
10Base-FL	12,3	33,5	156,5	0,1	2000
AUI (> 2 м)	0	0	0	0.1026	2+48

Розрахунок PDV:

Так як найвіддаленіші мережі проєктованої КМ мають різні типи мереж, тому потрібно провести подвійний розрахунок:

Перший розрахунок PDV(лівий сегмент – ЛКМ №2)

Лівий сегмент 2(10Base-T)

15.3 + 85 * 3 * 0.113 = 44.115 bt

Проміжний сегмент 4(10Base-T)

42.0 + 90 * 2 * 0.113 = 62.34 bt

Правий сегмент 5(10Base-5)

169.5 + 775 * 0.0866 + 16.5 * 4 * 0.1026 = 243.3866

Сума PDV = 349.8416 bt

Другий розрахунок PDV(лівий сегмент – ЛКМ №5)

Лівий сегмент 5(10Base-5)

11.8 + 775 * 0.0866 + 16.5 * 4 * 0.1026 = 85.6866 bt

Проміжний сегмент 4(10Base-T)

42.0 + 90 * 2 * 0.113 = 62.34 bt

Правий сегмент 2(10Base-T)

165.0 + 85 * 3 * 0.113 = 193.815 bt

Сума PDV = 341.8416 bt

Оскільки PDV у двох підрахунках не перевищують максимально допустиме значення 575bt, ця мережа відповідає критерію часу подвійного оберту сигналів.[16]

При розрахунку часу затримки сигналу для ЛКМ №5, ЛКМ №4 та ЛКМ №2 використовується таблиця 4.6

Таблиця 4.6 – Дані для розрахунку часу затримки

Назва обладнання	Приблизна затримка сигналу
Товстий коаксіальний кабель типу RG11	4.33 мкс/км
Тонкий коаксіальний кабель типу RG58	5.14 мкс/км
Вита пара неекранована (UTP) та екранована (STP)	5.7 мкс/км
Оптичне волокно товщиною 62.5/125 мікрон	5.0 мкс/км
Повторювач, багатопортовий повторювач (HUB)	0.1 мкс
Оптично-волоконний повторювач чи HUB	1.55 мкс
Оптично-волоконний трансивер	0.2 мкс

Розрахунок затримки часу:

ЛКМ №5: (775 / 1000) * 4.33 + 0.1 = 3.45575 мкс

ЛКМ №4: (90 * 2 / 1000) * 5.7 + 0.1 = 1.126 мкс

ЛКМ №2: (85 * 3 / 1000) * 5.7 + 0.1 * 2 = 1.6535 мкс

Так як швидкість заданих стандартів 10Мбіт/с, при якому 0.1 мкс дорівнює 1bt, тоді час затримки буде наступний:

ЛКМ №5: 0.345575 bt

ЛКМ №4: 0.1125 bt

ЛКМ №2: 0.16535 bt

Для обчислення PVV використовують значення максимальних зменшень міжкадрового інтервалу при проходжені повторювачів різних фізичних середовищ, рекомендовані IEEE та наведені в таблиці 4.7

Таблиця 4.7 – Зменшення міжкадрового інтервалу повторювача

Тип сегменту	Передаючий сегмент, bt	Проміжний сегмент, bt
10Base-2 або 10Base-5	16	11
10Base-FB	-	2
10Base-FL	10.5	8
10Base-T	10.5	8

Використовуючи дані таблиці 2.7 можна провести наступні розрахунки міжкадрового інтервалу.

Оскільки найбільш віддалені мережі проєктованої КМ мають різні типи, необхідно виконати подвійний розрахунок:

Перший розрахунок(лівий сегмент – ЛКМ №2)

Передаючий сегмент 2(10Base-T): 10.5 + 0.16535 = 10.66535 bt

Проміжний сегмент 4(10Base-T): 8 + 0.1125 = 8.1125 bt

PVV = 10.66535 + 8.1125 = 18.77785 bt

Другий розрахунок:

Передаючий сегмент 5(10Base-5): 16 + 0.345575 = 16.345575 bt

Проміжний сегмент 4(10Base-T): 8 + 0.1125 = 8.1125 bt

PVV = 16.345575 + 8.1125 = 24.458075 bt

Розрахунок PVV підтвердив, що проєктована КМ працюватиме коректно, оскільки зменшення міжкадрового інтервалу при проходженні повторювачів через різні фізичні середовища не перевищує максимально допустиме значення – 49 бітових інтервалів.[17]

4.5 Адресація та маршрутизація

Важливим етапом при побудові корпоративної комп’ютерної мережі є організація системи IP-адресації мереж і вузлів. Коректно спроєктована схема адресації забезпечує логічну структурованість мережі, спрощує адміністрування, підвищує рівень безпеки та дозволяє ефективно реалізувати маршрутизацію трафіку між сегментами мережі та при виході до глобальної мережі Internet.

IP-адресація визначає унікальні логічні адреси для кожного мережевого пристрою: робочих станцій, серверів, маршрутизаторів та інших вузлів. Для корпоративних мереж, як правило, використовуються приватні діапазони IP-адрес (згідно RFC 1918), що дозволяє раціонально використовувати адресний простір та забезпечує ізоляцію внутрішньої мережі від зовнішнього середовища.

Під час проєктування адресного простору необхідно враховувати кількість підмереж, кількість пристроїв у кожному сегменті, необхідність резервування адрес для майбутнього розширення, виділення окремих діапазонів для серверів та мережевого обладнання.

Кожна підмережа повинна мати мережеву адресу, маску підмережі, діапазон допустимих IP-адрес для хостів, адресу шлюзу за замовчуванням.

Маршрутизація в проєктованій корпоративній КМ здійснюється за допомогою маршрутизаторів, які забезпечують обмін трафіком між підмережами, а також вихід до ГКМ Internet через зовнішній інтерфейс. ля доступу до глобальної мережі використовується механізм трансляції мережевих адрес NAT, що дозволяє внутрішнім приватним IP-адресам взаємодіяти з публічними ресурсами.

У таблиці 4.8 наведено структуровані дані для IP-адресації та маршрутизації робочих станцій у корпоративній комп’ютерній мережі.

Таблиця 4.8 – Дані для адресації та маршрутизації

Номер варіанту, К	IP-адреси корпоративної КМ	IP-адреси вузлів маршрутизаторів Internet та КМ	Internet – адреси деяких вузлів ЛКМ
3	200.30.20.X 200.30.21.X 200.30.22.X 200.30.23.X	198.88.34.17 198.88.34.18	08:C1:01:00:A3:B7 08:C1:11:09:08:C1 08:C1:11:31:C8:F1 08:C1:1B:18:93:28

4.5.1 Побудова адресної таблиці прозорого моста

Алгоритм "прозорий міст" отримав свою назву через те, що його присутність і функціонування непомітні для пристроїв у мережі.

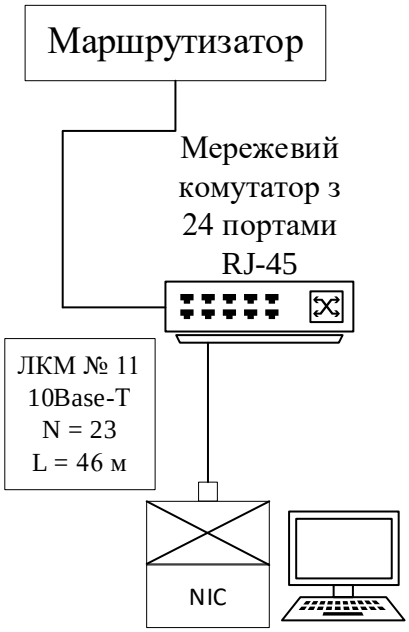
Міст автоматично створює свою адресну таблицю, аналізуючи трафік, який проходить через його порти. Він визначає адреси джерел кадрів і, на основі цієї інформації, встановлює, до якого сегмента мережі належить певний вузол.

Кожен порт моста працює як кінцева точка для сегмента мережі. Спочатку міст не має даних про те, які MAC-адреси прив'язані до його портів. Тому на початковому етапі він передає всі отримані кадри на всі порти, окрім того, з якого вони надійшли. У процесі такої передачі міст також вивчає адреси джерел кадрів і заповнює свою таблицю, встановлюючи відповідність між MAC-адресами та портами.

Згодом, використовуючи цю таблицю, міст оптимізує маршрутизацію трафіку. Коли кадр надходить на один із портів, міст шукає в таблиці адресу призначення. Якщо адреса знайдена, кадр пересилається лише на відповідний порт (за винятком порту, з якого він був отриманий). Якщо адреса не знайдена, кадр розсилається на всі порти, окрім порту входу. Широкомовні й багатоадресні повідомлення також передаються на всі порти.

Прозорий міст ефективно ізолює трафік усередині кожного сегмента, що зменшує обсяг видимого трафіку і, відповідно, скорочує час реакції мережі. Ступінь оптимізації залежить від співвідношення міжсегментного трафіку до загального, а також від обсягу широкомовного і багатоадресного трафіку.

На рисунку 4.6 подано приклад у вигляді кабельної схеми ЛКМ №11



PC 1-23

Рисунок 4.6 – ЛКМ №11

Для побудови таблиці слід використовувати MAC-адреси, які було наведено у таблиці 4.8

В таблиці 4.9 наведено приклад адресної таблиці прозорого моста для ЛКМ №11.

Таблиця 4.9 – Часткова адресна таблиця прозорого моста ЛКМ №11

Порт мосту	MAC-адреса
PC-1	08:C1:01:00:A3:B7
PC-2	08:C1:11:09:08:C1
PC-3	08:C1:11:31:C8:F1
PC-4	08:C1:1B:18:93:28

4.5.2 Визначення кількості адрес класу С

Для проєктованої корпоративної КМ передбачається статичний метод адресації IP-вузлів мережі.

Для адресації мережевого простору корпоративної КМ необхідно використовувати IP-адреси надані в таблиці 2.8, де X використовується для адресації вузлів.

У таблиці 4.10 показані адреси класу C, необхідні для корпоративної КМ та маски підмереж

Таблиця 4.10 – Адреси класу C

ЛКМ	IP-адреси класу C	Маска підмережі
1	200.30.20.0 – 200.30.20.127	255.255.255.128
2	200.30.20.128 – 200.30.20.255	255.255.255.128
4	200.30.21.0 – 200.30.21.127	255.255.255.128
5	200.30.21.128 – 200.30.21.255	255.255.255.128
6	200.30.22.0 – 200.30.22.127	255.255.255.128
7	200.30.22.128 – 200.30.22.255	255.255.255.128
10	200.30.23.0 – 200.30.23.127	255.255.255.128
11	200.30.23.128 – 200.30.23.255	255.255.255.128

4.5.3 Таблиці маршрутизації

Таблиці маршрутизації для маршрутизаторів A, B та C зазначено в таблицях 4.11, 4.12, 4.13 відповідно.

Таблиця 4.11 – Маршрутна таблиця маршрутизатора A

IP-адреса мережі призначення	Маршрутизація	IP-адреса шлюза	Вихідний мережний інтерфейс
200.30.20.0	Пряма	-	ie0
200.30.20.128	Пряма	-	ie1
200.30.21.0	Пряма	-	ie2
200.30.21.128	Не пряма	200.30.21.2	ie2
200.30.22.0	Не пряма	200.30.21.2	ie2
200.30.22.128	Не пряма	200.30.21.2	ie2

Таблиця 4.12 – Маршрутна таблиця маршрутизатора B

IP-адреса мережі призначення	Маршрутизація	IP-адреса шлюза	Вихідний мережний інтерфейс
200.30.21.0	Пряма	-	ie0
200.30.21.128	Пряма	-	ie1
200.30.22.0	Пряма	-	ie2
200.30.22.128	Пряма	-	ie3
200.30.20.0	Не пряма	200.30.21.1	ie0
200.30.20.128	Не пряма	200.30.21.1	ie0

Таблиця 4.13 – Маршрутна таблиця маршрутизатора C

IP-адреса мережі призначення	Маршрутизація	IP-адреса шлюза	Вихідний мережний інтерфейс
200.30.23.0	Пряма	-	ie0
200.30.23.128	Пряма		ie1
default	не пряма	198.88.34.17	pp0

Висновки до розділу 4

У даному розділі було розглянуто основні аспекти проєктування корпоративної комп’ютерної мережі. На основі вихідних даних виконано аналіз структури корпоративної мережі та сформовано граф проєктованої мережі з використанням заданих локальних комп’ютерних мереж і маршрутизаторів.

Визначено особливості взаємодії окремих сегментів мережі та способи їх підключення до глобальної мережі Internet.

Проведено аналіз сучасних мережевих технологій та стандартів передачі даних. Для побудови локальних сегментів мережі обрано технологію Ethernet, яка забезпечує високу швидкість передавання інформації, сумісність обладнання різних виробників та можливість подальшого масштабування мережевої інфраструктури.

Розглянуто основні технічні параметри локальних мереж, виконано вибір типів середовища передачі даних та мережевого обладнання. Для підключення робочих станцій передбачено використання структурованої кабельної системи на основі витої пари, а для підключення до глобальної мережі та магістральних з’єднань — волоконно-оптичних каналів зв’язку.

Запропоновані технічні рішення забезпечують необхідний рівень продуктивності, надійності та відмовостійкості мережі, а також створюють умови для стабільної роботи серверного обладнання, систем моніторингу та програмного забезпечення діагностики накопичувачів.

5 ПРОЕКТУВАННЯ СЕРВЕРНОЇ КІМНАТИ

5.1 Вихідні дані для проектування серверної кімнати

Для побудови локальної хмари необхідно забезпечити такі ресурси:

- кількість паралельних операцій – 40000;
- оперативна пам'ять – 256000 ГБ;
- дисковий простір – 16000 ТБ;
- розміри приміщення – 6 × 9 м;
- висота приміщення – 2,7 м;
- допустиме навантаження на підлогу – 300 кг/м².

5.2 Вибір обладнання для серверної кімнати

Для отримання максимальної кількості обчислювальних ресурсів доцільно використати:

Сервер Специфікація 2

- 128 ядер;
- 256 ГБ ОЗП;
- 2 ТБ дискового простору;
- потужність 800 Вт;
- висота 3U;
- вага 9 кг.

Комутатор 4

- 48+2 порти;
- потужність 180 Вт;
- висота 2U;
- вага 6 кг.

ДБЖ 4

- потужність 10500 Вт;
- 16 розеток;

- висота 4U;
- вага 18 кг.

5.3 Планування розміщення обладнання у серверній кімнаті

Приймаємо стандартну серверну стійку 42U.

Розміщуємо:

- 10 серверів Специфікація 2;
- 1 комутатор 4;
- 1 ДБЖ 4.

Займана висота:

10×3U+2U+4U=36U

Отже, обладнання поміщається в стійку 42U.

Ресурси однієї стійки

Кількість ядер:

10×128=1280

Оперативна пам'ять:

10×256=2560 ГБ

Дисковий простір:

10×2=20 ТБ

Споживана потужність:

10×800+180=8180 Вт

ДБЖ потужністю 10500 Вт повністю покриває навантаження.

Вага стійки:

10×9+6+18+125=239 кг

(125 кг – вага серверної стійки).

Визначення необхідної кількості стійок

За кількістю операцій

40000/1280=31,25

Потрібно:

32 стійки

За оперативною пам'яттю

256000/2560=100

Потрібно:

100 стійок

За дисковим простором

16000/20=800

Потрібно:

800 стійок

Найбільше значення отримано за дисковим простором.

Отже необхідно:

800 серверних стійок

Розміщення обладнання у серверній кімнаті

Площа приміщення:

$S=6\times9=54$ м^2

Приймаємо розміщення:

- 3 ряди;
- по 10 стійок у кожному ряду.

Усього:

30 стійок в кімнаті

Для розміщення всіх стійок потрібно:

800/30=26,7

Приймаємо:

27 серверних кімнат

Загальна кількість стійок:

27×30=810

Перевірка навантаження на підлогу

Максимально допустиме навантаження:

54×300=16200 кг

Фактична вага обладнання:

30×239=7170 кг

Питоме навантаження:

7170/54=132,8 кг/м^2

32,8<300 кг/м^2

Отже вимоги до навантаження на підлогу виконуються.

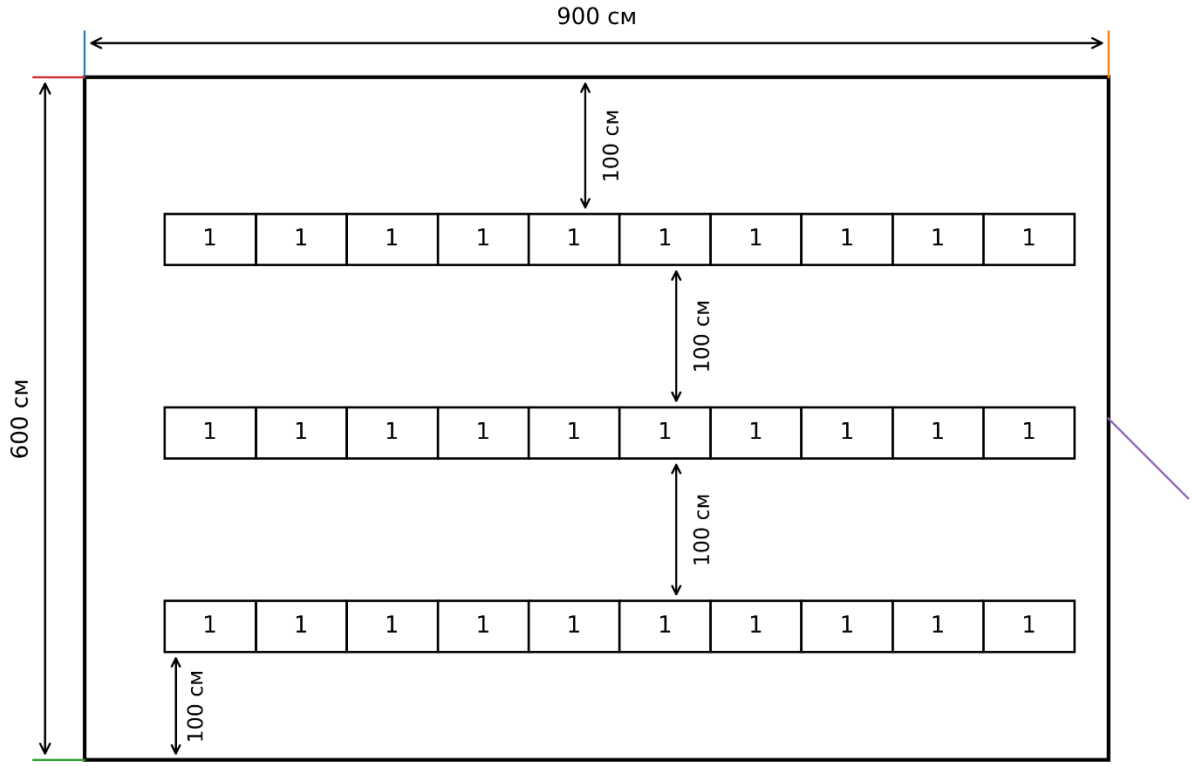


Рисунок 5.1 – Схема розміщення серверних стійок у приміщенні 6×9 м

У приміщенні розміщено 30 серверних стійок: три ряди по десять стійок. Між рядами передбачено проходи шириною 100 см, що забезпечує доступ до обладнання для монтажу, обслуговування та вентиляції. Така схема відповідає прийнятим розрахункам, за якими одна серверна кімната вміщує 30 стійок, а для повного забезпечення ресурсів локальної хмари потрібно 27 аналогічних приміщень.

Висновки до розділу 5

У п'ятому розділі виконано проєктування серверної інфраструктури локальної хмари відповідно до вимог варіанта 3. Було обрано сервери Специфікації 2, комутатори типу 4 та джерела безперебійного живлення типу 4. Встановлено, що одна серверна стійка забезпечує 1280 процесорних ядер, 2560 ГБ оперативної пам'яті та 20 ТБ дискового простору.

Для забезпечення необхідних ресурсів локальної хмари потрібно 800 серверних стійок. З урахуванням розміщення по 30 стійок у серверній кімнаті необхідно 27 серверних приміщень розміром 6×9 м. Проведений розрахунок показав, що навантаження на підлогу становить 132,8 кг/м², що не перевищує допустиме значення 300 кг/м². Отже, спроєктована серверна інфраструктура повністю задовольняє задані вимоги щодо продуктивності, пам'яті, дискового простору та безпечної експлуатації.

6 КОДУВАННЯ ІНФОРМАЦІЇ

6.1 Опис алгоритму

Одним із найпоширеніших методів виявлення та виправлення одиничних помилок є код Хемінга. Його основна ідея полягає у додаванні до інформаційних бітів спеціальних контрольних бітів, які розміщуються у кодовому слові на позиціях, що є степенями числа 2: 1, 2, 4, 8, 16 тощо. Завдяки цьому під час перевірки прийнятого кодового слова можна визначити позицію помилкового біта та виправити його шляхом інверсії [28].

Для кодування інформаційного слова довжиною m бітів необхідно визначити кількість контрольних бітів r. Значення r обирається так, щоб виконувалась умова:

$$2^r >= m + r + 1$$

де m – кількість інформаційних бітів, r – кількість контрольних бітів. Після визначення кількості контрольних бітів формується кодове слово, у якому позиції 1, 2, 4, 8 і т.д. резервуються для контрольних бітів, а решта позицій заповнюється інформаційними бітами [28, 29].

Кожен контрольний біт відповідає за перевірку певної групи позицій кодового слова. Група визначається за двійковим поданням номера позиції. Наприклад, контрольний біт у позиції 1 перевіряє всі позиції, у двійковому записі яких встановлений перший розряд, контрольний біт у позиції 2 – позиції з установленим другим розрядом, контрольний біт у позиції 4 – позиції з установленим третім розрядом. Значення контрольного біта обчислюється за парністю, тобто за допомогою операції додавання за модулем 2.

Під час декодування прийнятого кодового слова повторно обчислюються контрольні суми. Якщо всі перевірки дають нульовий результат, вважається, що помилки немає. Якщо хоча б одна перевірка виявляє невідповідність, формується синдром помилки. Значення синдрому у двійковому вигляді вказує номер позиції,

у якій виникла помилка. Після цього відповідний біт інвертується, і кодове слово вважається виправленим [28].

У межах даної роботи було розроблено програму кодування методом Хемінга мовою Python. Програма має графічний інтерфейс користувача та дозволяє виконувати кодування двійкових послідовностей, внесення штучної помилки, декодування, виправлення одиничної помилки, кодування текстового повідомлення, а також тестування за підготовленими наборами даних [20].

Алгоритм роботи розробленої програми наведено на рисунку 6.1.

На початковому етапі виконується запуск програми та ініціалізація головного вікна графічного інтерфейсу. Після цього користувач обирає один із доступних режимів роботи програми: кодування даних, декодування та виправлення, генерація помилки, тестування наборів даних, перегляд візуалізації та статистики або довідкової інформації.

У режимі кодування користувач вводить двійкову послідовність або завантажує її з файлу. Програма перевіряє коректність вхідних даних, після чого обчислює необхідну кількість контрольних бітів. Далі визначаються позиції контрольних бітів, формується структура кодового слова та розраховуються значення контрольних бітів за парністю. Після завершення обчислень програма виводить закодоване слово та показує розташування інформаційних і контрольних бітів.

У режимі декодування користувач вводить прийняте кодове слово. Програма обчислює синдром помилки шляхом повторної перевірки контрольних бітів. Якщо синдром дорівнює нулю, повідомляється, що помилка відсутня. Якщо синдром має ненульове значення, програма визначає позицію помилки та виправляє її шляхом інверсії відповідного біта. Після цього з виправленого кодового слова видаляються контрольні біти, а користувач отримує початкову інформаційну послідовність.

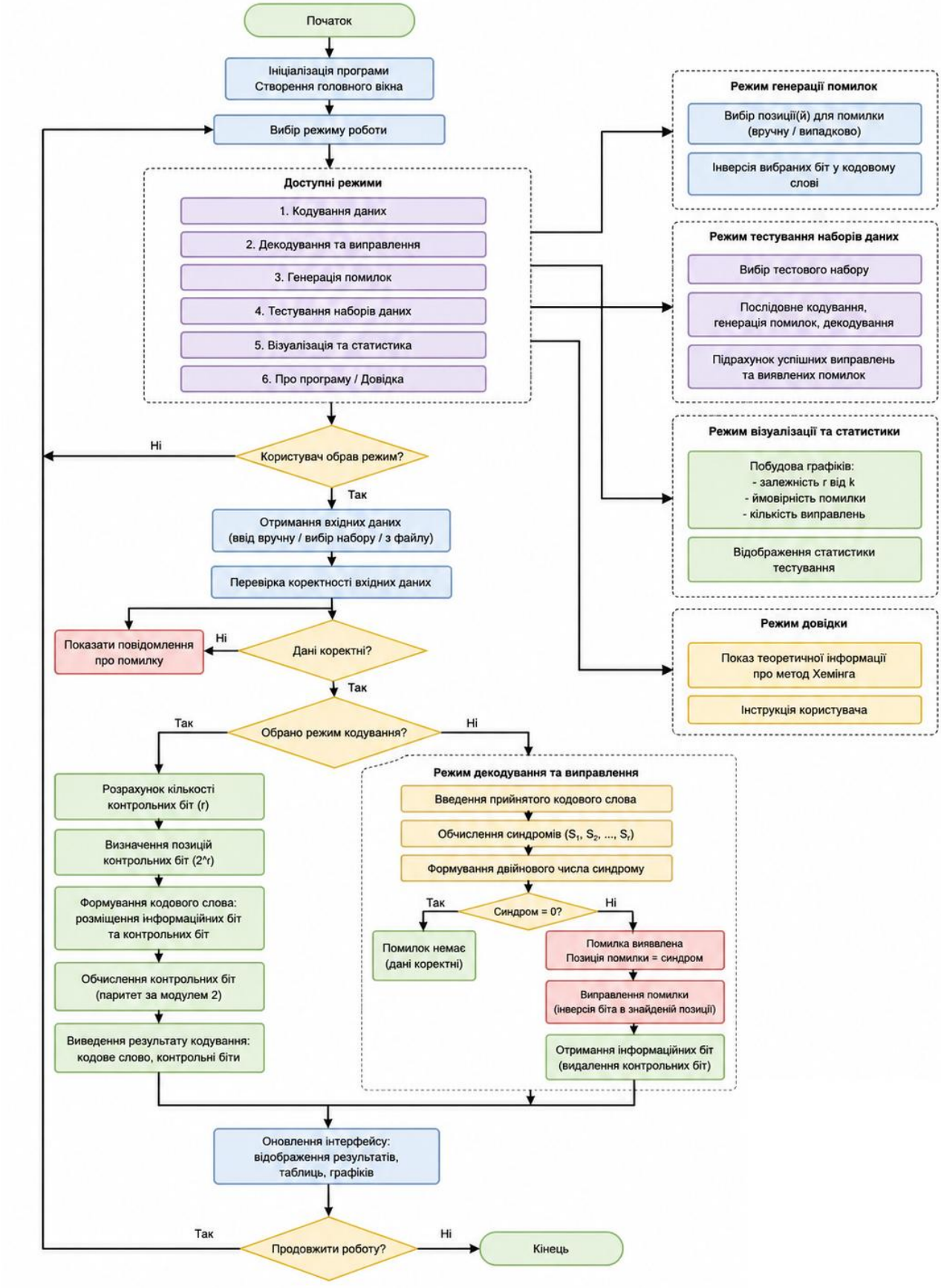


Рисунок 6.1 – Алгоритм роботи програми кодування методом Хемінга

Окремим етапом передбачено режим генерації помилок. У цьому режимі користувач може вручну задати позицію біта, у якому необхідно змінити значення. Це дозволяє на практиці перевірити здатність коду Хемінга виявляти та виправляти одиничні помилки.

Для зручності аналізу результатів у програмі реалізовано режим тестування наборів даних. Програма послідовно зчитує двійкові слова з текстового файлу, виконує їх кодування, за потреби вносить помилки, декодує отримані кодові слова та підраховує статистику успішних виправлень. Такий підхід дозволяє перевірити правильність роботи алгоритму не лише на одному прикладі, а на групі тестових даних.

Додатково в алгоритмі передбачено режим візуалізації та статистики. Він використовується для графічного відображення структури кодового слова, виділення контрольних бітів, позначення позиції помилки та побудови статистики тестування. Це робить роботу програми більш наочною та спрощує розуміння принципу дії коду Хемінга.

6.2 Опис роботи програми

Для демонстрації принципів завадостійкого кодування була розроблена програма кодування методом Хемінга мовою Python. Програмне забезпечення реалізоване у вигляді графічного застосунку з використанням бібліотеки Tkinter та призначене для кодування інформаційних повідомлень, моделювання помилок, декодування даних і аналізу результатів роботи алгоритму.

Головне вікно програми містить декілька функціональних вкладок, кожна з яких призначена для виконання окремих операцій із кодом Хемінга. Така структура дозволяє користувачеві послідовно виконувати всі етапи обробки інформації – від формування кодового слова до перевірки працездатності алгоритму на тестових наборах даних.

На рисунку 6.2 наведено головне вікно програми кодування методом Хемінга.

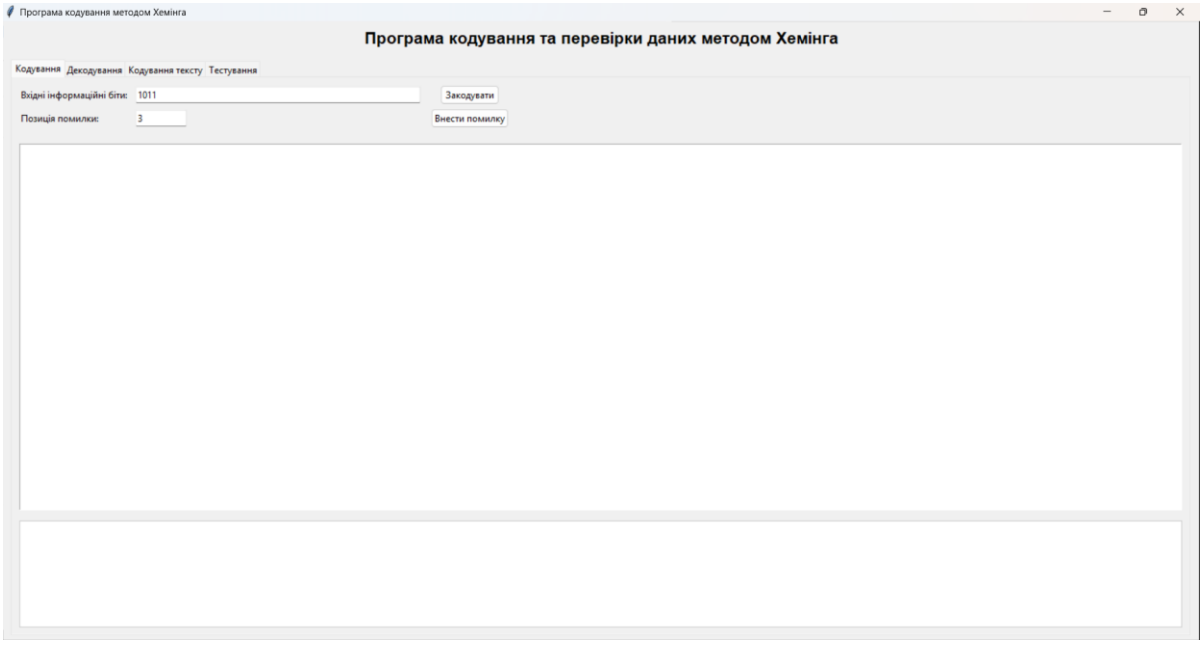


Рисунок 6.2 – Головне вікно програми кодування методом Хемінга

Першою вкладкою програми є режим кодування. У цьому режимі користувач вводить двійкову послідовність, після чого програма автоматично визначає необхідну кількість контрольних бітів та формує кодове слово Хемінга. Одночасно виконується візуалізація структури кодового слова, де контрольні біти виділяються окремим кольором, а також відображаються їх позиції.

Результат роботи режиму кодування наведено на рисунку 6.3.

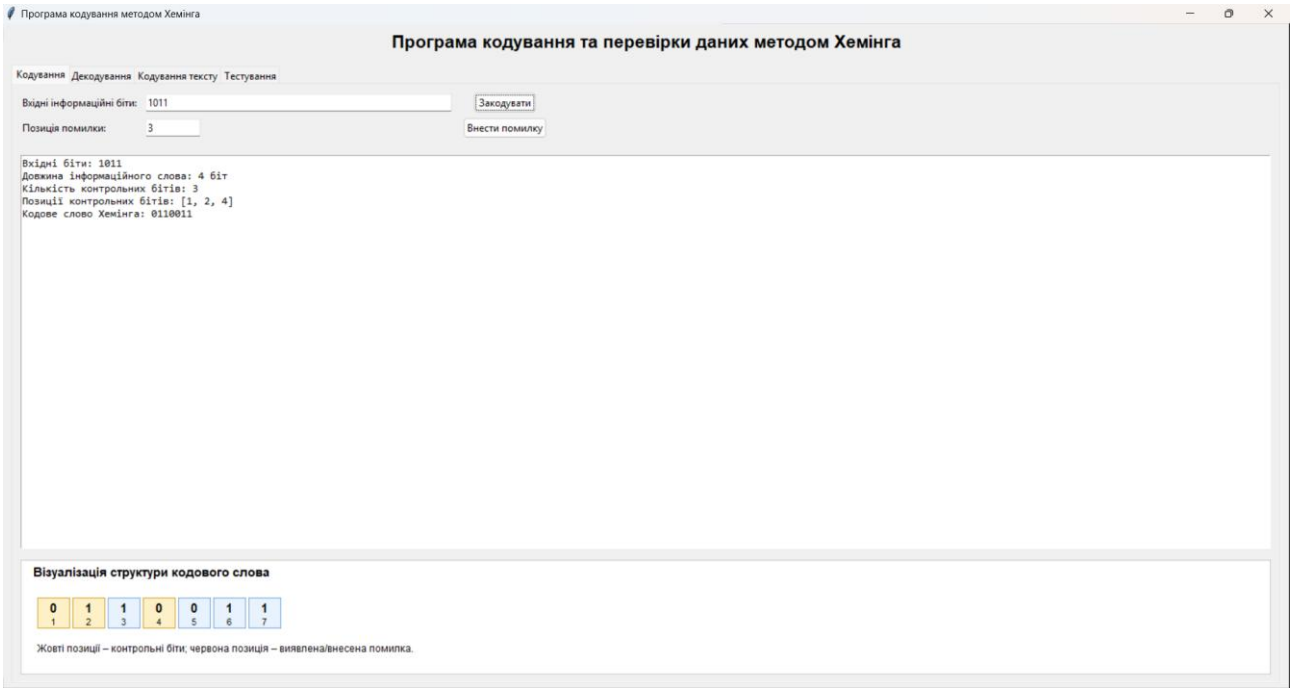


Рисунок 6.3 – Формування кодового слова методом Хемінга

Після створення кодового слова користувач може виконати моделювання помилки. Для цього задається позиція біта, значення якого буде змінено. Програма інвертує відповідний біт і формує пошкоджене кодове слово. Такий режим дозволяє перевірити здатність алгоритму виявляти та виправляти помилки під час передавання даних.

На рисунку 6.4 показано приклад внесення помилки до кодового слова.

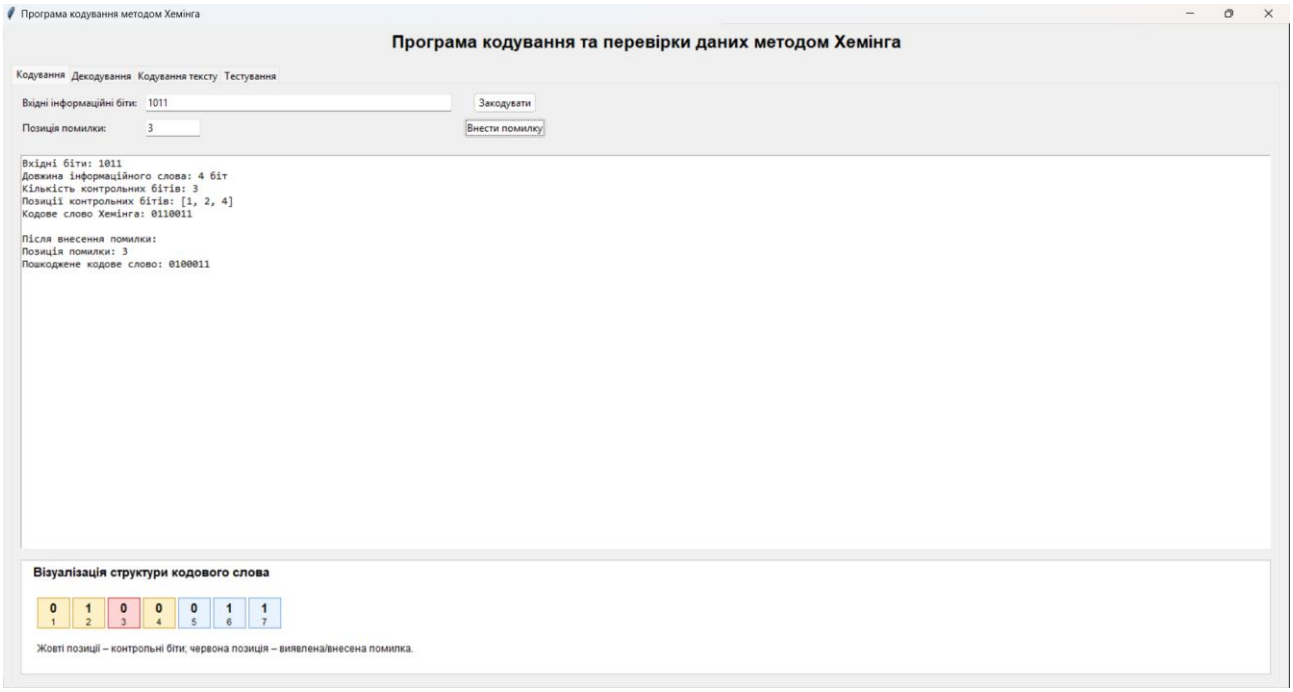


Рисунок 6.4 – Моделювання одиничної помилки у кодовому слові

Наступним режимом роботи є декодування. У даному режимі користувач вводить кодове слово або використовує сформоване раніше пошкоджене повідомлення. Програма виконує обчислення синдрому помилки, визначає її місце розташування та здійснює автоматичне виправлення. Після цього з кодового слова вилучаються контрольні біти і відновлюється початкова інформаційна послідовність.

Результат роботи режиму декодування наведено на рисунку 6.5.

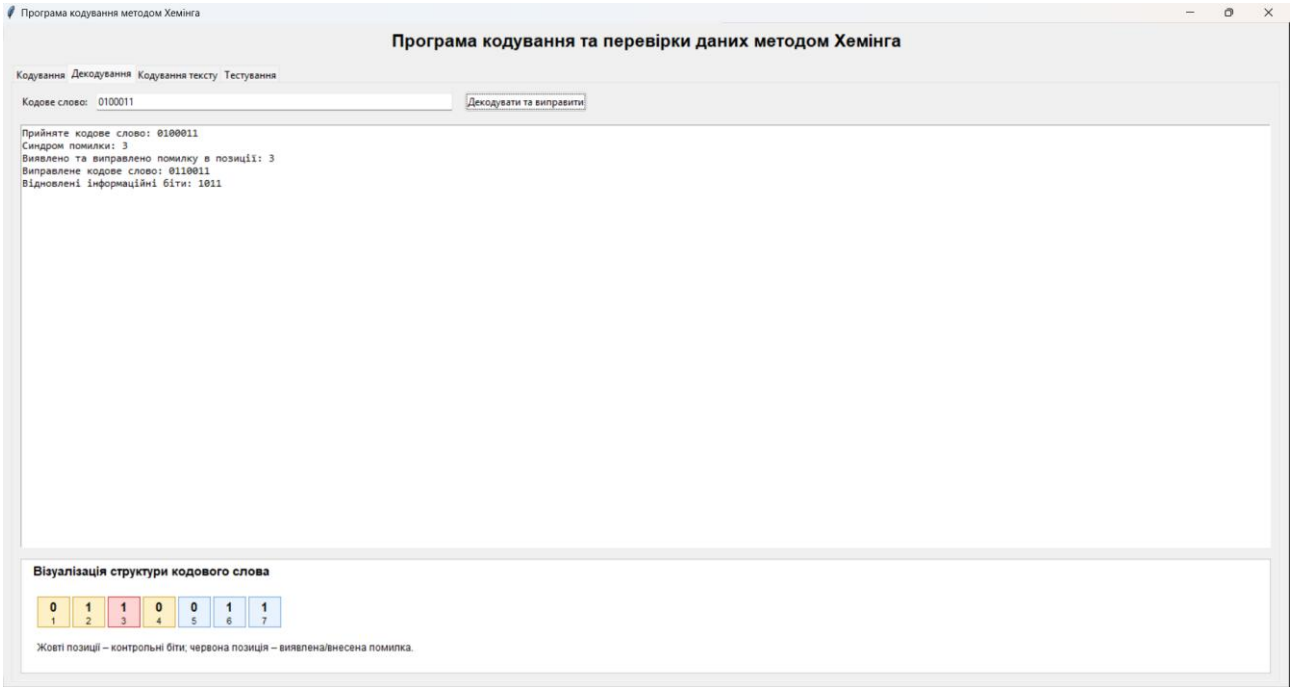


Рисунок 6.5 – Виявлення та виправлення помилки у кодовому слові

Окремою можливістю програми є кодування текстових повідомлень. Під час роботи в цьому режимі текстове повідомлення автоматично переводиться у двійковий код ASCII, після чого розбивається на блоки заданого розміру. Для кожного блоку формується власне кодове слово Хемінга, що дозволяє демонструвати використання алгоритму для передачі реальної текстової інформації.

На рисунку 6.6 представлено результат кодування текстового повідомлення.

Для перевірки працездатності алгоритму в програмі реалізовано режим тестування. Він дозволяє автоматично обробляти набори даних, які зберігаються у текстових файлах. Під час тестування виконується послідовне кодування повідомлень, внесення помилок, декодування та перевірка правильності відновлення початкових даних.

У процесі тестування програма формує статистичну інформацію про кількість оброблених блоків, число успішно виправлених помилок та кількість випадків без помилок.

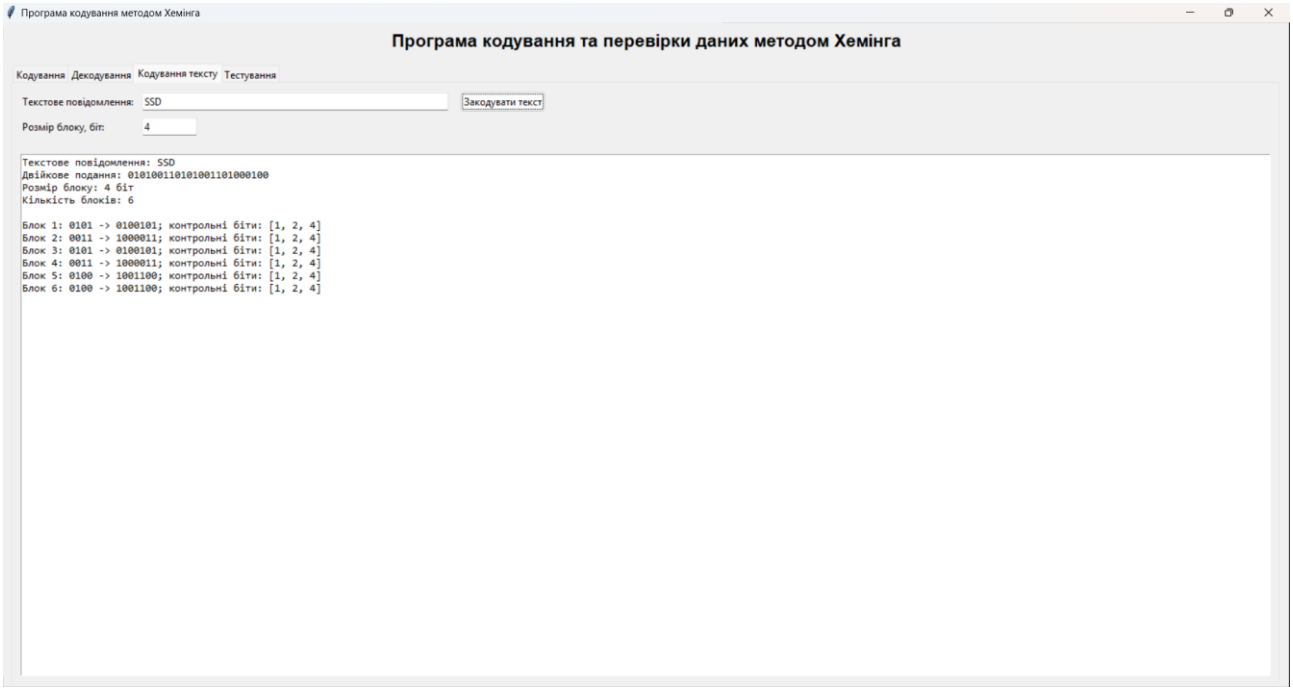


Рисунок 6.6 – Кодування текстового повідомлення методом Хемінга

Результати тестування наведено на рисунку 6.7.

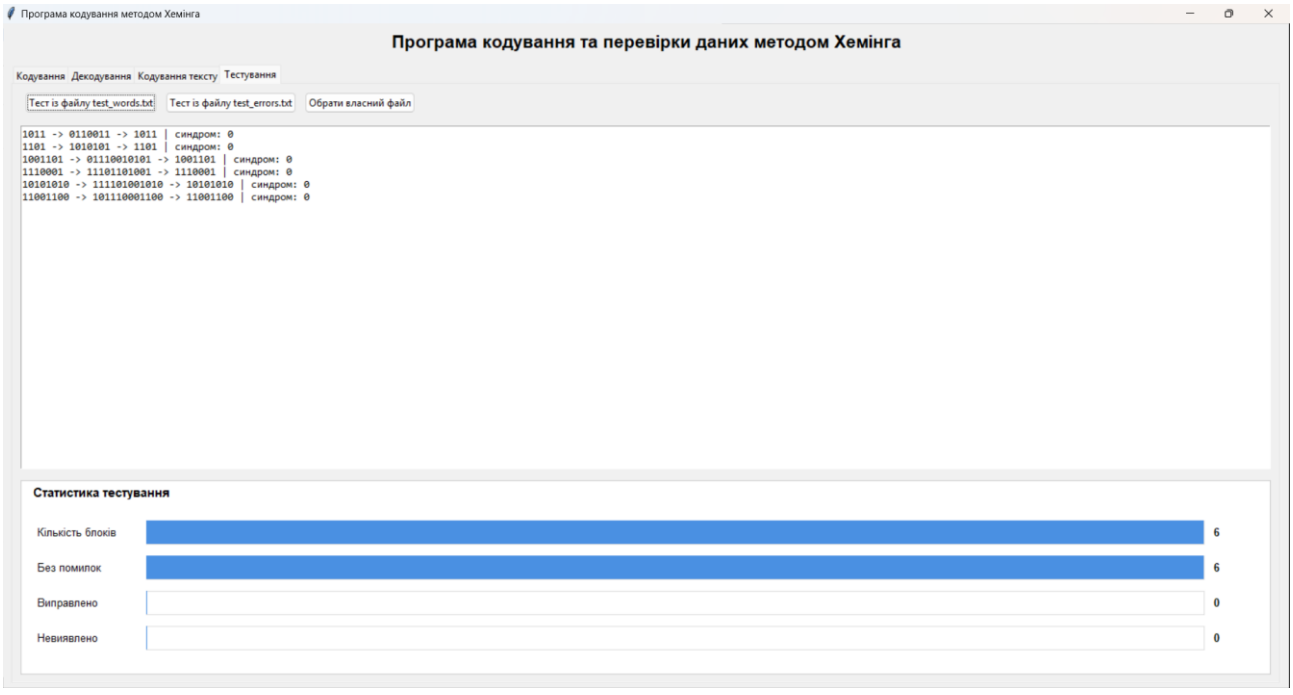


Рисунок 6.7 – Режим тестування наборів даних

Для підвищення наочності роботи алгоритму у програмі реалізовано графічну візуалізацію структури кодового слова та статистики тестування. Візуалізація дозволяє користувачу спостерігати розташування контрольних бітів,

позицію виявленої помилки та результати її виправлення. Крім того, відображаються графічні діаграми, що характеризують ефективність роботи алгоритму під час обробки тестових даних.

На рисунку 6.8 наведено приклад відображення статистичних результатів роботи програми.

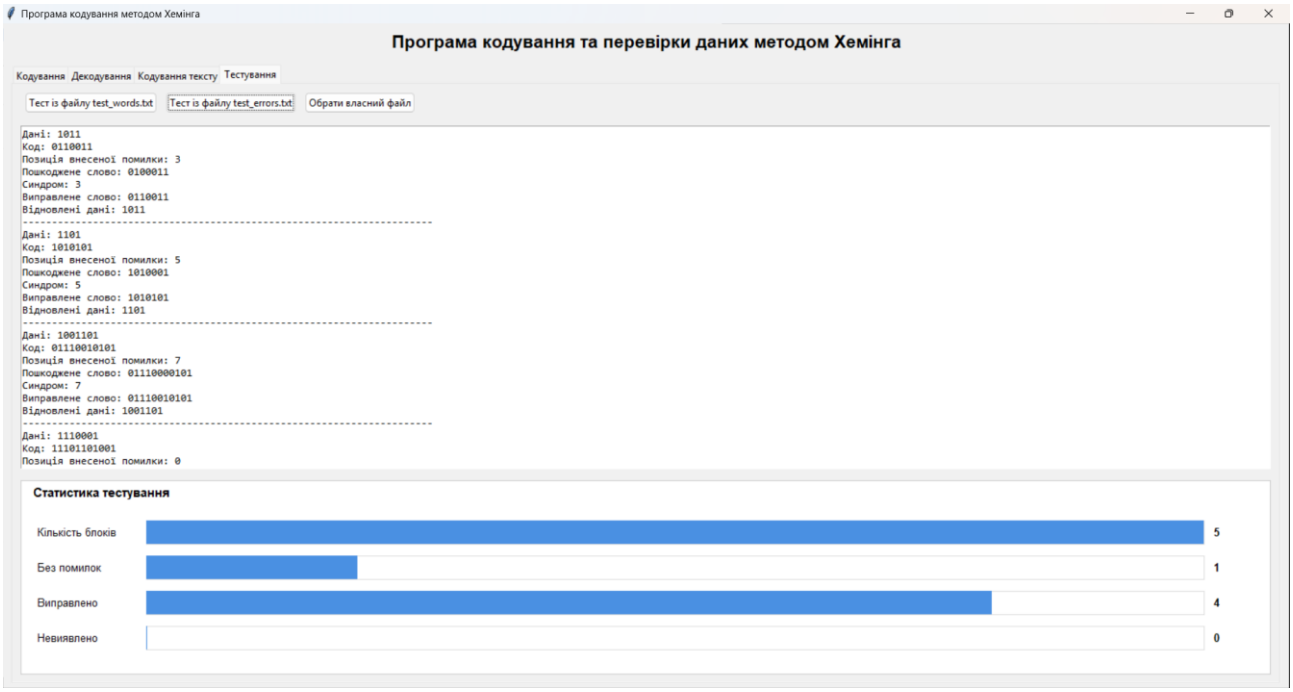


Рисунок 6.8 – Візуалізація результатів тестування

6.3 Опис коду програми

Програмне забезпечення для кодування інформації методом Хемінга реалізоване мовою Python із використанням бібліотеки Tkinter для створення графічного інтерфейсу користувача. Структура програми побудована за модульним принципом, що спрощує її супровід та подальшу модернізацію. Основними складовими програмного коду є функції кодування та декодування, модулі роботи з помилками, засоби обробки текстових повідомлень, механізми тестування та компоненти графічного інтерфейсу [20].

На початку програми реалізовано допоміжні функції для роботи з двійковими даними. Зокрема, функція is_power_of_two() використовується для

перевірки, чи є номер позиції степенем двійки. Такі позиції в алгоритмі Хемінга резервуються для контрольних бітів.

```
def is_power_of_two(number: int) -> bool:
    return number > 0 and (number & (number - 1)) == 0
```

Для перевірки коректності введених даних використовується функція only_binary(), яка контролює, щоб користувач вводив лише символи «0» та «1».

```
def only_binary(value: str) -> bool:
    return bool(value) and all(char in "01" for char in value)
```

Одним із ключових елементів програми є функція визначення кількості контрольних бітів. Вона реалізує класичну умову побудови коду Хемінга та автоматично обчислює необхідну кількість перевірочних розрядів залежно від довжини інформаційного повідомлення.

```
def required_parity_bits(data_length: int) -> int:
    r = 0

    while (2 ** r) < data_length + r + 1:
        r += 1

    return r
```

Основна операція кодування реалізована у функції hamming_encode(). Її завданням є формування кодового слова шляхом розміщення контрольних та інформаційних бітів у відповідних позиціях. Після формування структури кодового слова обчислюються значення контрольних бітів на основі перевірки парності.

```
def hamming_encode(data_bits: str):
    m = len(data_bits)
    r = required_parity_bits(m)
    n = m + r

    codeword = ["0"] * (n + 1)
```

Далі програма розміщує інформаційні біти між контрольними розрядами.

```
for position in range(1, n + 1):
    if is_power_of_two(position):
        codeword[position] = "0"
    else:
        codeword[position] = data_bits[data_index]
```

Після цього виконується розрахунок контрольних бітів.

```
for parity_position in [2 ** i for i in range(r)]:
    parity = 0

    for position in range(1, n + 1):
        if position & parity_position:
            parity ^= int(codeword[position])

    codeword[parity_position] = str(parity)
```

Для виявлення помилок у прийнятому кодовому слові реалізовано функцію hamming_syndrome(). Вона виконує повторний розрахунок контрольних бітів та формує синдром помилки.

```
def hamming_syndrome(codeword: str) -> int:
    syndrome = 0

    parity_position = 1

    while parity_position <= len(codeword):
        parity = 0

        for position in range(1, len(codeword) + 1):
            if position & parity_position:
                parity ^= int(indexed[position])
```

```
        if parity:
            syndrome += parity_position

        parity_position *= 2

    return syndrome
```

Отримане значення синдрому використовується для визначення позиції помилки. Якщо синдром дорівнює нулю, помилка відсутня. Якщо синдром має ненульове значення, відповідний біт вважається пошкодженим.

Процес виправлення помилки реалізовано у функції hamming_decode(). Вона виконує пошук помилкового біта, його інверсію та відновлення інформаційної послідовності.

```
def hamming_decode(codeword: str):
    corrected = list(codeword)

    syndrome = hamming_syndrome(codeword)

    if 1 <= syndrome <= len(codeword):
        corrected[syndrome - 1] = (
            "1" if corrected[syndrome - 1] == "0"
            else "0"
        )
```

Після виправлення пошкодженого біта програма видаляє контрольні розряди та повертає початкове інформаційне повідомлення.

Для моделювання помилок реалізовано функцію inject_error(). Вона дозволяє користувачу вибрати позицію біта, значення якого необхідно змінити.

```
def inject_error(codeword: str, position: int) -> str:
    modified = list(codeword)

    modified[position - 1] = (
```

```
            "1" if modified[position - 1] == "0"
            else "0"
        )

        return "".join(modified)
```

Окремий блок програми відповідає за кодування текстових повідомлень. Спочатку текст переводиться у двійковий формат за допомогою ASCII-кодів символів.

```
def text_to_binary(text: str) -> str:
    return "".join(
        format(ord(char), "08b")
        for char in text
    )
```

Після цього двійкова послідовність розбивається на блоки однакової довжини, які можуть бути незалежно закодовані методом Хемінга.

Для реалізації графічного інтерфейсу створено клас HammingApp. Саме він відповідає за формування головного вікна програми та організацію взаємодії користувача з алгоритмом.

```
class HammingApp:
    def __init__(self, root):
        self.root = root

        self.root.title(
            "Програма кодування методом Хемінга"
        )

        self.root.geometry("1120x780")
```

У головному вікні створено декілька вкладок, кожна з яких реалізує окремий режим роботи програми.

```
notebook = ttk.Notebook(self.root)
```

```
self.tab_encode = ttk.Frame(notebook)
self.tab_decode = ttk.Frame(notebook)
self.tab_text = ttk.Frame(notebook)
self.tab_tests = ttk.Frame(notebook)
```

Для відображення структури кодового слова використовується спеціальний графічний клас CodewordCanvas, який успадковується від елемента Canvas.

```
class CodewordCanvas(tk.Canvas):
    def draw(
        self,
        codeword,
        parity_positions,
        error_position
    ):
```

У межах цього класу реалізовано графічне відображення інформаційних та контрольних бітів, а також виділення позиції виявленої помилки.

Для візуалізації результатів тестування створено окремий клас StatsCanvas, який будує графічні діаграми на основі статистики роботи алгоритму.

```
class StatsCanvas(tk.Canvas):
    def draw(self, stats):
```

Під час тестування програма автоматично зчитує набори даних із текстових файлів, виконує їх кодування, моделює помилки та аналізує результати виправлення.

```
def run_error_test(self, relative_path):
```

За результатами виконання тестів формується статистика щодо кількості перевірених блоків, успішно виправлених помилок та випадків без помилок.

Висновки до розділу 6

У даному розділі було розглянуто особливості кодування інформації методом Хемінга та розроблено програмне забезпечення для реалізації даного алгоритму мовою Python.

Проведений аналіз показав, що код Хемінга належить до завадостійких кодів і дозволяє не лише виявляти, але й автоматично виправляти одиничні помилки, які можуть виникати під час передавання або зберігання інформації. Використання контрольних бітів забезпечує можливість визначення позиції пошкодженого біта та відновлення початкових даних без необхідності повторної передачі повідомлення.

На основі теоретичних положень було розроблено алгоритм роботи програми, який охоплює всі основні етапи обробки інформації: введення даних, визначення кількості контрольних бітів, формування кодового слова, моделювання помилки, обчислення синдрому, виправлення помилки та відновлення початкового повідомлення.

Для практичної реалізації алгоритму створено програмне забезпечення з графічним інтерфейсом користувача. Розроблена програма забезпечує кодування двійкових послідовностей, кодування текстових повідомлень, внесення штучних помилок, декодування даних, автоматичне виправлення одиничних помилок та тестування наборів даних із формуванням статистичних показників.

Особливістю створеного програмного забезпечення є наявність засобів візуалізації, які дозволяють відображати структуру кодового слова, контрольні біти, позицію помилки та результати тестування. Це підвищує наочність роботи алгоритму та спрощує його вивчення.

Проведене тестування підтвердило правильність реалізації алгоритму Хемінга та коректність роботи програмного забезпечення в усіх передбачених режимах. Програма успішно виконувала кодування інформації, виявляла та виправляла одиничні помилки, а також забезпечувала коректне відновлення початкових даних.

ВИСНОВОК

У кваліфікаційній роботі було розроблено комп’ютерну систему діагностики твердотільного накопичувача Kingston SSD A400. Метою роботи було створення концепції апаратно-програмного комплексу, призначеного для контролю технічного стану SSD-накопичувача, аналізу його експлуатаційних параметрів та своєчасного виявлення ознак можливих несправностей.

У першому розділі проведено дослідження сучасних твердотільних накопичувачів, розглянуто особливості їх конструкції та принципи функціонування. Виконано аналіз технічних характеристик накопичувача Kingston SSD A400, який було обрано як об’єкт діагностики. Також визначено основні параметри, що характеризують його технічний стан, зокрема температуру, залишковий ресурс пам’яті, швидкість роботи, кількість помилок передачі даних та інші показники, які можуть бути використані для оцінювання працездатності пристрою.

У другому розділі розроблено апаратну складову системи діагностики. На основі порівняльного аналізу сучасних мікроконтролерних платформ для реалізації системи було обрано ESP32-WROOM-32. Для відображення результатів діагностики обрано OLED-дисплей SSD1306, а для звукового інформування користувача — активний п’єзоелектричний динамік. Крім того, розроблено схему підключення всіх компонентів системи та визначено принцип їх взаємодії.

У третьому розділі виконано розробку програмної складової комп’ютерної системи діагностики. Обґрунтовано вибір мови програмування Python та середовища розробки Visual Studio Code. Розроблено алгоритм роботи системи, який забезпечує отримання діагностичних даних, їх аналіз, визначення технічного стану накопичувача та формування повідомлень для користувача.

На основі запропонованого алгоритму створено програмне забезпечення з графічним інтерфейсом користувача та реалізовано механізми візуалізації результатів діагностики.

Проведене тестування програмного забезпечення підтвердило правильність роботи розробленого алгоритму. У процесі тестування було змодельовано нормальний, попереджувальний та критичний стани накопичувача. Для кожного сценарію система коректно виконувала аналіз параметрів та формувала відповідні рекомендації щодо подальшої експлуатації SSD.

У четвертому розділі було виконано проєктування корпоративної комп’ютерної мережі відповідно до заданої топології. Розглянуто структуру мережі, особливості її побудови та організацію взаємодії між окремими сегментами мережевої інфраструктури. Отримані результати можуть бути використані під час створення інформаційної інфраструктури підприємства, у межах якої буде експлуатуватися система діагностики.

У п’ятому розділі виконано проєктування серверної кімнати для забезпечення функціонування інформаційної системи. Проведено вибір серверного та мережевого обладнання, розроблено план його розміщення у серверній шафі, а також виконано розрахунок тепловиділення обладнання та обґрунтовано вибір системи охолодження. Запропоновані технічні рішення забезпечують необхідний рівень надійності, масштабованості та безперервності роботи серверної інфраструктури.

У шостому розділі розроблено програму кодування інформації методом Хемінга. Реалізовано алгоритми формування контрольних бітів, кодування повідомлень, виявлення та виправлення одиничних помилок у переданих даних. Для демонстрації роботи алгоритму створено програмне забезпечення з графічним інтерфейсом користувача, що дозволяє виконувати кодування, декодування, моделювання помилок та візуалізацію результатів обробки інформації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мельник А.О. Архітектура комп’ютера. – Луцьк : Волинська обласна друкарня, 2023. – 470 с.

2. Морзе Н.В., Кузьмінська О.Г. Комп’ютерні мережі : навчальний посібник. – Київ : Університетська книга, 2022. – 384 с.

3. Stallings W. Computer Organization and Architecture. – 12th ed. – New York : Pearson, 2022. – 896 p.

4. Patterson D., Hennessy J. Computer Organization and Design RISC-V Edition. – Morgan Kaufmann, 2021. – 792 p.

5. Tanenbaum A.S., Wetherall D.J. Computer Networks. – 6th ed. – Pearson, 2021. – 960 p.

6. Barr M., Massa A. Programming Embedded Systems. – 2nd ed. – O’Reilly Media, 2021. – 336 p.

7. Yiu J. The Definitive Guide to ARM Cortex-M23 and Cortex-M33 Processors. – Newnes, 2023. – 624 p.

8. Monk S. Programming Arduino: Getting Started with Sketches. – 3rd ed. – McGraw-Hill Education, 2022. – 192 p.

9. Blum J. Exploring Arduino. – Wiley, 2021. – 528 p.

10. Evans M., Noble J., Hochenbaum J. Arduino in Action. – 3rd ed. – Manning Publications, 2022. – 520 p.

11. Margolis M. Arduino Cookbook. – 4th ed. – O’Reilly Media, 2023. – 810 p.

12. Upton E., Halfacree G. Raspberry Pi User Guide. – 5th ed. – Wiley, 2022. – 320 p.

13. Zeller C. ESP32 Programming for the Internet of Things. – Packt Publishing, 2023. – 406 p.

14.Dogan I. ESP32 Development using the Arduino IDE. – Elektor Publishing, 2022. – 290 p.

15.Monk S. Programming the ESP32 in MicroPython. – McGraw-Hill, 2023. – 224 p.

16.Kingston Technology. Kingston SSD A400 Product Specifications [Електронний ресурс]. – Режим доступу: https://www.kingston.com – дата звернення: 01.06.2026.

17.Kingston Technology. Kingston SSD A400 Datasheet [Електронний ресурс]. – Режим доступу: https://www.kingston.com/en/ssd/a400-solid-state-drive – дата звернення: 01.06.2026.

18.Serial ATA International Organization. Serial ATA Revision 3.5 Specification [Електронний ресурс]. – Режим доступу: https://sata-io.org – дата звернення: 01.06.2026.

19.NVM Express Organization. Storage Management Basics [Електронний ресурс]. – Режим доступу: https://nvmexpress.org – дата звернення: 01.06.2026.

20.Python Software Foundation. Python Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3 – дата звернення: 01.06.2026.

21.Arduino Documentation [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc – дата звернення: 01.06.2026.

22.Espressif Systems. ESP32-WROOM-32 Technical Reference Manual [Електронний ресурс]. – Режим доступу: https://www.espressif.com – дата звернення: 01.06.2026.

23.Espressif Systems. ESP32 Datasheet [Електронний ресурс]. – Режим доступу: https://www.espressif.com/en/support/documents/technical-documents – дата звернення: 01.06.2026.

24.Smartmontools Team. SMART Monitoring Tools Documentation [Електронний ресурс]. – Режим доступу: https://www.smartmontools.org – дата звернення: 01.06.2026.

25.

26.Li H., Xu J., Wang X. SSD Reliability Analysis Based on SMART Parameters // IEEE Access. – 2023. – Vol. 11. – P. 54721–54735.

27.Kim S., Park J. Predictive Failure Detection of Solid-State Drives Using Machine Learning // Sensors. – 2024. – Vol. 24, №4. – P. 2156–2170.

28.Ahmed M., Khan R. Performance Evaluation of SATA SSD Storage Systems // Journal of Computer Engineering. – 2023. – Vol. 15, №2. – P. 44–58.

29.Lin S., Costello D. Error Control Coding. – 3rd ed. – Pearson, 2022. – 1248 p.

30.Peterson W., Brown D. Cyclic Codes for Error Detection. – Springer, 2021. – 418 p.

31.Cisco Networking Academy. Introduction to Networks. – Cisco Press, 2023. – 720 p.

32.Cisco Systems. Routing and Switching Essentials. – Cisco Press, 2022. – 680 p.

33.Руденко В.Д., Кравченко О.П. Проектування локальних комп’ютерних мереж : навчальний посібник. – Київ : КПІ ім. Ігоря Сікорського, 2022. – 312 с.

34.ASHRAE. Thermal Guidelines for Data Processing Environments. – Atlanta : ASHRAE, 2022. – 220 p.

35.Hewlett Packard Enterprise. Data Center Design Guide [Електронний ресурс]. – Режим доступу: https://www.hpe.com – дата звернення: 01.06.2026.

ДОДАТОК А

Програмне забезпечення мікроконтролера ESP32 для системи діагностики Kingston SSD A400

```
from machine import Pin, I2C
import ssd1306
import time
import sys
import json

# Налаштування I2C для OLED-дисплея SSD1306
# SDA підключено до GPIO21
# SCL підключено до GPIO22
i2c = I2C(0, scl=Pin(22), sda=Pin(21), freq=400000)

# Ініціалізація дисплея 128x64
display = ssd1306.SSD1306_I2C(128, 64, i2c)

# Підключення звукового модуля до GPIO23
buzzer = Pin(23, Pin.OUT)
buzzer.value(0)


def show_message(line1="", line2="", line3="", line4=""):
    """
    Виведення інформації на OLED-дисплей.
    Дисплей має роздільну здатність 128x64,
    тому використовується до 4 рядків тексту.
    """
    display.fill(0)
```

```
    display.text(str(line1), 0, 0)
    display.text(str(line2), 0, 16)
    display.text(str(line3), 0, 32)
    display.text(str(line4), 0, 48)
    display.show()


def short_signal():
    """
    Короткий звуковий сигнал.
    Використовується при помилці приймання даних.
    """
    buzzer.value(1)
    time.sleep(0.2)
    buzzer.value(0)


def alarm():
    """
    Звукове оповіщення про критичний стан SSD.
    Формується три короткі сигнали.
    """
    for i in range(3):
        buzzer.value(1)
        time.sleep(0.2)
        buzzer.value(0)
        time.sleep(0.2)


def analyze_data(data):
```

```
"""
Аналіз діагностичних параметрів Kingston SSD A400.

Очікувані параметри:
temp   - температура накопичувача, °C;
health - залишковий ресурс, %;
errors - кількість помилок;
smart  - SMART-статус накопичувача.
"""

temp = data.get("temp", 0)
health = data.get("health", 0)
errors = data.get("errors", 0)
smart = data.get("smart", "OK")

status = "NORMAL"

# Критичний стан
if temp >= 60 or health <= 20 or errors > 0 or smart != "OK":
    status = "CRITICAL"
    alarm()

# Попереджувальний стан
elif temp >= 50 or health <= 40:
    status = "WARNING"
    buzzer.value(0)

# Нормальний стан
else:
    status = "NORMAL"
```

```
        buzzer.value(0)

    # Виведення результатів на дисплей
    show_message(
        "Kingston A400",
        "Temp: " + str(temp) + " C",
        "Health: " + str(health) + " %",
        status
    )


def wait_for_data():
    """
    Очікування та приймання даних від персонального комп'ютера.
    Дані надходять через USB-UART у форматі JSON.
    """

    while True:
        try:
            line = sys.stdin.readline()

            if line:
                data = json.loads(line)
                analyze_data(data)

        except Exception:
            show_message(
                "DATA ERROR",
                "Wrong format",
                "Check PC script"
```

```
            )
            short_signal()
            time.sleep(1)


# Стартове повідомлення
show_message(
    "SSD DIAG",
    "ESP32 READY",
    "Waiting data..."
)

time.sleep(1)

# Запуск основного циклу програми
wait_for_data()
```

ДОДАТОК Б

Основний програмний код програми кодування методом Хемінга

Б.1 Реалізація кодування методом Хемінга

```
def is_power_of_two(number: int) -> bool:
    return number > 0 and (number & (number - 1)) == 0

def required_parity_bits(data_length: int) -> int:
    r = 0

    while (2 ** r) < data_length + r + 1:
        r += 1

    return r

def hamming_encode(data_bits: str):

    m = len(data_bits)
    r = required_parity_bits(m)
    n = m + r

    codeword = ["0"] * (n + 1)

    data_index = 0

    for position in range(1, n + 1):

        if is_power_of_two(position):
            codeword[position] = "0"
        else:
```

```
            codeword[position] = data_bits[data_index]
            data_index += 1

    for parity_position in [2 ** i for i in range(r)]:

        parity = 0

        for position in range(1, n + 1):

            if position & parity_position:
                parity ^= int(codeword[position])

        codeword[parity_position] = str(parity)

    return "".join(codeword[1:])
```

Б.2 Реалізація визначення синдрому помилки

```
def hamming_syndrome(codeword: str):

    syndrome = 0

    indexed = ["0"] + list(codeword)

    parity_position = 1

    while parity_position <= len(codeword):

        parity = 0

        for position in range(1, len(codeword) + 1):
```

```
            if position & parity_position:
                parity ^= int(indexed[position])

        if parity:
            syndrome += parity_position

        parity_position *= 2

    return syndrome
```

Б.3 Реалізація декодування та виправлення помилок

```
def hamming_decode(codeword: str):

    corrected = list(codeword)

    syndrome = hamming_syndrome(codeword)

    if 1 <= syndrome <= len(codeword):

        corrected[syndrome - 1] = (
            "1"
            if corrected[syndrome - 1] == "0"
            else "0"
        )

    corrected_word = "".join(corrected)

    data_bits = []

    for position, bit in enumerate(corrected_word, start=1):
```

```
        if not is_power_of_two(position):
            data_bits.append(bit)

    return "".join(data_bits), corrected_word, syndrome
```

Б.4 Реалізація моделювання помилки

```
def inject_error(codeword: str, position: int):

    modified = list(codeword)

    modified[position - 1] = (
        "1"
        if modified[position - 1] == "0"
        else "0"
    )

    return "".join(modified)
```

Б.5 Реалізація перетворення тексту у двійковий код

```
def text_to_binary(text: str):

    return "".join(
        format(ord(char), "08b")
        for char in text
    )
```

Б.6 Фрагмент реалізації графічного інтерфейсу

```
class HammingApp:

    def __init__(self, root):

        self.root = root
```

```
        self.root.title(
            "Програма кодування методом Хемінга"
        )

        self.root.geometry("1120x780")

        self.create_widgets()

    def create_widgets(self):

        notebook = ttk.Notebook(self.root)

        self.tab_encode = ttk.Frame(notebook)
        self.tab_decode = ttk.Frame(notebook)
        self.tab_text = ttk.Frame(notebook)
        self.tab_tests = ttk.Frame(notebook)

        notebook.add(
            self.tab_encode,
            text="Кодування"
        )

        notebook.add(
            self.tab_decode,
            text="Декодування"
        )

        notebook.add(
            self.tab_text,
            text="Кодування тексту"
```

```
        )

        notebook.add(
            self.tab_tests,
            text="Тестування"
        )
```

Б.7 Головна функція запуску програми

```
if __name__ == "__main__":

    root = tk.Tk()

    app = HammingApp(root)

    root.mainloop()
```