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

Пояснювальна записка

до кваліфікаційної роботи
Бакалавр
(освітньо-кваліфікаційний рівень)

на тему *Розробка комп'ютерної системи діагностики ноутбуку Asus ROG Strix G17 2023*

Development of a computer diagnostic system for the Asus ROG Strix G17 2023 laptop

Виконав: студент 4 курсу, групи 4КСМ
спеціальності
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)

Оксененко П. П.
(прізвище та ініціали)

Керівник *Іванчук О.В.*
(прізвище та ініціали)

Рецензент *Вороненко М.О.*
(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 _____
(шифр і назва)

Спеціальність *123 "Комп'ютерна інженерія"*
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії _____
Комп'ютерних систем та мереж
_____ А.А. Григорова
«___» _____ 2026 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Оксененко Павлу Павловичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Розробка Розробка комп'ютерної системи діагностики ноутбуку Asus ROG Strix G17 2023*
Development of a computer diagnostic system for the Asus ROG Strix G17 2023 laptop

керівник проекту (роботи) *Іванчук О.В., доктор філософії*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «13» січня 2026 р. № 48-С
2. Строк подання студентом проекту (роботи) 28.05.2026
3. Вихідні дані до проекту (роботи) *Методичні рекомендації до виконання кваліфікаційної роботи бакалавра. Матеріали практики*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Використання джерел зарядної станції, Проектування та розробка пристрою діагностики, Розробка програмного забезпечення системи, Корпоративна комп'ютерна мережа, Моделювання серверного приміщення, Захист інформації в комп'ютерній мережі.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Пристрій діагностики. Схема електрична принципова; ЛКМ. Схема електрична Розташування. Програма пристрою діагностики. Програмне забезпечення вузла Алгоритм. ПЗ шифрування даних Алгоритм

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.	Вивчення об'єкта	06.02-26.02	*виконано*
2.	Огляд стану питання	01.03-19.03	*виконано*
3.	Розробка пристрою діагностики	20.03-10.04	*виконано*
4.	Розробка програмного забезпечення системи	11.04-20.04	*виконано*
5.	Проектування комп’ютерної мережі	21.04-28.04	*виконано*
6.	Оформлення ПЗ та креслень	15.05	*виконано*
7.			
8.			
9.			
10.			
11.			
12.			
13.			

Студент _____ Оксененко П.П.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Іванчук О.В.
(підпис) (прізвище та ініціали)

ВІДОМІСТЬ ОБСЯГУ КВАЛІФІКАЦІЙНОЇ РОБОТИ

№ п/п	Формат	Позначення	Найменування	Кількість	Шифр док-та	Примітки
1.	А4	123.22КСМ004	Реферат	1	РФ	
2.	А4	123.22КСМ004	Пояснювальна записка	81	КРПЗ	
3.	А2	123.22КСМ004	Пристрій діагностики. Схема електрична принципова	1	Е3	
4.	А1	123.22КСМ004	ЛКМ. Структурна схема мережі	1	Е7	
5.	А2	123.22КСМ004	Програма пристрою діагностики. Алгоритм	1	АГ1	
6.	А2	123.22КСМ004	Програма вузла комп’ютерної мережі. Алгоритм	1	АГ2	
7.	А2	123.22КСМ004	ПЗ шифрування даних. Алгоритм	1	АГ3	
8.						
9.						
10.						

					ХНТУ 123.21КСМ027			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Оксененко П.П.			ВІДОМІСТЬ ПРОЕКТУ	Літ.	Арк.	Аркушів
Перев.		Іванчук О.В.					1	1
Реценз.								
Н. контр.		Іванчук О.В.						
Затверд.		Григорова А.А.						

РЕФЕРАТ

Дипломний проект містить: 81 сторінок, 19 рисунків, 28 таблиць, 33 посилань, 4 додатки.

Об'єкт дослідження – ноутбук Asus ROG Strix G17 2023.

Ціль проекту – розробка комп’ютерної системи діагностики ноутбуку Asus ROG Strix G17 2023.

В дипломному проекті було розроблено комп’ютерну систему діагностики ноутбуку Asus ROG Strix G17 2023. Було спроектовано електричну-принципову схему для пристрою діагностики. Розроблене програмне забезпечення для пристрою діагностики.

Корпоративна мережа має 8 підмереж, для яких визначено обладнання та IP-адреси. Мережа перевірена на відповідність стандарту Ethernet за допомогою розрахунків та моделювання. Мережа використовую шифрування даних за допомогою трьох видів шифрування: симетричного алгоритму AES, блокового шифру Ascon та криптографічного алгоритму RSA.

НОУТБУК, КОМП’ЮТЕРНА СИСТЕМА ДІАГНОСТИКИ,
КОРПОРАТИВНА МЕРЕЖА, ДІАГНОСТИКА.

АНОТАЦІЯ

В дипломному проекті було виконано аналіз для діагностики ноутбуку. Було розроблено комп’ютерна система діагностики для ноутбуку Asus ROG Strix G17 2023 – спроектована принципово-електричну схему пристрою, програмне забезпечення пристрою. Комп’ютерна система діагностування є частиною спроектованої корпоративної мережі. Для мережі визначено список необхідного обладнання, IP-адреса та перевірено на відповідність стандарту Ethernet за допомогою розрахунків та моделювання. До мережі додано захист за допомогою 3 методів шифрування: симетричного алгоритму AES, блокового шифру Ascon та криптографічного алгоритму RSA.

					ХНТУ 123.22КСМ004			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Оксененко П.П.			АНОТАЦІЯ	Літ.	Арк.	Аркушів
Перев.		Іванчук О.В.					1	1
Реценз.								
Н. контр.		Іванчук О.В.						
Затверд.		Григорова А.А.						

SUMMERY

In the diploma project, an analysis was performed for diagnosing a laptop. A computer diagnostic system was developed for the Asus ROG Strix G17 2023 laptop - a schematic diagram of the device, device software were designed. The computer diagnostic system is part of the designed corporate network. A list of necessary equipment, an IP address were determined for the network and checked for compliance with the Ethernet standard using calculations and modeling. Protection was added to the network using 3 encryption methods: the symmetric AES algorithm, the Ascon block cipher and the RSA cryptographic algorithm.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК

ПК	–	Персональний комп'ютер
LPC	–	Low Pin Count
ISA	–	Industry Standard Architecture
ОС	–	Операційна система
USB	–	Universal Serial Bus
RAM	–	Random Access Memory

ВСТУП

Діагностика ноутбука – це оцінка технічного стану пристрою. Виявлення причини некоректної роботи або поломок, які істотно впливають на працездатність ноутбука.

Зазвичай це комплекс заходів, які потрібно провести, щоб визначити несправність. Якісна та точна діагностика – це запорука успіху подальшого ремонту. Під час проведення діагностики проводиться програмне та апаратне тестування пристрою.

Діагностика обладнання включає проведення серії автоматичних та ручних тестів, що виконуються програмним забезпеченням для оцінки стану, ефективності роботи та продуктивності різних апаратних компонентів і периферійних пристроїв.

1 ОПИС ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Що таке діагностика ноутбуку

Діагностика ноутбука – невід'ємна частина професійного ремонту, що дозволяє швидко та точно виявити причини несправностей. Процес включає кілька етапів:

1. Зовнішній огляд – огляд корпусу ноутбука, клавіатуру, дисплей та роз'єм на наявність видимих пошкоджень, таких як тріщини, сліди залиття або сильні забруднення. Цей етап дозволяє виявити очевидні проблеми, які можуть перешкоджати нормальній роботі пристрою.
2. Тестування системи охолодження – перевірка стану системи охолодження, розбір та чищення ноутбука від пилу, при необхідності заміна термопасти.
3. Перевірка роботи компонентів – діагностика включає тестування процесора, оперативної пам'яті, відеокарти та жорсткого диска за допомогою спеціального програмного забезпечення.
4. Тестування програмного забезпечення – причина проблем може бути не в комплектуючих ноутбука, а в програмних збоях. Проведення перевірки операційної системи, виявляє шкідливі програми та помили, які могли спричинити нестабільну роботу ноутбука.

На ноутбуки Asus є інструмент MyAsus System Diagnostics, який оцінює стан обладнання ПК та оптимізує продуктивність. Він пропонує перевірку всієї системи, індивідуальні тести обладнання та усунення несправностей на основі сценаріїв для вирішення таких проблем, як повільне завантаження, зниження заряду батареї або перебої з Wi-Fi.

Існують декілька сценаріїв, які найчастіше зустрічаються з пристроями ПК, а також загальну діагностику обладнання. Кожен сценарій супроводжується коротким описом, що дозволяє швидко визначитися та

вибрати необхідну перевірку. В таблиці 1.1 було наведено опис можливих сценаріїв.

Таблиця 1.1 – Опис сценаріїв діагностики.

Сценарій	Опис	Елементи, що слід перевірити
Діагностика в один клік	Можна вибрати цю опцію, щоб запустити повну перевірку системи.	Адаптер, Пам'ять, Wi-Fi, Bluetooth, Жорсткий диск, Твердотільний диск, Акумулятор, Вентилятор, Перевірка системи.
Індивідуальна діагностика	Щоб мати повне уявлення про стан апаратного забезпечення ПК, рекомендується запустити перевірку апаратного забезпечення.	Адаптер, Пам'ять, Wi-Fi, Bluetooth, Жорсткий диск, Твердотільний диск, Акумулятор, Вентилятор, Перевірка системи.
Помилка синього екрану	Екран смерті, як правило, пов’язаний із обладнанням чи драйвером. Часто демонструють код STOP, який можна використати, щоб з’ясувати першопричину синього екрану.	Пам'ять, твердотільний диск, перевірка секторів, накопичувач пристроїв.
Повільна система	Недостатня пам’ять і місце на жорсткому диску – загальні причини уповільнення роботи системи.	Пам'ять, твердотільний диск, простір на твердотільному диску, використання додатків, вентилятор.

Проблема підключення вайфай	Стійкість мережевих операторів та пристроїв Wi-Fi Інтернету є однією з найпоширеніших причин зривів мережі.	Бездротовий пристрій та налаштування, Мережне підключення
Помилка драйверу	Встановлення неправильного драйвера може спричинити нестабільну роботу системи та її збій.	Драйвери
Система не відповідає	Випадкові збої зазвичай виникають внаслідок налаштувань Windows, збоїв обладнання або пошкодження шкідливого програмного забезпечення.	Жорсткий диск, перевірка секторів, розподіл файлів
Проблема з батареєю	Перевірка джерела живлення та всіх знімних шнурів живлення. Акумулятор правильно розміщений у відсіку.	Батарея
Повільний час завантаження	Поганий жорсткий диск або запуск занадто багато програм у фоновому режимі може вплинути на час завантаження та збільшити час завантаження.	Пам'ять, жорсткий диск, перевірка часу завантаження, перевірка планувальника завдань

1.2 Шина LPC

Шина LPC – комп’ютерна шина, що використовується на персональних комп’ютерах для підключення пристроїв з низькою пропускною здатністю до процесора.

Шина LPC була представлена компанією Intel в 1998 році як програмно-сумісна заміна шини промислової стандартної архітектури (ISA). Основною перевагою шини LPC є невелика кількість необхідних для роботи сигналів: для роботи шини потрібно лише сім сигналів, що спрощує розведення і без того напханих провідниками сучасних материнських плат . Використання шини LPC дозволяє відмовитися від розведення від 30 до 72 провідників, які довелося б розвести при використанні шини ISA.

Специфікація на шину LPC визначає сім обов'язкових сигналів, необхідні забезпечення двосторонньої передачі: чотири з цих сигналів використовуються передачі як адресної інформації, так передачі даних; три використовуються для управління – сигнали frame, сигнал скидання і тактовий сигнал.

Пропускна спроможність шини LPC залежить від режиму обміну. Виділяють окремі режими обміну для роботи з пристроями вводу-виведення, пристроями типу «пам'ять», сеансами DMA та іншими. Однак в будь-якому випадку пропускна здатність шини LPC вища, ніж шини ISA під час роботи в аналогічному режимі.

Електричні характеристики шини LPC походять від характеристик звичайної шини PCI. Вона має те саме обмеження, що для перемикання будь-якого сигналу шини, щоб інший пристрій розмовляв, потрібно два цикли простою. У першому циклі шина активно перебуває у високому стані. У другому циклі шина не перебуває у стані високого рівня. У другому циклі шина не перебуває у стані високого рівня за допомогою резисторів. Новий пристрій може почати надсилати дані по шині на третьому циклі. Операції LPC витрачають значну частину свого часу на виконання таких перемикань.

Специфікація LPC вимагає сім сигналів:

1. Чотири послідовні сигнали LAD[3:0] для передачі мультиплексних даних, включаючи тип циклу, напрямок циклу, вибір мікросхеми, адресу, дані та час очікування.

2. Один тактовий сигнал LCLK 33 МГц, що надається хостом.

3. Один LFRAME# для позначення початку або зупинки транзакції.

4. Один LRESET# для скидання шини.

Для передачі переривань та керування живленням можна додатково додати шість сигналів бічної смуги; якщо їх використовувати, то загалом буде тринадцять сигналів. Ці сигнали включають:

1. LDRQ# для закодованого запиту DMA/ведучого шини.

2. ERIRQ для периферійних пристроїв, що потребують підтримки переривань.

3. Тактовий сигнал CLKRUN# служить для периферійних пристроїв, яким потрібен DMA або керування шиною в системі, яка може зупинити шину PCI.

4. LPME# для периферійних пристроїв, які хочуть запитувати пробудження зі стану низького енергоспоживання.

5. LPCPD# для вимкнення живлення.

6. LSMI# для випадків, коли периферійний пристрій хоче викликати SMI# для інструкції вводу/виводу для повторної спроби.

Після налаштування інтерфейс LPC зазвичай підключається через головний пристрій до PCI або головної шини. До інтерфейсу LPC підключаються периферійні пристрої нижче за течією, включаючи компоненти вводу/виводу, флеш-пам'ять та інші вбудовані контролери.

Інтерфейс LPC також має кілька обмежень, зокрема:

1. Вбудовано до 13 контактів (7 обов'язкових та 6 додаткових сигналів бічної смуги), що збільшує вартість контактів.

2. Потреба технології сигналізації вводу-виводу 3,3 В, що збільшує витрати.

3. Нижня пропускна здатність та тактова частота шини 33 МГц.

1.3 POST-карта

POST-карта – плата розширення, що має власний цифровий індикатор для виведення на нього кодів ініціалізації системної плати. За останнім виведеним кодом можна визначити, який з компонентів системної плати є несправним.

Ці коди залежать від виробника BIOS материнської плати. У разі відсутності помилок і нормального проходження тесту POST видає на свій індикатор значення, що не змінюється протягом роботи комп'ютера.

Тест POST – перевірка апаратного забезпечення комп'ютера, що виконується при його включенні. Виконується програмами, що входять до BIOS материнської плати.

2 РОЗРОБКА ПРИСТРОЮ МОНІТОРИНГУ БЛОКА ЖИВЛЕННЯ

2.1 Структурна схема та опис елементів пристрою

В якості пристрою для діагностики ноутбуку було прийнято рішення використати мікроконтролер Raspberry Pi Pico для основи проектування. Мікроконтролер служить ядром пристрою та приймає всі дані. Модуль SLB9660 відповідає за інтерфейс LPC. За допомогою цього модулю з ноутбуку дістається необхідна інформація та передається в мікроконтролер Raspberry Pi Pico. Інформацію було виведено на LCD-дисплей 1602 через інтерфейс I2C. щоб передати всю інформацію на сервер, треба з’єднання до ПК через USB-кабель.

На рисунку 2.1 була зображена структурна схема пристрою для діагностики ноутбуку.

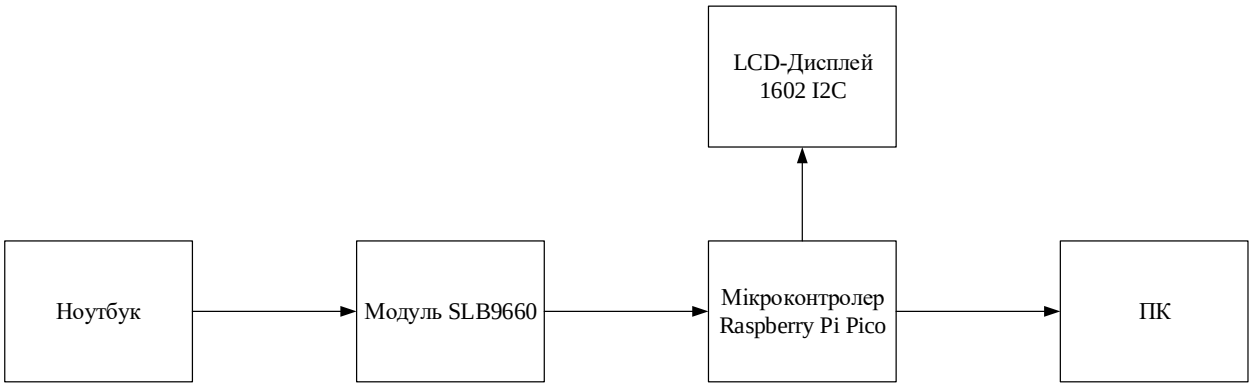


Рисунок 2.1 – Структурна схема пристрою

2.2 Апаратна частина реалізації пристрою

2.2.1 Плата Raspberry Pi Pico

Raspberry Pi Pico – Мікроконтролер на базі 32-бітного високопродуктивного чіпа RP2040, який має два ядра ARM Cortex-M0+ із максимальною тактовою частотою 133 МГц. На рисунку 2.2 зображено сам мікроконтролер Raspberry Pi Pico.

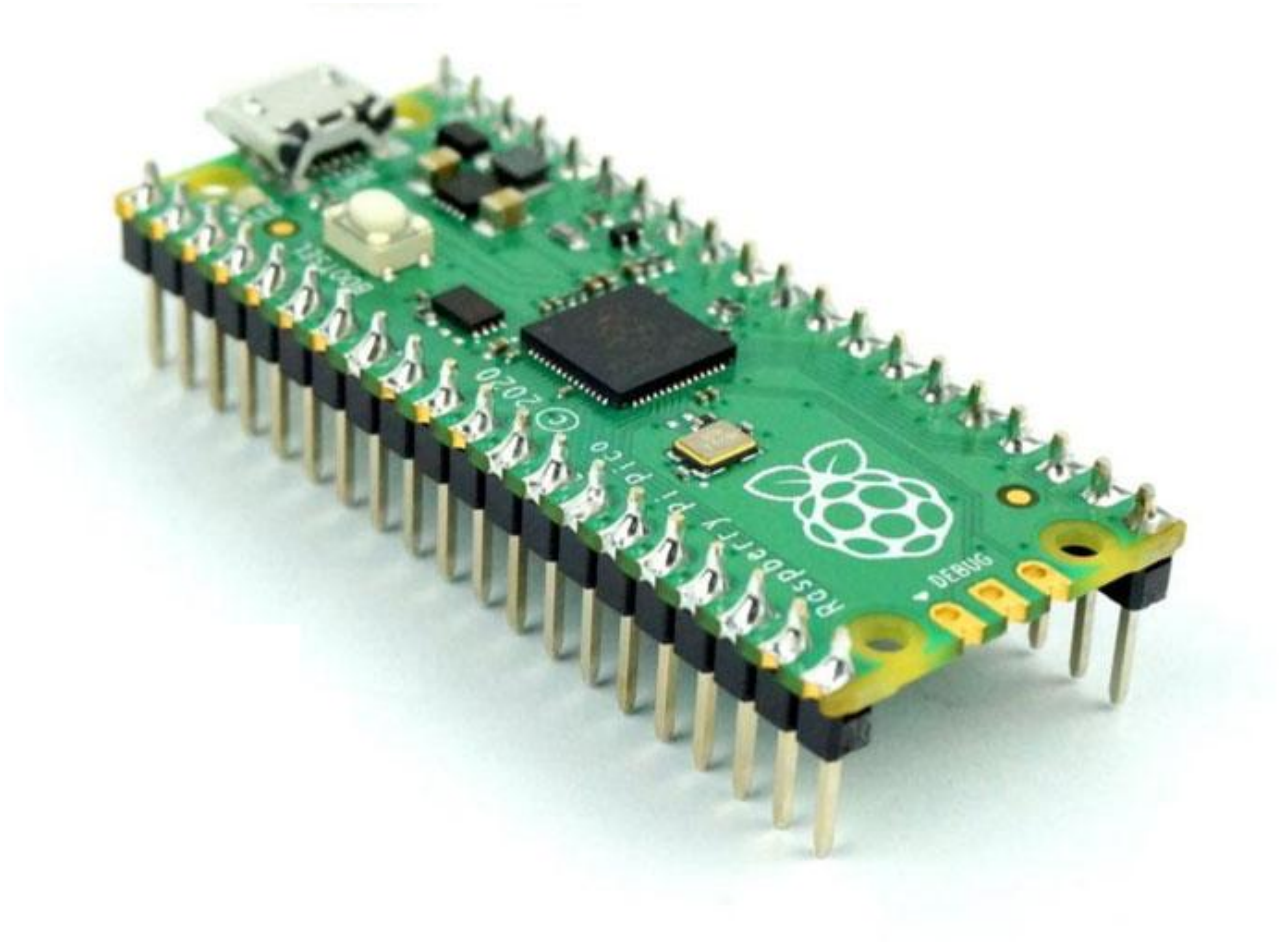


Рисунок 2.2 – Мікроконтролер Raspberry Pi Pico

RP2040 виробляється за сучасною 40-нанометровою технологією, забезпечуючи високу продуктивність, низьке динамічне енергоспоживання і малий витік, а також різні режими низького енергоспоживання для підтримки тривалої роботи від батареї. Його основними перевагами, крім продуктивності, є досить малі габарити, що полегшує його використання в мікросистемах і бюджетна вартість.

Завдяки великій вбудованої пам'яті, симетричному двоядерному процесору, детермінованою структурою шин і багатому набору периферійних пристроїв, доповнених унікальною підсистемою програмованого введення-виведення (PIO), RP2040 забезпечує професійних користувачів неперевершеною потужністю та гнучкістю. В таблиці 2.1 описано характеристика Raspberry Pi Pico.

Таблиця 2.1 – Характеристика Raspberry Pi Pico.

Мікроконтролер	RP2040
Робоча напруга	3.3В
Вхідна напруга	1.8-5.5В
Цифрові входи/виходи	24
Аналогові входи	3
Постійний струм через вхід/вихід	12мА
Флеш-пам'ять	2МБ
RAM	264КБ
Тактова частота	133МГц
Розміри	53ммх21мм

2.2.2 Модуль SLB9660

Модуль SLB9660 – модуль платформи, що базується на технології апаратної безпеки. Вона підтримує інтерфейс LPC, а переривання передаються за допомогою протоколу послідовних переривань (SERIRQ). Модуль зображено на рисунку 2.3.

Рисунок 2.3 – модуль SLB9660

Інтерфейс не підтримує стандартні цикли вводу-виводу (читання та запис). Це означає, що адресація пристрою за допомогою відображення вводу-виводу неможлива. В таблиці 2.2 описані назви та функції контактів модулю SLB9660.

Таблиця 2.2 – Функції

Назва	Функція
LRESET#	Зовнішній сигнал скидання. Активація цього контакту безумовно скидає пристрій. Сигнал має низький рівень активного сигналу та зазвичай підключений до сигналу хоста.
LCLK	Вхід тактової частоти. Цей контакт забезпечує зовнішній тактовий сигнал для мікросхеми та зазвичай підключений до тактової частоти PCI хоста.
LFRAME#	Сигнал кадрування LPC. Цей контакт підключено до сигналу LPC та вказує на початок нового циклу на шині LPC або завершення перерваного циклу. Сигнал має активний низький рівень.
LAD0	Біт адреси/даних LPC. Мультиплексована шина команд, адреси та даних LPC.
LAD1	
LAD2	
LAD3	

2.2.3 LCD-дисплей 1602 I2C

Модуль екрана, який керується через протокол I2C та доступний для таких платформ, як Raspberry Pi Pico. У порівнянні з традиційними інтерфейсами, I2C значно спрощує підключення та економить контакти GPIO,

що робить його ідеальним для швидкого проектування прототипів. LCD дисплей зображено на рисунку 2.4.

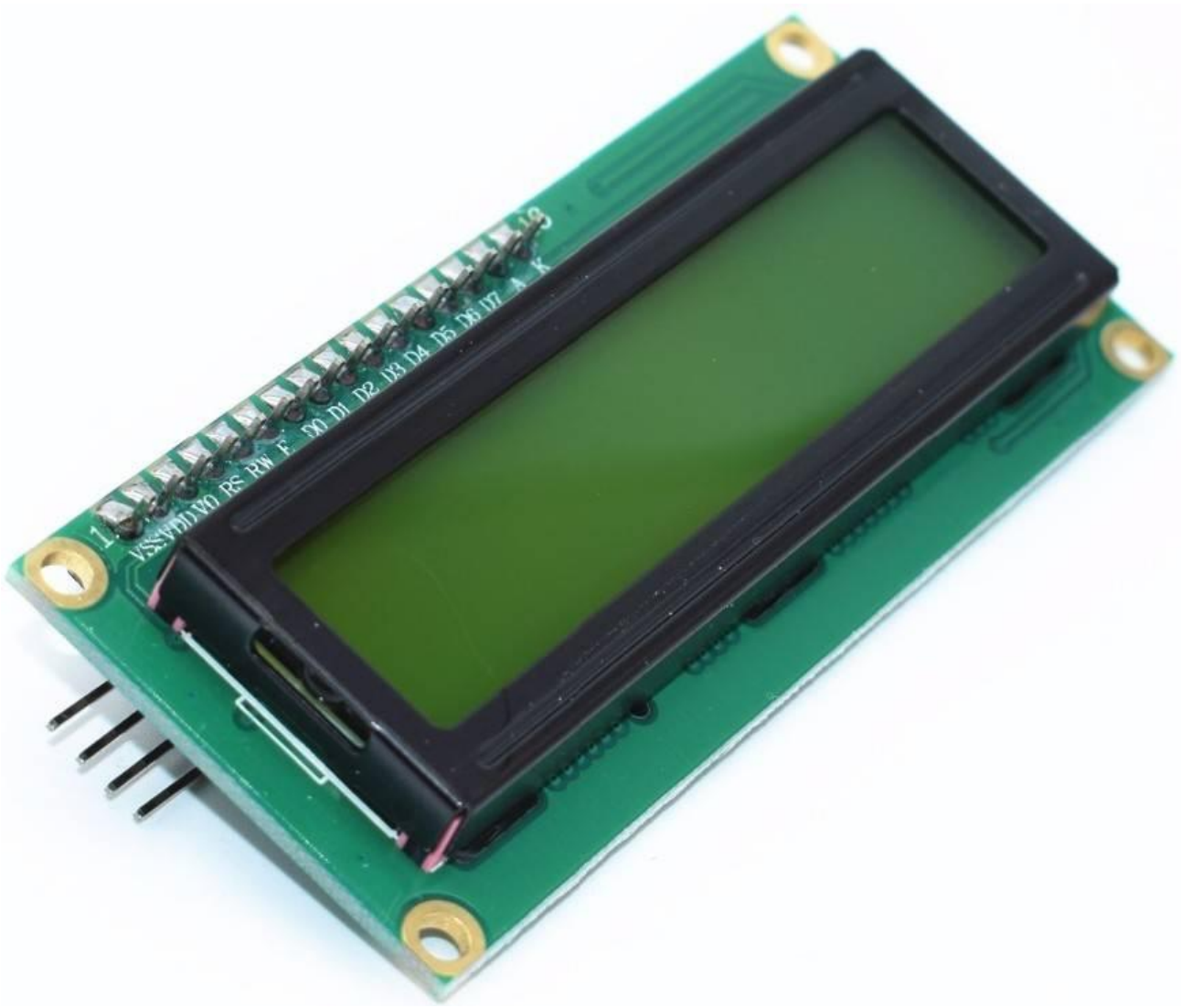


Рисунок 2.4 – LCD дисплей 1602

Основні особливості дисплею:

1. Може відображати до 32 символів (має два рядки по 16 символів в кожному), підтримуючи прокручування екрану та переміщення курсору.

2. Вишуканий зовнішній вигляд, тонка робота, високий рівень інтеграції, стабільна продуктивність.

3. Вбудований РК-драйвер AiP31068L та мікросхема драйвера підсвічування SN3193.

Послідовний інтерфейс I2C для LCD 1602 – відмінний інтерфейсний модуль на мікросхемі PCF8574T, який зображено на рисунку 2.5. Потрібна для розширення кількості портів вводу/виводу для міні-комп’ютерів Raspberry Pi Може використовуватись як інтерфейсна плата для підключення LCD 1602. Для налаштування контрастності дисплея встановлено змінний резистор. Характеристика LCD дисплею описано в таблиці 2.3.

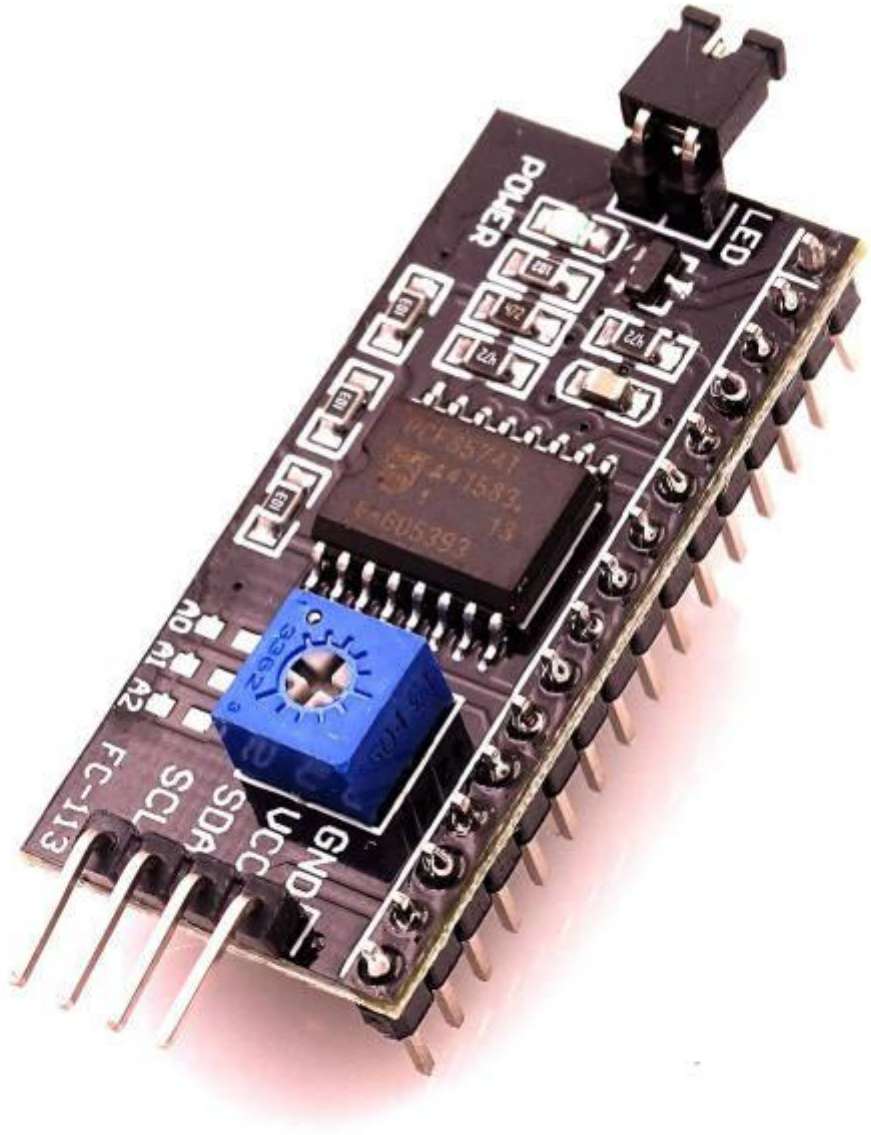


Рисунок 2.5 – Модуль інтерфейсу I2C

Таблиця 2.3 – Характеристика LCD дисплею.

Розміри	80мм*36мм
Робоча температура	0-50° C
Підсвічування	Зелений
Колір символів	Чорний
Розмір символу	4.35мм*2.95мм
Формат	16*2
Інтерфейс	HD44780
Видима область екрану	64.5мм*13.8мм
Напруга живлення	5В

2.3 Принципова схема пристрою

Пристрій складається з трьох компонентів: мікроконтролеру Raspberry Pi Pico, модулю SLB9660, та LCD-дисплею 1602 I2C.

Модуль SLB9660 включає в себе 28 контактів, серед яких було використано 7 потрібних під поточну задачу. Контакти 17, 20, 23 та 26 відповідають за адреси та дані LPC. Активація контакту 16 скидає пристрій. Контакт 21 забезпечує зовнішній тактовий сигнал для мікросхеми. Контакт 22 підключено до сигналу LPC та вказує на початок нового циклу на шині LPC. Контакт 18 призначений для GND (землі). Всі 7 контактів будуть підключені до мікроконтролера Raspberry Pi Pico напряму. Схема модулю SLB9660 зображено на рисунку 2.6.

LRESET#	16
LAD3	17
GND	18
LAD2	20
LCLK	21
LFRAME#	22
LAD1	23
LAD0	26

Рисунок 2.6 – Модуль SLB9660

Мікроконтролер Raspberry Pi Pico має 40 контактів (пінів), серед яких було використано 11. Контакти з 1 по 7 з'єднані напряму з модулем SLB9660, та приймають всю необхідну інформацію з інтерфейсу LPC для обробки даних. Контакти 11 та 12 відправляють дані через інтерфейс I2C на LCD-дисплей. Контакт 11 відповідає за лінію даних SDA, а контакт – за лінію тактування SCL. На контакт 38 подається живлення. Контакт 40 призначений для землі (GND). Схема мікроконтролеру зображена на рисунку 2.7.

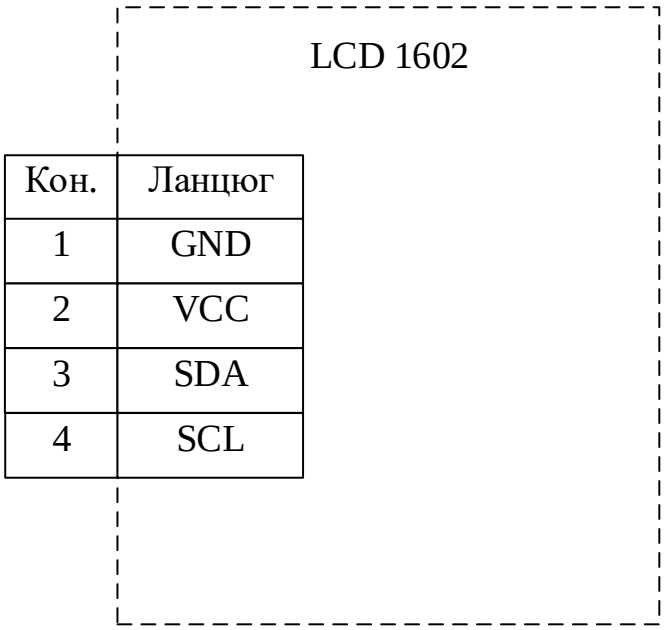
Raspberry Pi Pico

Кон.	Ланцюг
1	GP0
2	GP1
3	GND
4	GP2
5	GP3
6	GP4
7	GP5

Ланцюг	Кон.
GP8	11
GP9	12
GND	38
VCC	40

Рисунок 2.7 – Мікроконтролер Raspberry Pi Pico

LCD-дисплей має 16 контактів. Під час підключення LCD-дисплею використовується модуль шини I2C, який дуже сильно спрощує підключення і водночас використовує тільки 4 піна для під’єднання. Контакт 1 та 2 використовуються для землі (GND) та живлення (VCC). Контакт 3 та 4 використовуються для передачі даних через SDA та SCL. Схема дисплею зображена на рисунку 2.8.



Кон.	Ланцюг
1	GND
2	VCC
3	SDA
4	SCL

Рисунок 2.8 – LCD-дисплей 1602 I2C

Повністю електрична принципова схема відображена на кресленні ХНТУ 123.22КСМ004.Е3.

3 ПРОГРАМУВАННЯ ПРИСТРОЮ МОНІТОРИНГУ

Програма пристрою з діагностики ноутбуку за допомогою POST-кодів була написана на мові програмуванні С під плату Raspberry Pi Pico.

Програма складається з 7 бібліотек, яка складається з стандартних бібліотек C/C++, а також бібліотеки рівня С для доступу до всього обладнання мікроконтролера серії Raspberry Pi.

1. Бібліотека «stdio.h» – стандартна бібліотека мови С для вводу/виводу, та використовується в кожному С-проєкті.
2. Бібліотека «pico/stdlib.h» підключає базові функції для роботи з Raspberry Pi.
3. Бібліотека «hardware/pio.h» – бібліотека, яка контролює роботу з PIO (програмований ввід/вивід) на Raspberry Pi.
4. За допомогою бібліотеки «hardware/i2c.h» налаштовується робота з шиною I2C, через яку передається інформація на LCD-дисплей.
5. Файл «lcd16x2_i2c.h» для роботи з LCD-дисплеєм через I2C-адаптер на Raspberry Pi.
6. Файл «lpc_bus_sniffer.pio.h» автоматично генерується із програми для Raspberry Pi Pico SDK. Використовується для читання сигналів через PIO.
7. Бібліотека «hardware/clocks.h» потрібна для керування тактовими генераторами (clock system) мікроконтролера Raspberry Pi.

Конструкція typedef union __attribute__() дозволяє інтерпретувати 32 біта як набір полів. Така конструкція використовується для подання кадру шини LPC (Low Pin Count), де з 32-бітного слова витягуються адреса та керуючи сигнали.

Функція gpio_initialization() налаштовує мікроконтролер Raspberry Pi Pico для певної апаратної ролі. Вона включає чотири основні кроки:

1. Увімкнення тактового сигналу – мікроконтролер подає живлення на конкретні порти GPIO, перш ніж отримати доступ до його регістрів.

2. Встановлення напрямку – визначення чи є контакт входом (зчитування датчиків) чи виходом (керування світлодіодами).

3. Налаштування резисторів – встановлення внутрішніх резисторів або залишення їх в плаваючому стані для забезпечення стану контакту в режимі очікування.

4. Встановлення початкового стану – для виходів налаштування роботи з високого чи низького рівня під час завантаження.

Функція init_lpc_bus_sniffer() ініціалізує PIO-програму для перехоплення даних транзакцій на LPC-шині на мікроконтролері Raspberry Pi та повертає номер state mashine, який використовується. Програма декодує POST-коди та виводить їх на LCD-дисплей.

Функція int main() служить відправною точкою виконання програми. Вона зазвичай керує виконанням програми, викликаючи інші функції. Всередині функції main() виконує наступні дії:

1. Налаштування GPIO.
2. Запуск сніферу для LPC-шини через PIO.
3. Ініціалізація LCD-дисплею 1602 I2C.
4. Вивод на дисплей системної частоти.
5. Постійне зчитування даних з PIO та вивод на LCD-дисплей.

Повний код програми, яка потребує для роботи пристрою, наведено у додатку А.

4 РОЗРОБКА ПРОГРАМИ ШИФРУВАННЯ

Пристрій для діагностики пов'язаний із комп’ютером. На комп’ютері програма починає процес одного із трьох видів шифрування. Після цього дані відправляються з комп’ютера на сервер по корпоративній комп’ютерній мережі.

Шифрування – метод перетворення даних, щоб їх могли читати лише авторизовані особи. У процесі шифрування відкритий текст перетворюється в зашифрований за допомогою криптографічного ключа, приклад схеми зображено на рисунку 4.1 Шифрування використовується для захисту даних від крадіжки.

Рисунок 4.1 – Приклад шифрування

Існують два основні види шифрування: симетричне шифрування та асиметричне шифрування. Асиметричне шифрування також відоме як шифрування з відкритим ключем.

В симетричному шифруванні існує лише один ключ, та всі сторони, що взаємодіють, використовують один і той самий ключ як для шифрування, так і для дешифрування. Схема зображена на рисунку 4.2.

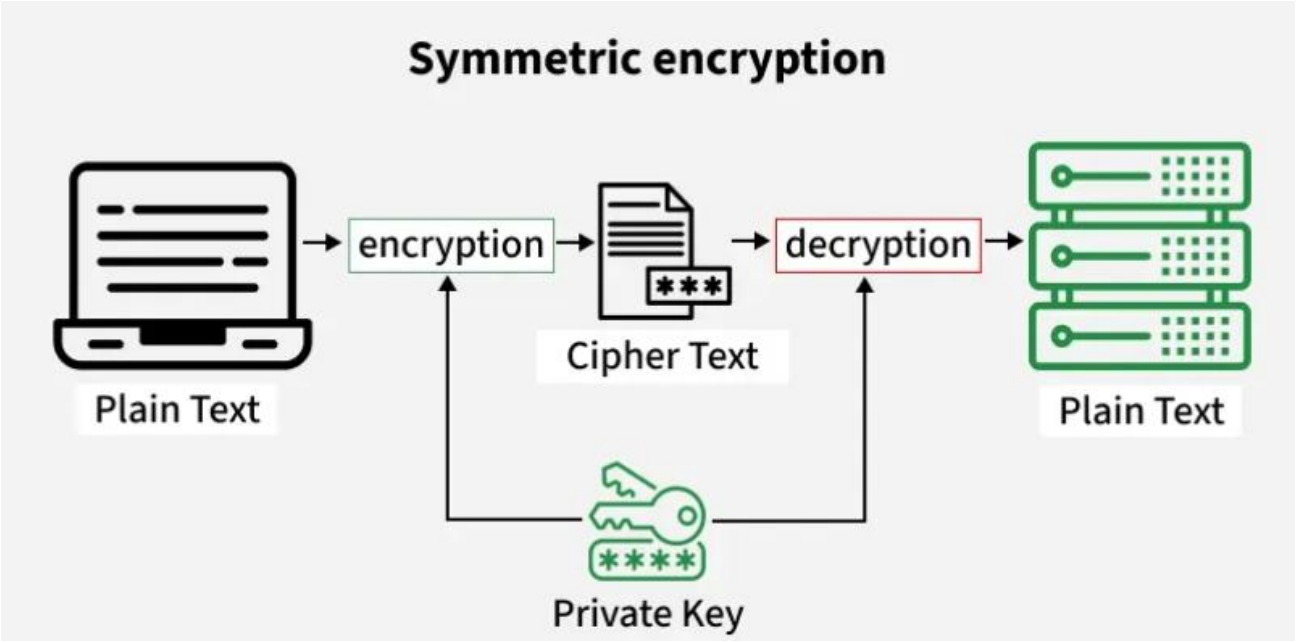


Рисунок 4.2 – Симетричне шифрування

В асиметричному шифруванні є два ключі: один ключ використовується для шифрування, а інший для дешифрування, як зображено на рисунку 4.3. Ключ дешифрування зберігається в таємниці, тоді як ключ шифрування є публічним для використання будь-ким. Асиметричне шифрування є базовою технологією для TLS (Transport Layer Security).

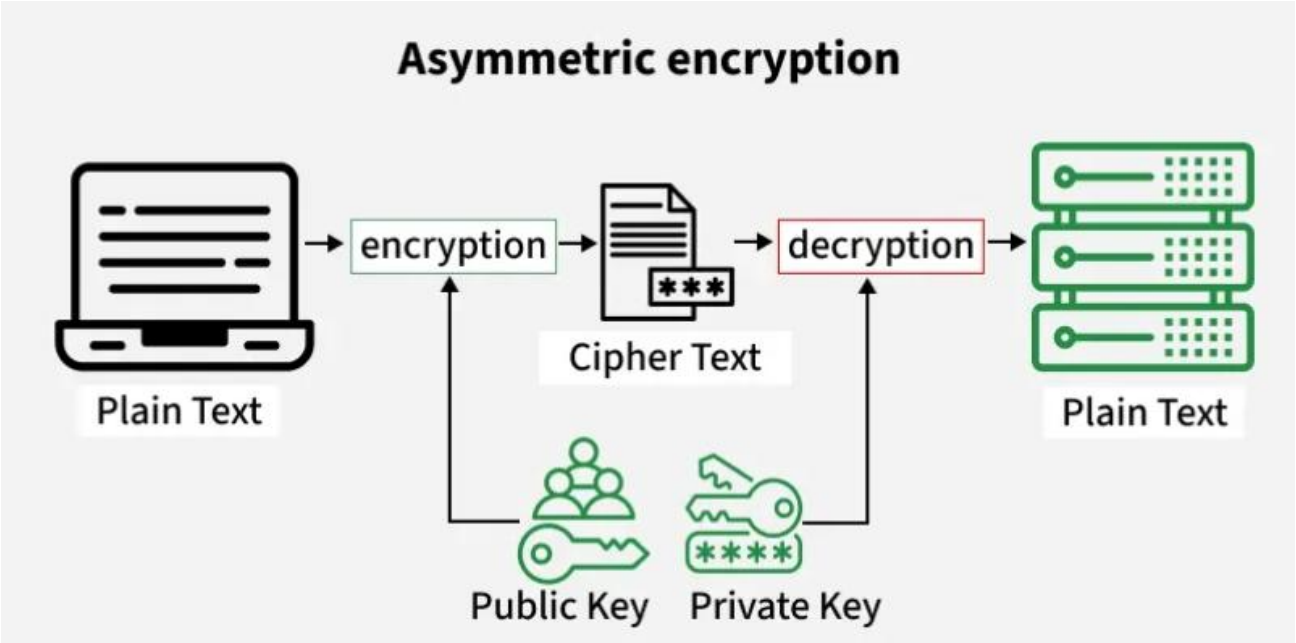


Рисунок 4.3 – Асиметричне шифрування

Більшість типів шифрування, з якими стикаються в інтернеті, насправді є гібридним шифруванням:

1. Клієнт та сервер використовують асиметричне шифрування для автентифікації та узгодження спільних симетричних ключів.

2. Після завершення рукостискання вони перемикаються на швидке симетричне шифрування для решти сеансу.

3. Протоколи, такі як протоколи безпеки транспортного рівня, потім використовують ці ключі як для шифрування, так і для розшифрування даних, якими вони обмінюються.

Криптографічний ключ – набір відомих математичних величин, погоджених як відправником, так і одержувачам. Як і фізичний ключ, він блокує (шифрує) дані таким чином, що лише той, хто має правильний ключ, може їх розблокувати (розшифрувати). Також є кілька важливих моментів:

1. Довші ключі значно ускладнюють атаки методом перебору, оскільки існує набагато більше можливих комбінацій.

2. Ключі шифрування повинні бути згенеровані з високою якістю випадковості, щоб зловмисники не могли їх вгадати або відтворити.

3. У сучасних системах шифрування ключі рідко вводяться людьми – вони створюються та зберігаються програмним забезпеченням.

Основні причини використання шифрування:

1. Безпека – шифрування допомагає запобігти витоку даних. В разі втрати або викрадення корпоративного пристрою із зашифрованим вмістом дані залишаться захищеними.

2. Конфіденційність – лише власник та одержувач можуть читати дані. Шифрування запобігає перехопленню конфіденційних даних зловмисниками, інтернет-провайдерами або навіть урядами.

3. Цілісність даних – шифрування також запобігає зловмисній діяльності, такі як перехопленню інформації під час передачі, оскільки зашифровані дані не можна переглянути або змінити під час передачі.

4. Нормативно-правові акти – багато галузевих та державних нормативних документів вимагають від компаній шифрувати дані користувачів.

Шифрування призначене для приховування від неавторизованих осіб даних, отриманих незаконним способом. Але в деяких ситуаціях шифрування може зробити дані недоступними для самого власника. Керування ключами – непросте завдання для підприємств, тому що вони повинні десь зберігатись, а зловмисники дуже добре вміють їх знаходити. Адміністратори повинні мати план захисту системи керування ключами.

Шифрування є основою для різноманітних технологій, але воно особливо важливе для забезпечення безпеки HTTP-запитів та відповідей. Протокол, що відповідає за це, називається HTTPS.

4.1 Шифрування AES

Шифрування AES – високонадійний алгоритм шифрування, який використовується для захисту даних шляхом перетворення їх у нечитабельний формат без належного ключа. Він широко використовується сьогодні, оскільки є набагато надійнішим за DES за потрійний DES, незважаючи на те, що його складніше реалізувати.

Шифрування AES використовує ключі різної довжини (128, 192 або 256 біт) для забезпечення надійного захисту від несанкціонованого доступу. Цей захід безпеки даних є ефективним та широко застосовується для захисту інтернет-зв’язку, захисту конфіденційних даних та шифрування файлів.

Сьогодні шифрування AES вважається золотим стандартом шифрування інформації в різних галузях, від державних установ до фінансових установ та технологічних компаній.

Шифрування AES перетворює відкритий текст на зашифрований за допомогою серії чітко визначених операцій, що виконуються протягом кількох раундів. Нижче описані ключові кроки:

1. Розширення ключа – початковий ключ шифрування до серії раундів ключів за допомогою алгоритму розкладу ключів. Ці раундові ключі використовуються на кожному етапі шифрування.

2. Початковий раунд (AddRoundKey) – дані відкритого тексту об'єднуються з ключем першого раунду за допомогою побітової операції XOR, змішуючи ключовий матеріал з даними.

3. Підбайти (підстановка байтів) – кожен байт у блоці даних замінюється відповідним байтом із заздалегідь визначеного поля підстановки, що вводить нелінійність у шифр.

4. Зсув рядків (ShiftRows) – рядки матриці даних циклічно зсуваються вліво, що допомагає розподілити значення байтів по блоці та збільшує дифузію.

5. Змішування стовпців (MixColumns) – кожен стовпець матриці даних трансформується за допомогою математичних операцій для подальшого перемішування даних та покращення розсіювання.

6. Змішування ключів (AddRoundKey) – ще один раундовий ключ поєднується з даними за допомогою XOR, тісно пов'язуючи процес шифрування із секретним ключем.

7. Фінальний раунд – пропускається крок MixColumns та завершується шифрування за допомогою SubBytes, ShiftRows та AddRoundKey, що створює остаточний зашифрований текст.

Кількість раундів (10, 12 або 14) залежить від довжини ключа, яка відповідно становить 128, 192 або 256 бітів:

1. AES-128 – цей варіант використовує 128-бітний ключ, та відомий балансом між швидкістю та безпекою. Він забезпечує надійний захист для загальних потреб безпеки даних, включаючи безпечний обмін файлами та базовий захист даних у програмах, де важлива висока швидкість.

2. AES-192 – ця версія пропонує вищий рівень безпеки, ніж AES-128. Хоча вона трохи повільніша, AES-192 часто використовується в галузях, які

потребують сильнішого шифрування, але не бажають додаткових обчислювальних ресурсів, як AES-256. Вона підходить для безпечного зв'язку в урядових установах.

3. AES-256 – найбезпечніша версія AES, використовує 256-бітний ключ і практично невразлива до атак методом перебору за допомогою сучасних технологій. Хоча це найбільш ресурсоємний варіант, йому надають перевагу в додатках, що вимагають максимальної безпеки, таких як фінансові транзакції, хмарне сховище та резервне копіювання даних. AES-256 широко використовується в секторах, що потребують безпеки найвищого рівня, таких як охорона здоров'я та фінансові послуги.

Визначальні особливості характеристики шифрування AES:

1. Шифрування із симетричним ключем – працює за алгоритмом симетричного ключа, тобто один і той самий ключ використовується як для шифрування, так і для дешифрування. Це спрощує процес шифрування та робить його швидшим, особливо корисним для захисту великих обсягів даних.

2. Різні розміри ключів – підтримує розміри ключів 128, 192 та 256 бітів. Ці опції забезпечують гнучкість для різних потреб безпеки, дозволяючи користувачам вибирати довжину ключа на основі бажаного балансу між швидкістю та безпекою.

3. Метод блочного шифрування – використовує підхід блочного шифрування, розділяючи дані на блоки (зазвичай 128 біт) та шифруючи кожен блок окремо. Така структура підвищує безпеку, гарантуючи, що кожен блок даних є незалежно захищеним.

4. Мережа підстановки-перестановки – алгоритм включає раунди підстановки та перестановки, перетворюючи відкритий текст на зашифрований у складний спосіб. Така конструкція забезпечує високий рівень безпеки шляхом ретельного змішування даних, що робить їх дуже стійкими до несанкціонованого доступу.

5. Ефективна продуктивність – його оптимізовано як для апаратного, так і для програмного забезпечення, забезпечуючи високу швидкість шифрування та дешифрування. Така ефективність дозволяє AES захищати дані без суттєвого впливу на продуктивність, що ідеально підходить для програм, які працюють в режимі реального часу.

6. Стійкість до відомих атак – він розроблений для забезпечення стійкості до відомих криптографічних атак, включаючи атаки методом перебору, диференціальний та лінійний криптоаналіз. Ця стійкість робить його придатним для середовищ з високим рівнем безпеки.

Шифрування AES широко використовується в різних секторах для забезпечення безпеки даних та конфіденційності:

1. Бездротовий захист (Wi-Fi) – зазвичай використовується в протоколах безпеки Wi-Fi, для шифрування даних, що передаються через бездротові мережі. Це гарантує, що конфіденційна інформація, така як паролі та особисті дані, залишається захищеною від несанкціонованого доступу.

2. Зашифрований перегляд (HTTPS) – веб-сайти використовують шифрування AES у протоколах HTTPS для захисту даних, що передаються між браузерами та серверами. Це шифрування допомагає захистити інформацію користувача, таку як облікові дані для входу та платіжні дані, від перехоплення зловмисниками.

3. Мобільні додатки – Багато мобільних додатків, що стосуються фінансових транзакцій або персональних даних, використовують шифрування AES для захисту даних на пристроях та під час передачі. Це включає банківські додатки, платформи соціальних мереж та додатки для обміну повідомленнями, що забезпечує користувачам спокій та впевненість у захисті їхніх даних.

4. Хмарне сховище – шифрування є важливим для захисту файлів, що зберігаються в хмарних середовищах. Такі сервіси, використовують AES, щоб забезпечити конфіденційність завантажених файлів та захист від несанкціонованого доступу.

5. Шифрування файлів та дисків – Операційні системи, пропонують варіанти шифрування на основі AES для захисту цілих жорстких дисків або окремих файлів. Це особливо корисно для захисту особистої або конфіденційної бізнес-інформації, що зберігається на фізичних пристроях.

6. Безпечний обмін повідомленнями – багато програм для зашифрованого обміну повідомленнями, використовують AES для повного захисту повідомлень, гарантуючи, що лише відправник та одержувач можуть прочитати вміст своїх розмов.

7. Військовий зв'язок – високий рівень безпеки та стійкість до атак роблять його придатним для захисту секретної та конфіденційної інформації.

4.2 Шифрування ASCON

Шифрування Ascon – сімейство алгоритмів автентифікованого шифрування та хешування, розроблених для того, щоб бути легкими та простими у впровадженні, навіть з додатковими контрзаходами проти атак побічних каналів.

Особливості шифрування Ascon полягають в наступних пунктах:

1. Автентифіковане шифрування, хешування та автентифікація повідомлень з однією легкою перестановкою з використанням конструкцій типу «sponge/duplex».

2. Ефективне програмне та апаратне забезпечення, особливо для пристроїв з обмеженими можливостями.

3. Легко та безпечно впроваджується, з ефективними засобами протидії побічним каналам та стійкістю до неправильного використання.

Усі члени сімейства Ascon використовують однакову полегшену перестановку. Ця перестановка ітеративно застосовує раундове перетворення на основі SPN до 16 раундів. Раундове перетворення складається з наступних трьох кроків:

1. Додавання констант раунду – XOR, специфічна для раунду 1-байтова константа до слова S2.

2. Нелінійний шар заміщення – застосовує 5-бітовий S-блок 64 рази паралельно з розрізом бітів (вертикально, поперек слів).

3. Лінійний дифузійний шар – XOR різних повернутих копій кожного слова (по горизонталі, в межах кожного слова).

Ascon-AEAD128 використовує дуплексний режим роботи для автентифікованого шифрування. Довжина ключа, тега та одноразового коду становить 128 бітів. Губка працює зі станом 320 бітів, з блоками повідомлень 128 бітів. Процес шифрування поділяється на чотири фази:

1. Ініціалізація – Ініціалізація стану ключем K та одноразовим числом N.

2. Обробка пов’язаних даних – Оновлення стану за допомогою пов’язаних блоків даних A.

3. Обробка відкритого тексту – Введення блоків відкритого тексту P у стан, та витяг блоків зашифрованого тексту C.

4. Фіналізація – Знову введення ключа K та витяг тегу T для автентифікації.

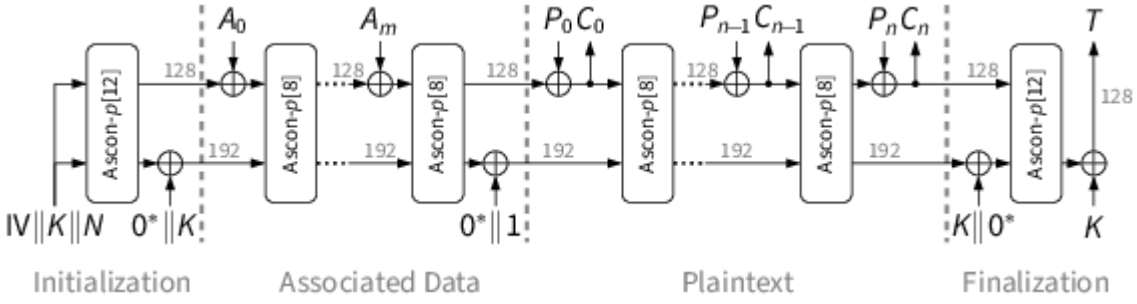


Рисунок 4.4 – Шифрування Ascon

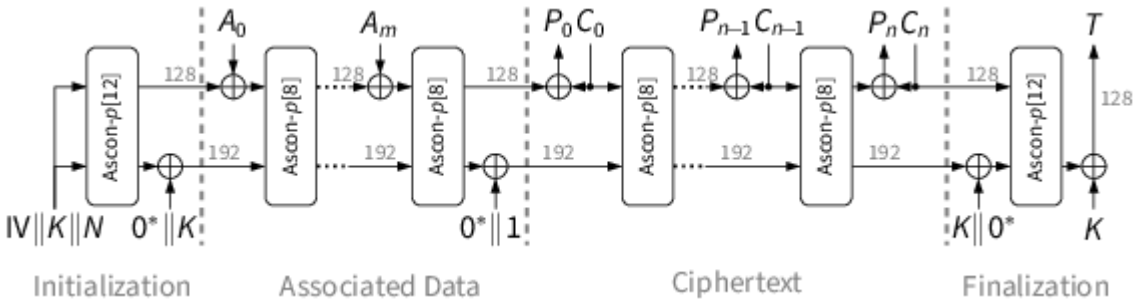


Рисунок 4.5 – Дешифрування Ascon

4.3 Шифрування RSA

Шифрування RSA – асиметричний алгоритм криптографії з відкритим ключем, що означає, що він працює з двома різними ключами: відкритий ключ для шифрування та закритий ключ для дешифрування.

Математика, що лежить в основі алгоритму шифрування RSA, включає одностороннюю функцію – її легко обчислити, але складно скасувати. Це робить RSA безпечним алгоритмом шифрування, який практично неможливо зламати методом повного перебору.

Шифрування RSA було названо на честь прізвищ творців: Рівеста, Шаміра та Адлемана, які описали алгоритм в 1977 році, працюючи в Массачусетському технологічному інституті.

Шифрування RSA використовується для забезпечення автентичності інтернет-повідомлень. Воно застосовується в цифрових підписах, протоколах безпечного зв’язку (SSH, HTTPS), шифруванні електронних листів, віртуальних приватних мережах та захисті програмного забезпечення.

Шифрування RSA широко вважається надійним та ефективним алгоритмом шифрування. Деякі переваги шифрування включають:

1. Автентифікація – надає відкриті та закриті ключі для шифрування повідомлень та автентифікації користувачів.

2. Швидкість – шифрування RSA швидше порівняно з іншими методами шифрування, такими як DSA.

3. Зберігає цілісність даних – дані не можна змінити під час їх передачі від одного користувача до іншого.

4. Ключі не надаються спільно – шифрування не вимагає спільного використання ключів, що забезпечує додаткову безпеку.

Шифрування досягається шляхом генерації пари ключів RSA. Сила алгоритму полягає в складності розкладанні на множники великих цілих чисел. Нижче наведено алгоритм RSA:

1. Вибір двох великих простих чисел – Процес починається з генерації двох великих простих числа випадковим чином для пари ключів, яку важко факторизувати. Ці числа зберігаються в таємниці. Вони множаться разом, щоб створити третє число, яке використовується як модуль для шифрування та дешифрування.

2. Генерація відкритих та закритих ключів – Відкритий ключ створюється шляхом вибору цілого числа, відомого як «e». Важливою вимогою до цього цілого числа «e» є те, що воно не повинно бути дільником (p-1) або (q-1), де p та q – прості числа, вибрані на першому кроці. Вибір «e» є критичним, оскільки воно використовується в процесі шифрування. Пара чисел (e, n), де n – це добуток простих чисел p та q, становить відкритий ключ. Цей відкритий ключ є загальнодоступним, щоб будь-хто міг використовувати його для шифрування повідомлення.

3. Шифрування – Відправник шифрує своє повідомлення, підносячи його до степеня e (з відкритого ключа), а потім взявши модуль n (також з відкритого ключа). В результаті отримується зашифрований текст.

4. Дешифрування – Одержувач розшифровує зашифрований текст, підносячи його до степеня d та знову обчислюючи модуль n. Отримане число потім ділиться на добуток простих чисел, а залишок є вихідним числовим повідомленням. Це число потім перетворюється назад у вихідний формат відкритого тексту, фактично завершуючи процес розшифрування та дозволяючи одержувачу прочитати повідомлення.

Шифрування RSA знаходить своє застосування в численних сферах та випадках використання:

1. VPN-з’єднання – Використовується у віртуальних приватних мережах (VPN) для безпечної передачі даних через ненадійні мережі. Зазвичай він використовується під час початкового процесу обміну ключами.

2. Безпечна передача файлів – В системах протоколу передачі файлів, таких як протокол передачі файлів SSH, RSA допомагає в безпечному обміні файлами. Це гарантує, що файли, що передаються між системами, залишаються конфіденційними.

3. Електронна пошта – Часто використовується в системах безпечної електронної пошти. Він допомагає зашифрувати вміст електронного листа, щоб його міг розшифрувати та прочитати лише передбачуваний одержувач із правильним ключем.

4. SSL/TLC для веб-безпеки – RSA є критично важливою частиною протоколу SSL/TLS, який використовується для захисту зв'язку в Інтернеті. Він гарантує, що дані, що передаються мережами, такі як інформація про кредитні картки або особисті дані, є безпечними та захищеними від хакерів.

Цифрові підписи – Використовується для створення цифрових підписів у різному програмному забезпеченні, що гарантує автентичність та цілісність даних. Це доводить, що дані походять від конкретного відправника та не були змінені під час передачі.

4.5 Програма шифрування

Програма написана на мові програмування C++. На початку програми оголошено кілька бібліотек для розширення функціоналу.

Бібліотека #include <commdlg.h> містить оголошення функцій, структур та констант для стандартних діалогових вікон Windows (відкриття та збереження файлу).

Бібліотека #include <windows.h> розширює функціонал необхідний для взаємодії програми з операційною системою Windows. Після підключення стають доступними такі функції, як створення вікон, робота з файлами, діалогові вікна.

Бібліотека #include <fstream> підключає стандартну бібліотеку C++ для роботи із файлами. Вона надає класи потоків для читання та запису даних у файли.

Бібліотека <vector> підключає контейнер std::vector. Це динамічний масив, який автоматично збільшує або зменшує свій розмір у міру додавання та видалення елементів.

Програма складається з 9 функцій. Функція readFile() зчитує вхідний файл. За допомогою класу ifstream читаються рядки з вхідного файлу. Якщо після перевірки is_open() видає помилку, то програма повідомляє про невдалу спробу відкрити файл.

Функція writeFile() записує результат в указаний файл. За допомогою класу ofstream йде запис в файл. Також є перевірка при операції, якщо сталася помилка при записі, програма повідомить про невдалу спробу запису.

Функція fakeAES() виконує симетричний алгоритм шифрування AES. Ця функція приймає текст і ключ, та повертає масив даних res.

Функція fakeASCON_encrypt() виконує шифрування за методом ASCON. Функція приймає 2 аргументи: текст, який треба зашифрувати, та ключ, за допомогою якого треба цей текст шифрувати.

Функція fakeASCON_decrypt() виконує дешифрування за методом ASCON. Функція приймає 2 аргументи: зашифрований текст, та ключ, який дасть можливість розшифрувати.

Функція fakeRSA_encrypt() виконує шифрування RSA. Ця функція також приймає 2 аргументи у вигляді тексту та ключа. Далі послідовно відбувається шифрування з кожним елементом масиву.

Функція fakeRSA_decrypt() виконує дешифрування RSA. За допомогою циклу for() кожний зашифрований елемент через ключ key розшифровується, та записується в масив res.

Функція chooseFile() налаштовує діалог вибору файлу, а саме вікно із вибором файлу в операційній системі. Після виклику функції шлях до вибраного файлу буде записаний в змінній filename.

Основна функція програми main() відповідає за вибір алгоритму шифрування, вибору файлу та виконання шифрування або дешифрування. Вибір файлу відбувається за допомогою стандартного вікна вибору файлу Windows. Також присутні перевірки на вміст файлу.

Алгоритм програми шифрування зображено на рисунку 4.6.

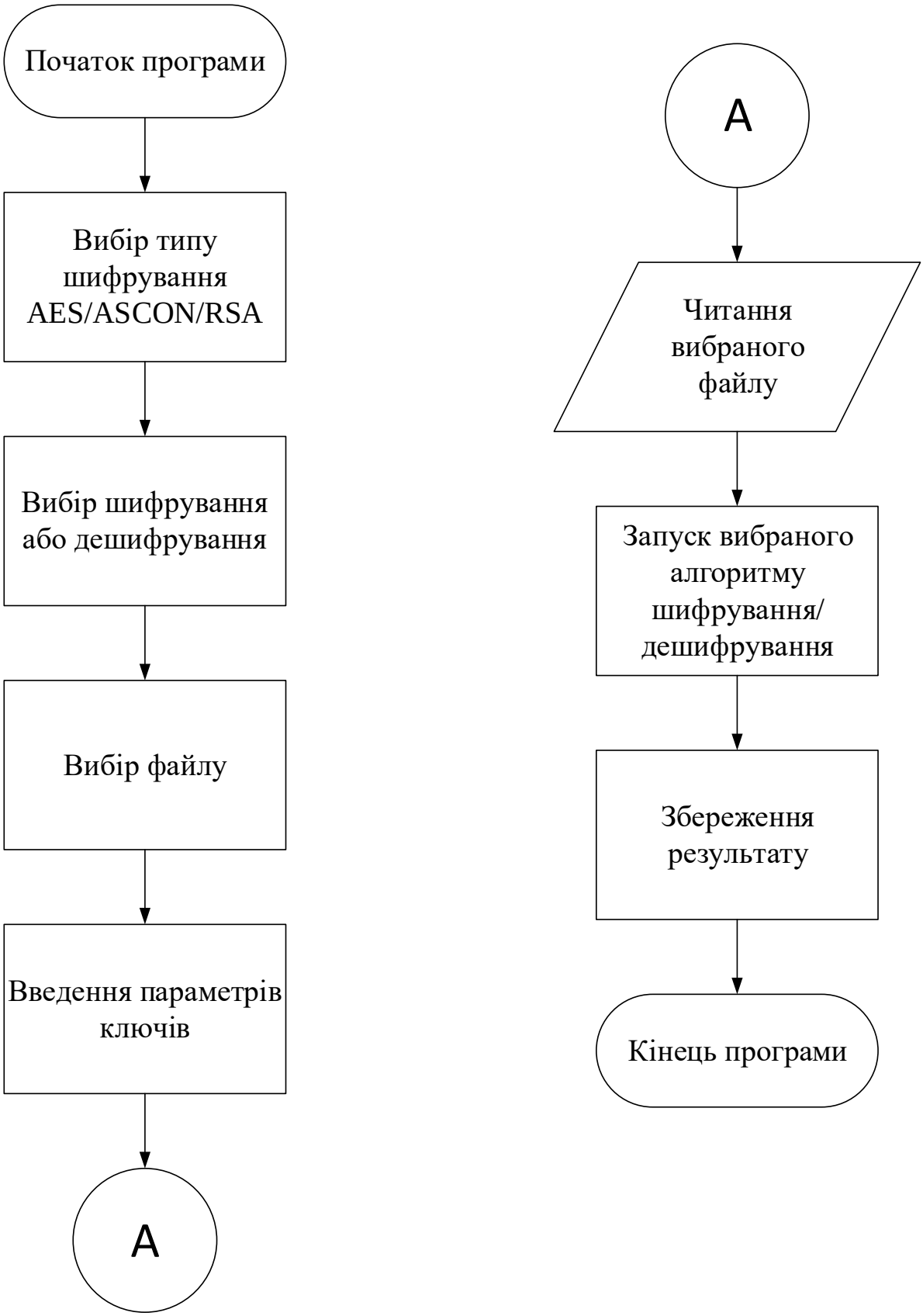


Рисунок 4.6 – Алгоритм програми шифрування

5 ПРОЕКТУВАННЯ ТА РОЗРОБКА КОМП'ЮТЕРНОЇ МЕРЕЖІ

5.1 Вхідні дані

Вхідні дані для побудови комп'ютерної мережі обираються за порядковим номером. Згідно з варіантом було обрано гілки комп'ютерної мережі та спосіб підключення до мережі Інтернет. Відповідні дані варіанту були вказані в таблиці 5.1.

Таблиця 5.1 – Вхідні дані.

Варіант	Номера гілок "дерева" які входять до корпоративної КМ								Інтеграція в Internet
									Маршрутизатор
1	1	2	4	7	8	10	11	12	C

Для кожної локальної мережі зазначені такі параметри: кількість робочих станцій N (PC) та проектна протяжність L. Довжина залежить від двох параметрів: де l – середня довжина кабелю між концентратором та робочими станціями в мережах, та максимальна допустима довжина кабелю в локальних мережах. Дані параметри вказані в таблиці 5.2, в яких застосовані такі скорочення стандартів:

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

BF – Стандарт IEEE 802.3, специфікація 100Base-FL;

B2 – Стандарт IEEE 802.3, специфікація 100Base-2;

B5 – Стандарт IEEE 802.3, специфікація 100Base-5;

TR – Стандарт IEEE 802.5.

Таблиця 5.2 – Параметри корпоративної мережі ЛКМ

№	Параметри ЛКМ	Номер гілки графу (ЛКМ)							
		1	2	4	7	8	10	11	12
12	Стандарт	B5	BT+B2	B2	-	BF	TR	BT	-
	Кількість PC, N	60-n	20+n	30+n	50-n	7+(n/2)	20+n	5+n	40-n
	Протяжність L	Nl/60	0.7l	Nl/30	Nl/50	0.9l	0.9l	0.7l	0.8l

IP-адреси проектованої комп'ютерної мережі, IP-адреси вузла маршрутизатора Інтернет, та IP-адреса зв'язаного з ним вузла маршрутизатора комп'ютерної мережі, які вказані в таблиці 5.3.

Таблиця 5.3 – Параметри ЛКМ корпоративної КМ

Номер варіанту	IP-адреси корпоративної КМ	IP-адреси вузлів маршрутизаторів Internet та КМ	Internet – адреси деяких вузлів ЛКМ
1	191.110.P.X	195.240.50.9 195.240.50.10	08:48:20:A1:90:01 08:48:88:54:67:97 08:48:OF:77:89:09 08:48:A5:A4:B1:2B

Для передачі даних між корпоративною комп'ютерною мережею та мережею Інтернет використовується оптоволоконний зв'язок.

5.2 Конфігурація комп'ютерної мережі

В корпоративній комп'ютерній мережі об'єднані 8 ЛКМ. Конфігурація мережі відповідає «дереву». Граф мережі зображено на рисунку 5.1. Гілки графу є робочі станції, що з'єднані за допомогою маршрутизаторів A, B, C.

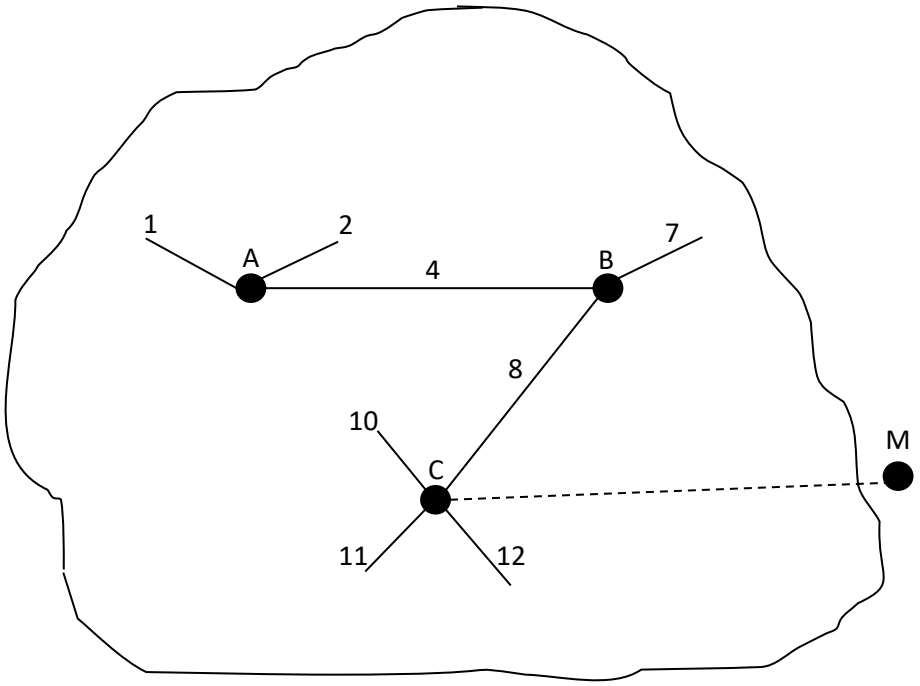


Рисунок 5.1 – Загальний граф комп’ютерної мережі

Маршрутизатор С використовується для підключення корпоративної комп’ютерної мережі до глобальної мережі Інтернет. Він підключення до головного маршрутизатора М через оптоволоконний зв'язок.

5.3 Будова локальної комп’ютерної мережі

Для вибору стандарту було використано таблицю 5.4, в якій наведено порівняльну характеристику стандартів. Для ЛКМ 2, 7, 12 обрано стандарт 10Base-T через низьку завантаженість на швидкість, низьку ціну та велику кількість підключень.

Таблиця 5.4 – Порівняльна характеристика стандартів.

	100Base-2	100Base-5	10Base-T	100Base-F
Низький вплив завантаженості на швидкість передачі	-	-	+	+

Низька ціна	+	+	+	-
Максимальна кількість комп’ютерів	30	100	1024	1024
Максимальна довжина кабелю	185	500	100	2000

Для кожної мережі розраховано кількість за умови, що n = 10.

Гілка 1: 60 – 10 = 50 робочих станцій.

Гілка 2: 20 + 10 = 30 робочих станцій.

Гілка 4: 30 + 10 = 40 робочих станцій.

Гілка 7: 50 – 10 = 40 робочих станцій.

Гілка 8: 7 + (10/2) = 12 робочих станцій.

Гілка 10: 20 + 10 = 30 робочих станцій.

Гілка 11: 5 + 10 = 15 робочих станцій.

Гілка 12: 40 – 10 = 30 робочих станцій.

Згідно із завданням протяжність гілок мережі залежить від 2 показників, де l – середня довжина кабелю між концентратором та ПК в мережах, де стандарт 10Base-T та Token Ring, та максимально допустима довжина кабелю в мережах, де стандарт 10Base-2 і 10Base-5.

Довжина інтерфейсного кабелю AUI складає (30+n)% від максимальної довжини, де n – номер варіанту. За умови, що n = 10, було розраховано значення, що в метрах дорівнює 20.

Використовуючи значення довжини сегменту кабелю з таблиці 5.5 було розраховано довжину кабелів для кожної мережі ЛКМ.

Таблиця 5.5 – Задані стандарти мереж

Тип мережі	Довжина сегменту, м
10Base-T	100
100Base-2	185
100Base-5	500
100Base-FL	2000
Token Ring	101

Отримані значення округлені в сторону більшої довжини для надійності:

Гілка 1 – стандарт B5: (50*500)/60 = 420 метрів.

Гілка 2 – стандарт BT: 0.7*100 = 70 метрів.

Гілка 4 – стандарт B2: (40*185)/30 = 250 метрів.

Гілка 7 – стандарт BT: (40*100)/50 = 80 метрів

Гілка 8 – стандарт BF: 0.9*2000 = 1800 метрів.

Гілка 10 – стандарт TR: 0.9*101 = 95 метрів.

Гілка 11 – стандарт BT: 0.7*100 = 70 метрів.

Гілка 12 – стандарт BT: 0.8*100 = 80 метрів.

Таблиця 5.6 – Параметри гілок ЛКМ.

№	Параметри ЛКМ	Номер гілки графу (ЛКМ)							
		1	2	4	7	8	10	11	12
1	Стандарт	B5	BT	B2	BT	BF	TR	BT	BT
	Кількість PC, N	50	30	40	40	12	30	15	30
	Протяжність L	420	70	250	80	1800	95	70	80

Так як приміщення, в яких планується установка та подальша робота з комп’ютером, повинні відповідати документації будинку. Площа на одного робітника, який працює за комп’ютером, повинна складати не менше 6,0 кв. м. Будинок має три поверхи. На рисунку 5.2 зображено розташування кімнат.

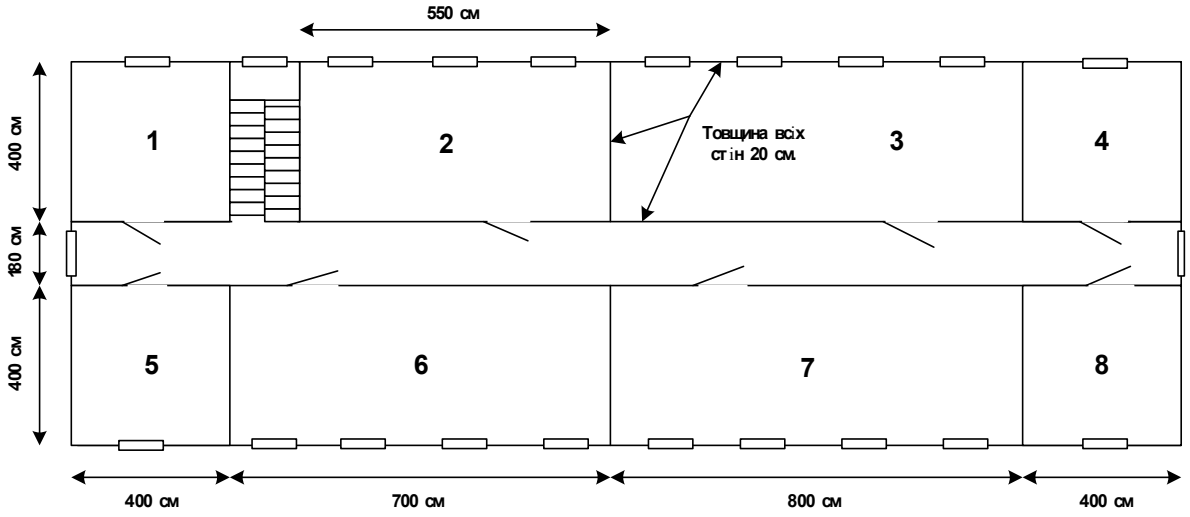


Рисунок 5.2 – Основні розміри кімнат на поверсі

На кожному поверсі можна розташувати 24 комп’ютери, та сервер в окремій кімнаті. В таблиці 5.7 вказані нові параметри ЛКМ. Для кожної мережі розраховано певну кількість робочих станцій за умови, що на кожному поверсі розташовано 24 комп’ютерів:

1 поверх:

Гілка 1: 14 робочих станцій;

Гілка 2: 10 робочих станцій;

2 поверх:

Гілка 4: 10 робочих станцій;

Гілка 7: 10 робочих станцій;

Гілка 8: 4 робочих станцій;

3 поверх:

Гілка 10: 9 робочих станцій;

Гілка 11: 9 робочих станцій;

Гілка 12: 6 робочих станцій;

Під нові параметри ЛКМ було перераховано довжину кабелів у менших гілках локальних мереж. Отримані значення були округлені в сторону більшої довжини:

Гілка 1 – стандарт B5: (14*500)/60 = 120 метрів.

Гілка 2 – стандарт BT: 0.7*100 = 70 метрів.

Гілка 4 – стандарт B2: (10*185)/30 = 65 метрів.

Гілка 7 – стандарт BT: (10*100)/50 = 20 метрів

Гілка 8 – стандарт BF: 0.9*2000 = 1800 метрів.

Гілка 10 – стандарт TR: 0.9*101 = 95 метрів.

Гілка 11 – стандарт BT: 0.7*100 = 70 метрів.

Гілка 12 – стандарт BT: 0.8*100 = 80 метрів.

Таблиця 5.7 – Параметри гілок ЛКМ.

№	Параметри ЛКМ	Номер гілки графу (ЛКМ)							
		1	2	4	7	8	10	11	12
1	Стандарт	B5	BT	B2	BT	BF	TR	BT	BT
	Кількість PC, N	14	10	10	10	4	9	9	6
	Протяжність L	120	70	65	20	1800	95	70	80

5.4 Перелік необхідних матеріалів та обладнання

Необхідні матеріали, обладнання та мережне забезпечення, які використовуються для побудови корпоративної комп’ютерної мережі зазначені у таблиці 5.8.

Кабельна схема комп’ютерної мережі приведена у додатку В.

Таблиця 5.8 – Матеріали, обладнання та мережне програмне забезпечення корпоративної мережі

№	Назва обладнання	Один. Вимір	Кількість одиниць обладнання								
			№ ЛКМ								Всього
			1	2	4	7	8	10	11	12	
А. Мережні пристрої											
1	Мережевий комутатор Ethernet з 8 портами RJ-45	Шт.		1						1	2
2	Мережевий комутатор Ethernet з 16 портами RJ-45	Шт.				1			1		2
3	Оптичний концентратор з 8 портами TX та RX	Шт.					1				1
4	NIC Ethernet з портом RJ-45	Шт.		11		11			10	7	39
5	NIC Ethernet з портом DB-15	Шт.	15				6				21
6	Трансивер 10Base5	Шт.	15								15
7	Оптично-волоконний трансивер	Шт.					6				6
8	Мережна інтерфейсна карта BNC	Шт.			11						11
9	Мережевий комутатор з 16 портами MIC	Шт.						1			1
10	NIC Token Ring з портом MIC	Шт.						10			10
11	Модем з інтерфейсом Ethernet	Шт.									1
Б. З'єднувачі											
17	Коннектор RJ-45	Шт.		22		22			20	14	78
20	DB15 – коннектор	Шт.	30				12				42

№	Назва обладнання	Один. Вимір	Кількість одиниць обладнання								
			№ ЛКМ								Всього
			1	2	4	7	8	10	11	12	
21	ST – коннектор	Шт.					24				24
23	BNC – коннектори	Шт.			22						22
24	BNC-T – коннектор	Шт.			11						11
25	Коннектор MIC	Шт.						20			20
В. Кабелі											
26	Кабель 2 пари UTP 3 категорії	М.		700		200			630	480	2010
27	Коаксіальний кабель RG11	М.	120								120
28	Трансиверний інтерфейсний AUI (4 пари UTP)	М.	300				120				420
29	Оптично-волоконний кабель	М.					1800				1800
30	Коаксіальний кабель RG58	М.			65						65
31	Екранований кабель. Тип 1	М.						950			950
Г. Комп’ютери											
32	Core i9-7980XE / RAM 64 ГБ / SSD 1 ТБ / LAN / GTX 1080Ti	Шт.	14	10	10	10	4	9	9	6	75
33	XEON Platinum 8180M / RAM 512 ГБ / SSD 20 ТБ / LAN 10 Гбит	Шт.		1		1			1		3
Д. Мережеві ОС											
34	Windows 11	Шт.	14	10	10	10	4	9	9	6	75
35	Windows Server 2016	Шт.		1		1			1		3
36	UNIX	Шт.									3

5.5 Просторові показники сигналу ЛКМ

В заданій комп’ютерній мережі використовуються 3 маршрутизатори. Тому необхідно для кожної ЛКМ виконати перевірку значення подвійного обороту сигналу. Згідно стандарту, найбільша протяжність мережі між станціями не повинна перевищувати 2500 метрів.

Таблиця 5.9 – Максимальна відстань передачі

Стандарт	Максимальна відстань передачі
100Base-2	925 метрів
100Base-5	2500 метрів
10Base-T	200 метрів

Оскільки в комп’ютерній мережі використовуються маршрутизатори, то перевірку треба виконувати для кожної локальної мережі:

ЛКМ 1 – 120 + 2*20 = 160 метрів.

ЛКМ 2 – 70*2 = 140 метрів.

ЛКМ 4 – 65*2 = 130 метрів.

ЛКМ 7 – 20*2 = 40 метрів

ЛКМ 8 – 1800 + 2*20 = 1840 метрів.

ЛКМ 10 – 95(9-1) = 760 метрів.

ЛКМ 11 – 70*2 = 140 метрів.

ЛКМ 12 – 80*2 = 160 метрів.

Після розрахунків результатів в жодній ЛКМ не перевищується допустиму протяжність вузлів мережі між робочими станціями.

5.6 Час затримки сигналів в ЛКМ

В комп’ютерній мережі використовують 5 стандартів: 10Base-T, 100Base-2, 100Base-5, 100Base-FL та Token Ring. Розрахунок часу подвійного

обороту необхідно виконувати лише для мереж ЛКМ та ЛКМ. Значення часу подвійного обороту не має перевищувати 575 bt. Тоді вузли мереж зможуть вчасно знаходити колізію.

Найбільша відстань між вузлами ЛКМ складає:

ЛКМ 1 + ЛКМ 4 + ЛКМ 8 + ЛКМ 10 = 120 + 65 + 1800 + 95 = 2080 метрів.

Використовуючи таблицю 1.10 було розраховано значення PDV:

Лівий сегмент ЛКМ 1: 11,8 + 120*0.0866 = 23.22 bt.

Проміжний сегмент ЛКМ 4: 46,5 + 65*0.1026 = 53.17 bt.

Проміжний сегмент ЛКМ 8: 33,5 + 1800*0.1 = 213.5 bt.

Правий сегмент ЛКМ 10: 165,0 + 95*0.113 = 175.74 bt.

PDV: 23.22 + 53.17 + 213.5 + 175.74 = 465.63 bt.

Перевірка в зворотному напрямку:

Лівий сегмент ЛКМ 10: 15,3 + 120*0.0866 = 25.7 bt.

Проміжний сегмент ЛКМ 8: 33,5 + 1800*0.1 = 213.5 bt.

Проміжний сегмент ЛКМ 4: 46,5 + 65*0.1026 = 53.17 bt.

Правий сегмент ЛКМ 1: 169,5 + 95*0.113 = 180.24 bt.

PDV: 25.7 + 213.5 + 53.17 + 180.24 = 472.61 bt.

Отримані результати не перевищають стандарту міжкадрового інтервалу, оскільки менше 49 бітових інтервалів.

Таблиця 5.10 – Затримки сигналу мережевим обладнанням.

Тип сегмента	База лівого сегмента,	База проміжног	База правого	Затримка середовища	Максималь на довжина
100Base-5	11,8	46,5	169,5	0,0866	500
100Base-2	11,8	46,5	169,5	0,1026	185
10Base-T	15,3	42,0	165,0	0,113	100
100Base-FB	-	24,0	-	0,1	2000
100Base-FL	12,3	33,5	156,5	0,1	2000
FOIRL	7,8	29,0	152,0	0,1	1000
AUI (>2 м)	0	0	0	0,1026	2+48

Використовуючи таблицю 5.11 розраховано час затримки сигналу в комп’ютерній мережі.

ЛКМ 1: 100Base-5, комутатор, протяжність 120 метрів:

0.12*4.33 + 0.1 = 0.62 мкс. = 0.062 bt.

ЛКМ 2: 10Base-T, комутатор, протяжність 70 метрів:

0.07*5.7 + 0.1 = 0.5 мкс. = 0.05 bt.

ЛКМ 4: 100Base-2, протяжність 65 метрів:

0.065*5.14 + 0.1 = 0.44 мкс. = 0.044 bt.

ЛКМ 7: 10Base-T, комутатор, протяжність 20 метрів:

0.02*5.7 + 0.1 = 0.22 мкс. = 0.022 bt.

ЛКМ 8: 10Base-F, повторювач, протяжність 1800 метрів:

1.8*5.0 + 0.1 = 9.1 мкс. = 0.91 bt.

ЛКМ 10: Token Ring, комутатор, протяжність 95 метрів:

0.095*5.7 + 0.1 = 0.65 мкс. = 0.065 bt.

ЛКМ 11: 10Base-T, комутатор, протяжність 70 метрів:

0.07*5.7 + 0.1 = 0.5 мкс. = 0.05 bt.

ЛКМ 12: 10Base-T, комутатор, протяжність 80 метрів:

0.08*5.7 + 0.1 = 0.56 мкс. = 0.056 bt.

Таблиця 5.11 – Затримки сигналу мережним обладнанням

Назва обладнання	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, концентратор, міст, комутатор	0.1 мкс/км
Оптично-волоконний повторювач чи HUB	1.55 мкс/км
Оптично-волоконний трансивер	0.2 мкс/км

Для розрахунку зменшення міжкадрового інтервалу повторювачами було визначено найбільшу відстань передачі пакету в мережі. Схема зображена на рисунку 5.3.

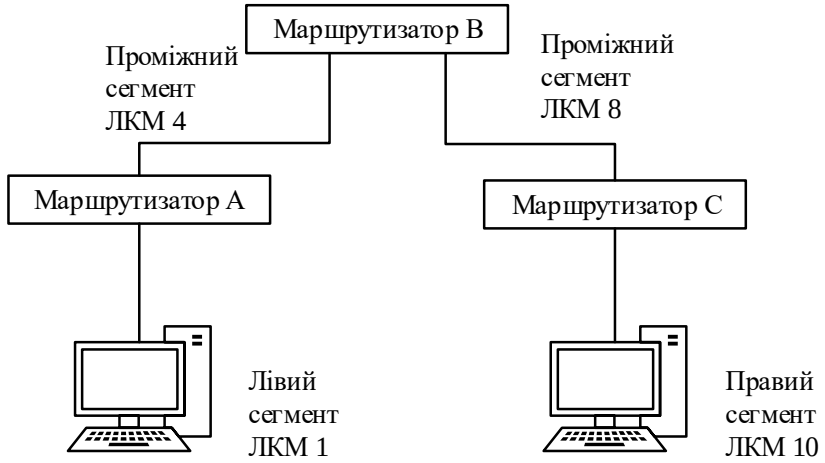


Рисунок 5.3 – Схема ЛКМ із найбільшою протяжністю

Найбільша відстань складає:

ЛКМ 1 + ЛКМ 4 + ЛКМ 8 + ЛКМ 10 = 120 + 65 + 1800 + 95 = 2080 метрів.

Використовуючи таблицю 1.12 було розраховано значення PVV:

Лівий сегмент 1 (10Base-5) 10.5 + 0.062 = 10.562 bt.

Проміжний сегмент 4 (10Base-2) 2 + 0.044 = 2.044 bt.

Проміжний сегмент 8 (10Base-FL) 8 + 0.91 = 8.91 bt.

PVV = 10.562 + 2.044 + 8.91 = 21.516 bt.

Лівий сегмент 10 (Token Ring) 16 + 0.065 = 16.065 bt.

Проміжний сегмент 8 (10Base-FL) 8 + 0.91 = 8.91 bt.

Проміжний сегмент 4 (10Base-2) 2 + 0.044 = 2.044 bt.

PVV = 16.065 + 8.91 + 2.044 = 27.019 bt.

Згідно з отриманими розрахунками міжкадровий інтервал відповідає стандарту, оскільки менше 49 бітових інтервалів.

Таблиця 5.12 – Зменшення міжкадрового інтервалу повторювачами

Тип сегменту	Передавальний сегмент, bt	Проміжний сегмент, bt
10Base-T	16	11
100Base-2	-	2
100Base-5	10.5	8
100Base-FL	10.5	8

5.7 Адресація вузлів у мережі

5.7.1 Кількість зайнятих IP-адрес у мережах

Кількість зайнятих IP-адресів в локальних комп'ютерних мережах визначається сумою вузлів, серверів та маршрутизаторів. Кількість зайнятих адрес ЛКМ відображено в таблиці 5.13.

Таблиця 5.13 – Кількість зайнятих IP-адрес

Номер ЛКМ	Кількість адрес
1	14
2	10
4	10
7	10
8	4
10	9
11	9
12	6

5.7.2 Схема IP-адресації мереж та вузлів

В комп'ютерній мережі 191.110.P.X третій октет P означає адресу локальної підмережі, четвертий октет X визначає адресу вузлів ЛКМ.

Для комп’ютерної мережі, яка проектується, значення третього октету визначається за формулою:

P=8(n-1)+i, де i – номер гілки комп’ютерної мережі, n – номер варіанту.

Нижче отримані значення третього октету:

ЛКМ 1: P = 8*(10 – 1) + 1 = 73

ЛКМ 2: P = 8*(10 – 1) + 2 = 74

ЛКМ 4: P = 8*(10 – 1) + 4 = 76

ЛКМ 7: P = 8*(10 – 1) + 7 = 79

ЛКМ 8: P = 8*(10 – 1) + 8 = 80

ЛКМ 10: P = 8*(10 – 1) + 10 = 82

ЛКМ 11: P = 8*(10 – 1) + 11 = 83

ЛКМ 12: P = 8*(10 – 1) + 12 = 84

У таблиці 5.15 розписано адресний розподіл для всіх вузлів комп’ютерної мережі.

Таблиця 5.15 – Розподіл адресного простору

ЛКМ	Адреса підмережі	Широкомовний IP	Адреси вузлів	Маска мережі
1	191.110.73.0	191.110.73.255	191.110.73.1-191.110.73.254	255.255.255.0
2	191.110.74.0	191.110.74.255	191.110.74.1-191.110.74.254	255.255.255.0
4	191.110.76.0	191.110.76.255	191.110.76.1-191.110.76.254	255.255.255.0
7	191.110.79.0	191.110.79.255	191.110.79.1-191.110.79.254	255.255.255.0
8	191.110.80.0	191.110.80.255	191.110.80.1-191.110.80.254	255.255.255.0
10	191.110.82.0	191.110.82.255	191.110.82.1-191.110.82.254	255.255.255.0
11	191.110.83.0	191.110.83.255	191.110.83.1-191.110.83.254	255.255.255.0
12	191.110.84.0	191.110.84.255	191.110.84.1-191.110.84.254	255.255.255.0

5.7.3 Таблиці IP-маршрутизації

Використовуючи схему IP-адресації приведеної в Додатку Б, розроблені IP-таблиці маршрутів для маршрутизаторів комп'ютерної мережі, кінцевого та проміжного вузла в мережі. Отримані маршрути приведені в таблицях 5.16-5.20.

Таблиця 5.16 – Маршрутна таблиця маршрутизатора А.

IP-адреса мережі призначення	Маршрутизація		IP-адреса шлюзу	Вихідний мережний інтерфейс
191.110.73.0	Прямий		-	ie1
191.110.74.0	Прямий		-	ie2
default	Непрямий		191.110.76.11	ie0

Таблиця 5.17 – Маршрутна таблиця маршрутизатора В.

IP-адреса мережі призначення	Маршрутизація	IP-адреса шлюзу	Вихідний мережний інтерфейс
191.110.73.0	Непрямий	191.110.76.11	ie0
191.110.74.0	Непрямий	191.110.76.11	ie0
191.110.82.0	Непрямий	191.110.80.5	ie0
191.110.83.0	Непрямий	191.110.80.5	ie0
191.110.84.0	Непрямий	191.110.80.5	ie0
191.110.76.0	Прямий	-	ie1
191.110.79.0	Прямий	-	ie2
191.110.80.0	Прямий	-	ie0
default	Непрямий	191.110.80.5	ie0

Таблиця 5.18 – Маршрутна таблиця маршрутизатора С.

IP-адреса мережі призначення	Маршрутизація	IP-адреса шлюзу	Вихідний мережний інтерфейс
191.110.80.0	Прямий	-	ie0
191.110.82.0	Прямий	-	ie2
191.110.83.0	Прямий	-	ie1
191.110.84.0	Прямий	-	ie3
default	Непрямий	195.240.50.9	pp0

Таблиця 5.19 – Маршрутна таблиця кінцевого РС ЛКМ 11.

IP-адреса мережі призначення	Маршрутизація	IP-адреса шлюзу	Вихідний мережний інтерфейс
191.110.83.0	Прямий	-	ie1
default	Непрямий	191.110.80.1	ie1

Таблиця 5.20 – Маршрутна таблиця проміжного РС ЛКМ 4.

IP-адреса мережі призначення	Маршрутизація	IP-адреса шлюзу	Вихідний мережний інтерфейс
191.110.79.0	Прямий	-	ie2
191.110.80.0	Прямий	-	ie0
191.110.76.0	Прямий	-	ie1
default	Непрямий	191.110.76.1	ie1

Default – напрям за замовчуванням

5.8 Алгоритм маршрутизації пакету

Для вузла ЛКМ 2 за IP-адресом 191.110.74.5, було розроблено алгоритм маршрутизації пакету, який відправляється вузлу ЛКМ 8 з IP-адресою 191.110.80.2. послідовність відправки пакету було зображено на рисунку 5.4.

Вузол використовує власну таблицю маршрутизації. Саме по ній визначається IP-адреса шлюзу маршрутизатора А (191.110.74.1/24), на який

відправляється пакет. Маршрутизатор А перевіряє власну таблицю маршрутизації, не знайшовши у власній таблиці необхідний вузол, маршрутизатор відправляє пакет на шлюз за замовчуванням (191.110.76.11/24). Пакет отримує маршрутизатор В. Маршрутизатор В перевіривши власну таблицю маршрутизації, визначає що вузол призначення знаходиться в одній з мереж, безпосередньо підключеної до нього. Після чого маршрутизатор С передає пакет отримувачу (191.110.80.2/24).

Рисунок 5.4 – Алгоритм маршрутизації пакету в мережі

6 МОДЕЛЮВАННЯ СЕРВЕРНОГО ПРИМІЩЕННЯ

Для комп’ютерної мережі треба спроектувати серверне приміщення, що покриють необхідні навантаження мережі. Для кожного ядра в серверній має обробляти одну операцію. Загальний обсяг пам’яті серверної має бути рівним або більшим за необхідний. Також обсяг дискового простору має бути рівним або більшим за необхідний. Нижче наведені таблиці 6.1-6.4 із специфікаціями обладнання для проектування серверної кімнати.

Таблиця 6.1 – Специфікація доступних серверів

Специфікація	Кількість ядер	ОЗП	Обсяг диску	Блок живлення	Висота в стійці, U	Вага
Специфікація 1	64	128	2ТБ	600Вт	2U	7 кг
Специфікація 2	128	256	2ТБ	800Вт	3U	9 кг
Специфікація 3	16	128	16ТБ	700Вт	4U	10 кг
Специфікація 4	16	64	32ТБ	650Вт	3U	8 кг
Специфікація 5	16	256	2ТБ	550 Вт	2U	7 кг

Таблиця 6.2 – Специфікація доступних комутаторів

Комутатор	Кількість портів	Блок живлення	Висота в стійці, U	Вага
Комутатор 1	16+2	100 Вт	1U	4 кг
Комутатор 2	24+2	120 Вт	1U	4 кг
Комутатор 3	32+2	140 Вт	2U	5 кг
Комутатор 4	48+2	180 Вт	2U	6 кг

Таблиця 6.3 – Специфікація доступних джерел безперебійного живлення

Джерело	Потужність	Висота в стійці, U	Вага	Кількість розеток для підключення обладнання
Джерело 1	6000 Вт	2U	10 кг	10
Джерело 2	7500 Вт	3U	12 кг	12
Джерело 3	9000 Вт	3U	14 кг	14
Джерело 4	10500 Вт	4U	18 кг	16

Таблиця 6.4 – Специфікація доступних серверних стійок

Серверна стійка	Висота, U	Висота	Ширина	Глибина	Вага
Стійка 1	42U	1990 см	60 см	100 см	125 кг
Стійка 2	40U	1880 см	60 см	100 см	116 кг
Стійка 3	38U	1770 см	60 см	100 см	110 кг

Висота приміщення: 2,6м = 2600мм.

Враховуємо 300мм для фальш підлоги та простір для систем охолодження над стійкою 300мм.

Максимальна висота для стійки = 2600мм – 300мм – 300мм = 2000мм.

Стійка 1 (1990см) максимальною висотою підходить під встановлення в серверній кімнаті.

Розрахунок необхідної кількості серверів. Підбираємо комбінацію серверів для покриття вимог: 55000 ядер, 48000 ГБ ОЗВ, 3000 ТБ дискового простору.

Розрахунок для покриття потреби в ядрах: було обрано «специфікацію 2», що містить 128 ядер. 55000/128 = 430 серверів.

Розрахунок ОЗП: 430 * 256 = 110080 ГБ

Розрахунок дискового простору: 430 * 2 = 860 ТБ

Розрахунок для покриття дефіциту дискового простору, який необхідно покрити 4 специфікацією серверу: 3000 – 860 = 2140 ТБ.

2140 ТБ / 32 ТБ = 67 серверів.

Підсумкова кількість серверів для поточної серверної кімнати, що покриють необхідні навантаження: 430 серверів 2 специфікації та 67 серверів 4 специфікації. Тобто загалом потрібно 497 серверів.

Було розраховано об’єм серверів в юнітах (U):

Сервера «спец.2»: 430 * 3 = 1290 U.

Сервера «спец.4»: 67 * 3 = 201 U.

Загалом потрібно 1290 + 201 = 1491 юніти.

Для живлення було використано джерело «специфікації 4», як найпотужніше із можливих варіантів.

Для підключення 497 серверів потрібно використовувати комутатор на 48 портів. Кількість комутаторів 497 серверів / 48 портів = 11 комутаторів. Ці комутатори займають 11*2 = 22 юніти.

Розрахунок кількості стійок для серверів та комутаторів. Загальний об’єм обладнання, включаючи сервера та комутатори: 1491 + 22 = 1513 юнітів.

Кожна стійка має 42 юніти, для кожної стійки займає ДЖБ в 4 юніти. Розраховано корисний об’єм однієї стійки: 42 – 4 = 38 юнітів. Необхідна кількість стійок: 1513 / 38 = 40 стійок.

Для розміщення всього обладнання нам знадобиться 40 стійок. В кожній буде встановлено одне «джерело».

№ Планування приміщень та перевірка навантаження на підлогу.

Розрахунок максимальної кількості стійок в одному приміщенні: розміри 1 стійки – ширина 60 см, глибина 100 см.

Також є вимоги для розташування стійок: відстань від стіни – 1 метр, та прохід між рядами стійками – 1 метр.

У кімнаті шириною 7 метрів можна розмістити три ряди стійок: 1м (стіна) + 1м (глибина ряду 1) + 1м (прохід) + 1м (глибина ряду 2) + 1м (прохід) + 1м (глибина ряду 3) + 1м (стіна).

У довжину 8 метрів по 1м від стіни, тому можна розмістити: 6м / 0.6м (ширина стійки) = 10 стійок в одному ряду.

Максимум на приміщення: 3 ряди * 10 стійок = 30 стійок.

Кількість приміщень: 40 стійок / 30 = 2 приміщення з параметрами 7м*8м.

Перевірка навантаження на підлогу в приміщенні: максимальне допустиме навантаження 350 кг/м². Було розраховано загальну вагу всього обладнання:

Стійки: 40 * 125 = 5000 кг

Сервера: 430 * 9 + 67 * 8 = 3870 + 536 = 4406 кг

Комутатори: 11 * 6 = 66 кг

ДБЖ: 40 * 18 = 720 кг

Загальна вага: 5000 + 4406 + 66 + 720 = 10192 кг.

Загальна площа приміщень під серверне обладнання: 6 * 11 = 66 м²

Було розраховано навантаження для загальної площі в серверному приміщенні: 10192 кг / 66 м² = 155 кг/м² (менше дозволених 350 кг/м²).

Схематичний вигляд одного приміщення зображено на рисунку 6.1.

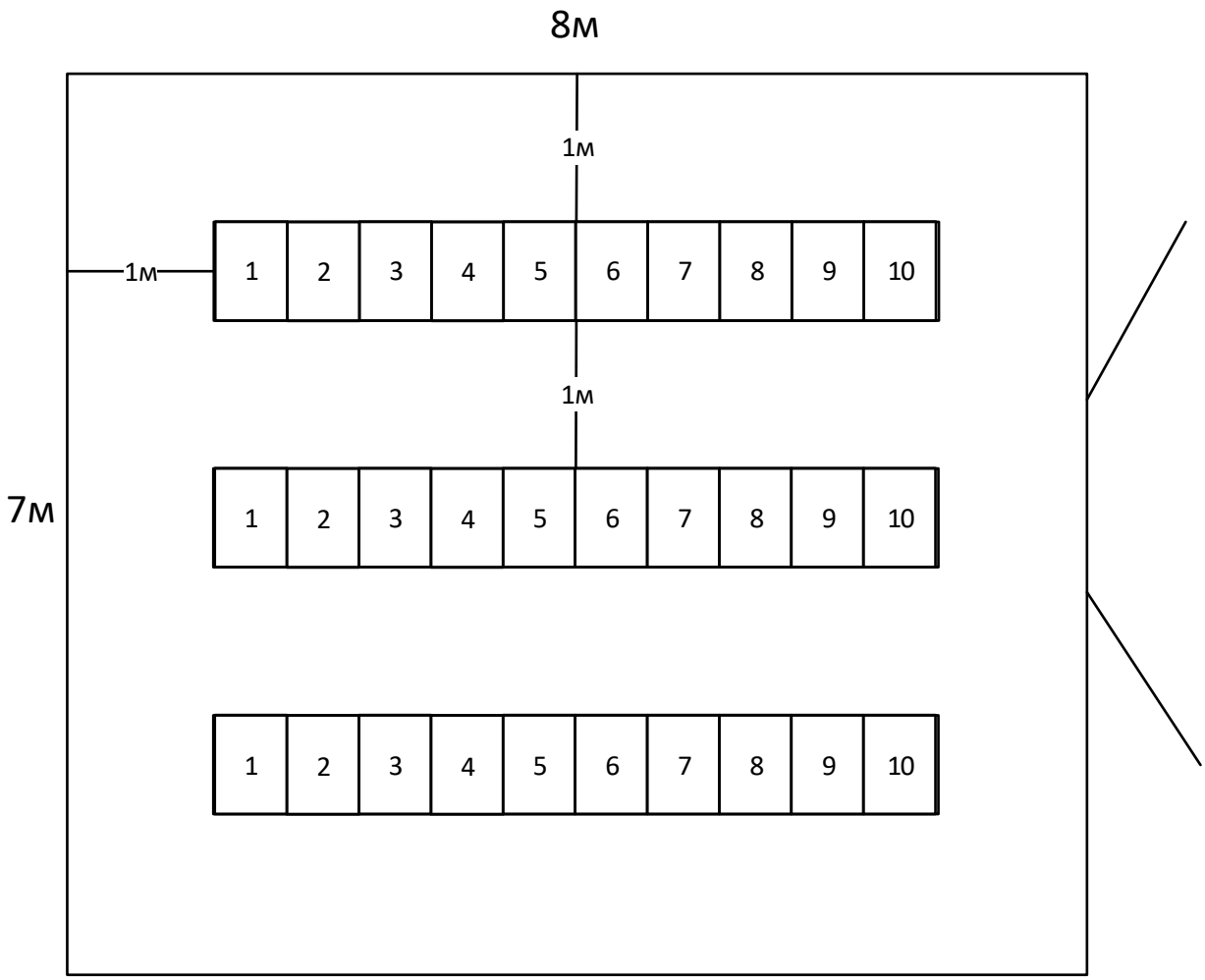


Рисунок 6.1 – Схема приміщення

ВИСНОВОК

В результаті виконання кваліфікаційної роботи були розглянуті загальні питання, пов’язані із діагностикою ноутбуків, LPC-шиною та POST-кодами.

Були вибрані мікросхеми та комплектуючи деталі для роботи пристрою, та була спроектована принципова схема пристрою для діагностики ноутбуку Asus ROG Strix G17 2023.

Розроблена програма пристрою діагностики на мові програмування С з урахуванням платформи на мікроконтролері Raspberry Pi Pico, а також вивід на LCD-дисплей інформації діагностики.

Спроектована корпоративна комп’ютерна мережа та моделювання серверного приміщення в рамках цієї мережі.

Детально спроектована корпоративна комп’ютерна мережа в будівлі з трьома поверхами, визначено перелік необхідних матеріалів для забезпечення корпоративної комп’ютерної мережі, отримано схему IP-адресації вузлів та мереж.

Спроектовано інфраструктуру для локальної хмари. Змодельовано серверне приміщення, оптимізовано обладнання. Встановлено, що для задоволення заданих вимог до обчислювальних ресурсів необхідно розгорнути обладнання у двох окремих серверних кімнатах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ноутбук MyASUS – Діагностика системи. URL: https://www.asus.com/ua-ua/support/faq/1045716/ (дата звернення: 31.05.2026).
2. Low Pin Count. URL: https://en.wikipedia.org/wiki/Low_Pin_Count (дата звернення: 31.05.2026).
3. Intel Low Pin Count (LPC). URL: https://www.intel.com/content/dam/www/program/design/us/en/documents/low-pin-count-interface-specification.pdf (дата звернення: 31.05.2026).
4. Інтерфейс LPC (Low Pin Count). URL: https://grokipedia.com/page/Low_Pin_Count (дата звернення: 31.05.2026).
5. Що таке протокол eSPI та як він покращує LPC. URL: https://www.totalphase.com/blog/2021/09/what-is-the-espi-protocol-and-how-does-it-improve-upon-lpc/ (дата звернення: 31.05.2026).
6. POST-карта. URL: https://uk.wikipedia.org/wiki/ POST-карта (дата звернення: 31.05.2026).
7. Плати мікроконтролерів Pico. URL: https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html (дата звернення: 31.05.2026).
8. Мікроконтролер Raspberry Pi Pico. URL: https://evo.net.ua/mikrokomputer-raspberry-pi-pico/ (дата звернення: 31.05.2026).
9. Raspberry Pi PICO. URL: https://arduino.ua/prod4358-raspberry-pi-pico (дата звернення: 31.05.2026).
10. Data Sheet SLB 9660. URL: https://www.infineon.com/assets/row/public/documents/30/49/infineon-data-sheet-slb9660-1.2-rev1.2-ds-en.pdf (дата звернення: 31.05.2026).
11. SLB 9660 TT1.2 FW4.40. URL: https://www.bdtic.com/en/infineon/SLB+9660+TT12+FW440 (дата звернення: 31.05.2026).

12. Символьний дисплей 1602A LCD module. URL: https://www.mini-tech.com.ua/ua/1602-simvolniy-displey-zeleniy (дата звернення: 31.05.2026).

13. Як використовувати 16x2 РК-дисплей та модуль I2C з Arduino. URL: https://www.instructables.com/How-to-Use-16x2-Lcd-and-I2C-Module-With-Arduino/ (дата звернення: 31.05.2026).

14. 16x2 РК-дисплей з Raspberry Pi та I2C. URL: https://www.hackster.io/Techatronic/16x2-lcd-display-with-raspberry-pi-using-i2c-39009f (дата звернення: 31.05.2026).

15. Що таке шифрування. URL: https://www.cloudflare.com/learning/ssl/what-is-encryption/ (дата звернення: 31.05.2026).

16. Що таке шифрування. URL: https://www.internetsociety.org/issues/encryption/what-is/ (дата звернення: 31.05.2026).

17. Що таке шифрування і як воно працює. URL https://cloud.google.com/learn/what-is-encryption (дата звернення: 31.05.2026).

18. Що таке симетричне шифрування. URL: https://www.geeksforgeeks.org/ethical-hacking/what-is-a-symmetric-encryption/ (дата звернення: 31.05.2026).

19. Асиметричне шифрування. URL: https://www.geeksforgeeks.org/computer-networks/what-is-asymmetric-encryption/ (дата звернення: 31.05.2026).

20. Розширений стандарт шифрування (AES). URL: https://www.geeksforgeeks.org/computer-networks/advanced-encryption-standard-aes/ (дата звернення: 31.05.2026).

21. Пояснення шифрування AES: як воно працює, переваги та реальне використання. URL: https://www.splashtop.com/blog/aes-encryption (дата звернення: 31.05.2026).

22. Розширений стандарт шифрування (AES). URL https://en.wikipedia.org/wiki/Advanced_Encryption_Standard (дата звернення: 31.05.2026).

23. Як використовується AES-шифрування в кібербезпеці та чому це важливо. URL: https://doverunner.com/blogs/how-to-use-aes-encryption-and-its-importance/ (дата звернення: 31.05.2026).

24. Ascon специфікація. URL: https://ascon.isec.tugraz.at/specification.html (дата звернення: 31.05.2026).

25. Аскон (шифр). URL: https://en.wikipedia.org/wiki/Ascon_(cipher) (дата звернення: 31.05.2026).

26. Ascon: Крихітний титан безпеки Інтернету речей (глибоке технічне занурення). URL https://thesagekhan.medium.com/ascon-the-tiny-titan-of-iot-security-a-deep-technical-dive-8273ab4786b6 (дата звернення: 31.05.2026).

27. Криптосистема RSA. URL: https://en.wikipedia.org/wiki/RSA_cryptosystem (дата звернення: 31.05.2026).

28. Алгоритм RSA в криптографії: пояснення Рівеста Шаміра Адлемана. URL: https://www.splunk.com/en_us/blog/learn/rsa-algorithm-cryptography.html (дата звернення: 31.05.2026).

29. Шифрування RSA та алгоритм RSA: повний огляд. URL: https://cybertalents.com/blog/rsa-encryption (дата звернення: 31.05.2026).

30. Що таке шифрування за алгоритмом RSA та як воно працює. URL: https://securew2.com/blog/what-is-rsa-asymmetric-encryption (дата звернення: 31.05.2026).

31. Алгоритм RSA в криптографії. URL: https://www.geeksforgeeks.org/computer-networks/rsa-algorithm-cryptography/ (дата звернення: 31.05.2026).

32. Шифрування RSA: Посібник для початківців. URL: https://medium.com/@Espress0/rsa-encryption-beginners-guide-6caee3e8cd80 (дата звернення: 31.05.2026).

33. Що таке RSA-шифрування і як воно працює. URL: https://nordvpn.com/ru/blog/rsa-encryption/ (дата звернення: 31.05.2026).

ДОДАТОК А

Код програми пристрою діагностики

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/i2c.h"
#include "hardware/clocks.h"
#include "lcd16x2_i2c.h"
#include "lpc_bus_sniffer.pio.h"

#define SYS_FREQ_IN_KHZ (198 * 1000)
#define LPC_BUS_PIN_BASE (0U)
#define LPC_BUS_PIN_COUNT (6U)
#define LED_POST_CODE_PIN_BASE (8U)
#define LED_POST_CODE_PIN_COUNT (8U)
#define LED_STATUS_PIN PICO_DEFAULT_LED_PIN
#define INTERVAL_IM_ALIVE_MS (1000)

#define LCD_I2C_PORT i2c1
#define LCD_I2C_SDA 14
#define LCD_I2C_SCL 15
#define LCD_I2C_FREQ 100000

typedef union __attribute__((__packed__)) {
  struct {
    uint16_t addr : 16;
    uint8_t cy_dir : 4;
    uint8_t start : 4;
    uint16_t resv : 8;
  }val;
  uint32_t value;
}lpcIOFrame;

void gpio_initialization() {
    gpio_init(LED_STATUS_PIN);
    gpio_set_dir(LED_STATUS_PIN, GPIO_OUT);
    for(uint i = LPC_BUS_PIN_BASE; i < LPC_BUS_PIN_BASE +
LPC_BUS_PIN_COUNT; i++) {
        gpio_init(i);
        gpio_disable_pulls(i);
    }
    for(uint i = LED_POST_CODE_PIN_BASE; i < LED_POST_CODE_PIN_BASE +
LED_POST_CODE_PIN_COUNT; i++) {
        gpio_init(i);
        gpio_set_dir(i, GPIO_OUT);
```

```
        gpio_put(i, 1);
    }
}

uint32_t reverse_nibbles(uint32_t in_val) {
    uint32_t out_val = 0;
    for(uint8_t i = 0; i < 32; i += 4) {
        out_val <<= 4;
        out_val |= in_val >> i & 0xf;
    }
    return out_val;
}

int init_lpc_bus_sniffer(PIO pio) {
    uint offset;
    lcd_put_str("ініціалізація аналізу LPC шини");
    int sm = pio_claim_unused_sm(pio, true);
    if(sm < 0){
        lcd_put_str("Помилка: неможливо отримати free state");
        return -1;
    }
    if (pio_can_add_program(pio, &lpc_bus_sniffer_program)) {
        offset = pio_add_program(pio, &lpc_bus_sniffer_program);
    } else {
        lcd_put_str("Помилка: програма не може бути завантажена");
        return -2;
    }
    lpc_bus_sniffer_program_init(pio, sm, offset, LPC_BUS_PIN_BASE,
LPC_BUS_PIN_COUNT, LED_POST_CODE_PIN_BASE,
LED_POST_CODE_PIN_COUNT);
    pio_sm_set_enabled(pio, sm, true);
    lpcIOFrame filter;
    filter.value = 0;
    filter.val.start = 0x0;
    filter.val.cy_dir = 0x2;
    filter.val.addr = 0x80;
    lcd_put_str("Filter: 0x%08x\n", reverse_nibbles(filter.value));
    pio->txf[sm] = reverse_nibbles(filter.value);
    lcd_put_str("Програма завантажена в %d, sm: %d\n", offset, sm);
    return sm;
}

int main() {
    gpio_initialization();
    set_sys_clock_khz(SYS_FREQ_IN_KHZ, true);
    stdio_init_all();
```

```
    sleep_ms(500);
    PIO _pio0 = pio0;
    int sm_lpc_bus_sniffer = init_lpc_bus_sniffer(_pio0);
    if (sm_lpc_bus_sniffer < 0) return -1;

    i2c_init(LCD_I2C_PORT, LCD_I2C_FREQ);
    gpio_set_function(LCD_I2C_SDA, GPIO_FUNC_I2C);
    gpio_set_function(LCD_I2C_SCL, GPIO_FUNC_I2C);
    gpio_pull_up(LCD_I2C_SDA);
    gpio_pull_up(LCD_I2C_SCL);
    struct lcd16x2 lcd;
    lcd_init(LCD_I2C_PORT, LCD_I2C_ADDRESS, &lcd);

    lcd_put_str("Частота системи %u kHz\n",
frequency_count_khz(CLOCKS_FC0_SRC_VALUE_CLK_SYS));
    lcd_put_str("POST коди");

    while(1) {
        if (!pio_sm_is_rx_fifo_empty(_pio0, sm_lpc_bus_sniffer)) {
            uint32_t rxdata = _pio0->rxf[sm_lpc_bus_sniffer];
            lcd_put_str("Дані: %02x\n", rxdata);
        }
    }
}
```

ДОДАТОК Б

Програмне забезпечення захисту даних в корпоративній мережі

```
#include <iostream>
#include <fstream>
#include <vector>
#include <chrono>
#include <string>
#include <windows.h>
#include <commdlg.h>

using namespace std;

vector<unsigned char> readFile(const string& filename) {
    ifstream f(filename, ios::binary);
    if (!f.is_open()) {
        cerr << "Не вдалося відкрити файл: " << filename << endl;
        return {};
    }
    return vector<unsigned char>((istreambuf_iterator<char>(f)),
istreambuf_iterator<char>());
}

void writeFile(const string& filename, const vector<unsigned char>& data) {
    ofstream f(filename, ios::binary);
    if (!f.is_open()) {
        cerr << "Не вдалося записати у файл: " << filename << endl;
        return;
    }
    f.write((char*)data.data(), data.size());
}

vector<unsigned char> fakeAES(const vector<unsigned char>& data, const string& key) {
    vector<unsigned char> res = data;
    for (size_t i = 0; i < data.size(); ++i)
        res[i] ^= key[i % key.size()];
    return res;
}

vector<unsigned char> fakeASCON_encrypt(const vector<unsigned char>& data, const
string& key) {
    vector<unsigned char> res = data;
    for (size_t i = 0; i < data.size(); ++i)
        res[i] = (data[i] + key[i % key.size()]) % 256;
    return res;
}
```

```
vector<unsigned char> fakeASCON_decrypt(const vector<unsigned char>& data, const
string& key) {
    vector<unsigned char> res = data;
    for (size_t i = 0; i < data.size(); ++i)
        res[i] = (data[i] - key[i % key.size()] + 256) % 256;
    return res;
}


vector<unsigned char> fakeRSA_encrypt(const vector<unsigned char>& data, const string&
key) {
    vector<unsigned char> res = data;
    unsigned int e = 17;
    for (size_t i = 0; i < data.size(); ++i)
        res[i] = (unsigned char)((data[i] * e + key[i % key.size()]) % 256);
    return res;
}

vector<unsigned char> fakeRSA_decrypt(const vector<unsigned char>& data, const string&
key) {
    vector<unsigned char> res = data;
    unsigned int e = 17;
    for (size_t i = 0; i < data.size(); ++i) {
        int val = (data[i] - key[i % key.size()]);
        if (val < 0) val += 256;
        res[i] = (unsigned char)((val * 241) % 256);
    }
    return res;
}


string chooseFile() {
    char filename[MAX_PATH] = "";
    OPENFILENAMEA ofn;
    ZeroMemory(&ofn, sizeof(ofn));
    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = nullptr;
    ofn.lpstrFile = filename;
    ofn.nMaxFile = MAX_PATH;
    ofn.lpstrFilter = "All Files\0*.*\0Text Files\0*.TXT\0";
    ofn.nFilterIndex = 1;
    ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

    if (GetOpenFileNameA(&ofn))
        return string(filename);
```

```
    else
        return "";
}

int main() {
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);

        int method;
        int action;
        cout << "=== Алгоритми шифрування (AES, ASCON, RSA) ===\n";
        cout << "Оберіть метод шифрування\n"<<"1. AES\n"<<"2. ASCON\n"<<"3.
RSA\n";
        cin>>method;
        cout << "Оберіть дію\n"<<"1. Шифрування\n"<<"2. Дешифрування\n";
        cin>>action;
        cout << "Виберіть вхідний файл...\n";

    string filename = chooseFile();
    if (filename.empty()) {
        cout << "Файл не обрано.\n";
        return 0;
    }

    string key;
    cout << "Введіть ключ шифрування: ";
    cin >> key;

    vector<unsigned char> data = readFile(filename);
    if (data.empty()) {
        cout << "Вхідний файл порожній або не відкрився.\n";
        return 0;
    }
        cout << "Розмір файлу: " << data.size() << " байт\n\n";
        cout << "\nВиконання...\n";


        if(method == 2 && action == 1) {
                auto start = chrono::high_resolution_clock::now();
                auto enc_ascon = fakeASCON_encrypt(data, key);
                auto stop = chrono::high_resolution_clock::now();
                double enc_ascon_time = chrono::duration<double, milli>(stop - start).count();
                writeFile("encrypted_ascon.bin", enc_ascon);
                cout << "[ASCON] Шифрування: " << enc_ascon_time << " мс\n";
        }
```

```
    if(method == 2 && action == 2) {
        start = chrono::high_resolution_clock::now();
        auto dec_ascon = fakeASCON_decrypt(enc_ascon, key);
        stop = chrono::high_resolution_clock::now();
        double dec_ascon_time = chrono::duration<double, milli>(stop - start).count();
        writeFile("decrypted_ascon.txt", dec_ascon);
        cout << "[ASCON] Дешифрування: " << dec_ascon_time << " мс\n";
    }

    if(method == 1 && action == 1) {
        start = chrono::high_resolution_clock::now();
        auto enc_aes = fakeAES(data, key);
        stop = chrono::high_resolution_clock::now();
        double enc_aes_time = chrono::duration<double, milli>(stop - start).count();
        writeFile("encrypted_aes.bin", enc_aes);
        cout << "[AES] Шифрування: " << enc_aes_time << " мс\n";
    }

    if(method == 1 && action == 2) {
        start = chrono::high_resolution_clock::now();
        auto dec_aes = fakeAES(enc_aes, key);
        stop = chrono::high_resolution_clock::now();
        double dec_aes_time = chrono::duration<double, milli>(stop - start).count();
        writeFile("decrypted_aes.txt", dec_aes);
        cout << "[AES] Дешифрування: " << dec_aes_time << " мс\n";
    }

    if(method == 3 && action == 1) {
        start = chrono::high_resolution_clock::now();
        auto enc_rsa = fakeRSA_encrypt(data, key);
        stop = chrono::high_resolution_clock::now();
        double enc_rsa_time = chrono::duration<double, milli>(stop - start).count();
        writeFile("encrypted_rsa.bin", enc_rsa);
        cout << "[RSA] Шифрування: " << enc_rsa_time << " мс\n";
    }

    if(method == 3 && action == 2) {
        start = chrono::high_resolution_clock::now();
        auto dec_rsa = fakeRSA_decrypt(enc_rsa, key);
        stop = chrono::high_resolution_clock::now();
        double dec_rsa_time = chrono::duration<double, milli>(stop - start).count();
        writeFile("decrypted_rsa.txt", dec_rsa);
        cout << "[RSA] Дешифрування: " << dec_rsa_time << " мс\n";
    }
  return 0;
}
```

ДОДАТОК В

Рисунок В.1 – Кабельна схема корпоративної мережі

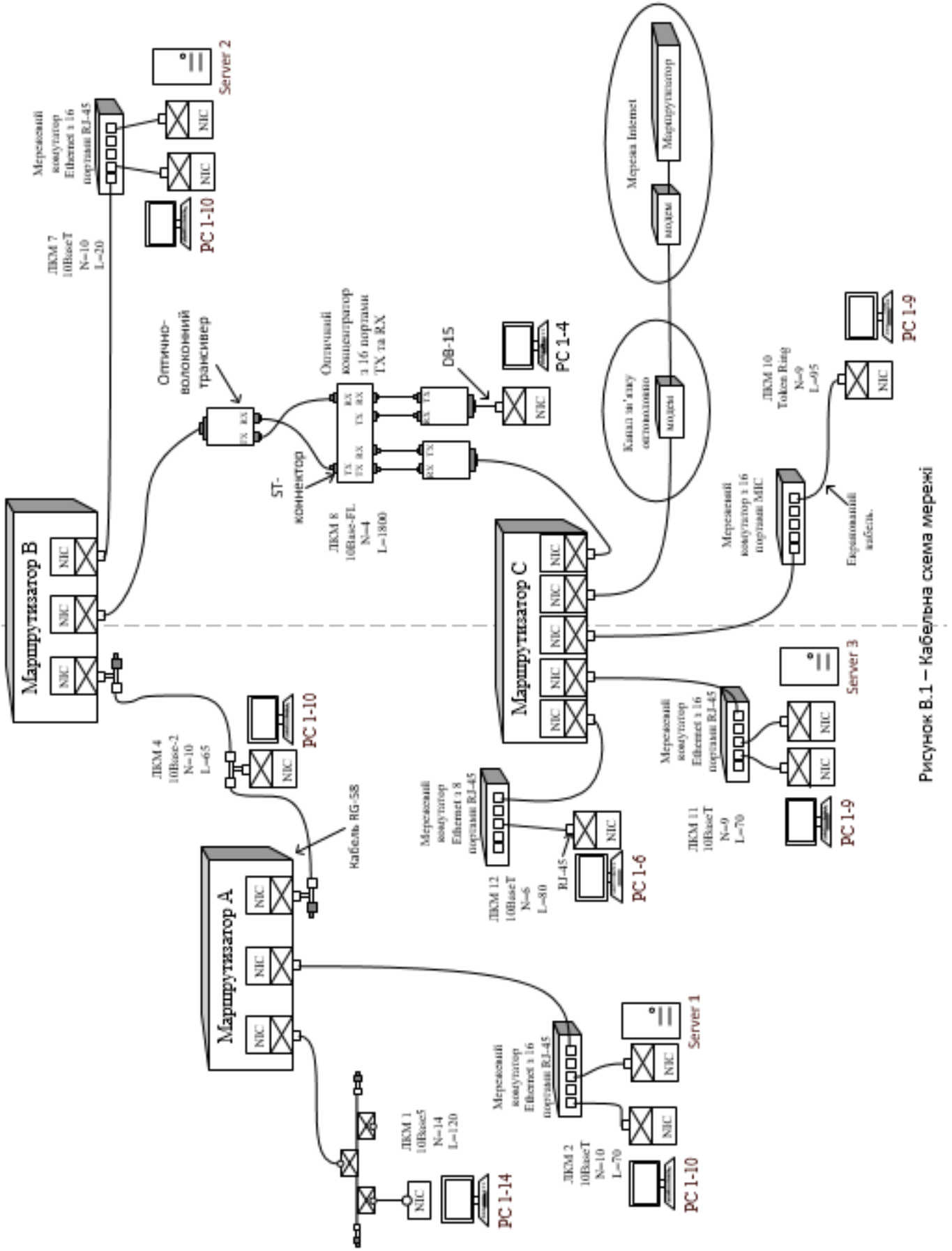


Рисунок В.1 – Кабельна схема мережі

ДОДАТОК Г

Рисунок Г.1 – Схема IP-адресації корпоративної мережі