МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра електроніки і комп'ютерної техніки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи магістра на тему:

«**Електронна система мультисервісного доступу до корпоративного хмарного сховища»**

Завідувач кафедри: Опанасюк А.С.

Керівник
кваліфікаційної роботи: Бережна О.В.

Консультант
з техніко-економічної частини: Маценко О. М

Виконав студент
гр. ЕС.м-11: Погуляй О.Р.

Суми 2022 р.

РЕФЕРАТ

Кваліфікаційна робота містить 102 сторінки, 73 рисунка, 14 таблиць, 32 джерела літератури.

Пояснювальна записка складається з семи розділів: огляд літератури і постановка задачі проектування, науково-дослідної частини, розроблення, обгрунтування алгоритму функціонування та структурної схеми, розробка функціональної та принципової схеми пристрою, розроблення функціональної схеми, розробка принципової схеми та вибір елементної бази, розроблення програмного забезпечення.

Графічна матеріал роботи містить схему-алгоритм, структурну схему, функціональну схему та принципову схеми системи мультисервісного доступу до корпоративного хмарного сховища.

У першому розділі проведений огляд можливих схем побудови хмарних сховищ, їх переваги та недоліки та проведений розгляд декількох конкурентних пристроїв.

Другий розділ містить огляд на роботи в сфері захисту інформації, в області безпеки передачі даних через канал зв'язку SSH. Розглянутий принцип роботи SSH протоколу, на основі якого були виділені частини, які потенційно можуть бути вразливими. Проаналізовано методи протидії втрати конфіденційності.

Третій розділ містить розробку алгоритму роботи системи мультисервісного доступу до корпоративного хмарного сховища. На базі схеми алгоритму побудована структурна схема цієї ж системи.

Четвертий розділ доповнює структурну схема, на базі чого проводиться розробка функціональної схеми системи.

П'ятий розділ містить вибір елементної бази, з деякими поясненнями що до принципу їх роботи, також приведені схеми деяких реалізацій в принциповій схемі.

Шостий розділ показує принцип налаштування системи на низинному рівні. Також він містить приклад коду, для виведення тексту на мові програмування асемеблер.

Сьомий розділ містить стандартну процедуру розрахунку техніко-економічної частини.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface;
SSD – Solid-State Drive;
P2P – peer-to-peer;
SSH – Secure Shel;
VMI – Virtual Machine Introspection;
MITM – man-in-the-middle;
PKL – Packet length;
PDL – Padding length;
MAC – Message authentication code;
ПК – персональний комп’ютер;
ІТ – інтернет-технології;
IP – Internet Protocol;
RAID – Redundant Array of Independent Disks;
ПЗ – програмне забезпечення;
TCO – Transmission Control Protocol;
ARM – Advanced RISC Machine;
ПЗП – Постійний запам'ятовуючий пристрій;
ОЗП – Оперативний запам'ятовуючий пристрій;
HDD – Hard Disk Driver;
NAS – Network Attached Storage;
DAS – Direct-Attached Storage;
S.M.A.R.T. – Self-Monitoring, Analysis and Reporting Technology;
NOOBS – New Out Of Box Software;
SSD – Solid-State Drive.

ВСТУП

Хмарні технології продовжують прогресувати з кожним роком. У результаті цього щороку з'являються нові онлайн-сервіси, давно відомих нам стаціонарних продуктів. Так наприклад, люди можуть використовувати службу Adobe для редагування відео на сайті Figma, а студенти використовувати робочу програму Multisim прямо в браузері, все що необхідно – доступ до мережі інтернет. Можна також використовувати послуги Nvidia для доступу до їхніх серверів і використовувати ресурси їхніх серверів для візуалізації відео та обчислень 3D-графіки. Вони пропонують підписку, яка дозволяє робити це віддалено. Крім того, ці послуги дозволяють користувачам купувати доступ до віддалених серверів для інших цілей. Кілька років тому це було неможливо, або дуже дорого, але зараз це нова, зручна реальність.

Люди не усвідомлюють, скільки даних вони зберігають у хмарі. Людям більше не потрібно транспортувати фізичні носії для зберігання; натомість вони можуть зберігати інформацію онлайн, не турбуючись про її пошкодження чи втрату.

Разом з новими технологіями формувалися нові бізнес-тенденції. Наприклад, деякі люди зберігають свої файли на Google Drive, Microsoft OneDrive або MegaCloud. Ці тенденції, разом із технологічним прогресом – спричинили необхідність розвитку іншої проблеми: стрімкого зростання витрат, пов'язаних із хмарним сховищем. Google безкоштовно пропонує 2 терабайти онлайн-сховища. Однак для безкоштовного використання доступно лише 15 гігабайт. За річну вартість 2,76 доларів США користувачі можуть отримати доступ до 2 додаткових терабайт пам'яті. Це значно дорожче, ніж використання хмарного сховища Google. Тому деякі компанії вирішили використовувати власне хмарне сховище замість Google [1].

NAS – це як маленький сервер, який може зберігати файли в різних форматах. Це утворення виникло, як альтернатива власників хмарних гігантів. Основною їх перевагою є безпека даних, адже сервер можна розташувати поряд з робочим офісом і не турбуватися про можливість викрадення даних.

1 ОГЛЯД ЛІТЕРАТУРИ І ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ

1.1 Огляд літератури

За останні кілька десятиліть, значно збільшилися об’єми даних в робочих програмах. Відповідно, це спричинило прискорене розширення робочих обсягів пам’яті в робочих терміналах і збільшення кількості цифрових записів. Експоненціально зростаючі обсяги даних, які обробляються великомасштабними розподіленими додатками з інтенсивним використанням даних, створюють зростаючий тиск на базові служби зберігання для своєчасного та ефективного зберігання та пошуку даних.

Використання хмарного сховища, приблизна структурна схема якого наведена на рис. 1.1, є однією з найкращих стратегій для ефективного зберігання зростаючих обсягів даних. Однак аутсорсинг даних у загальнодоступному хмарному сховищі призводить до проблеми збереження конфіденційності даних.

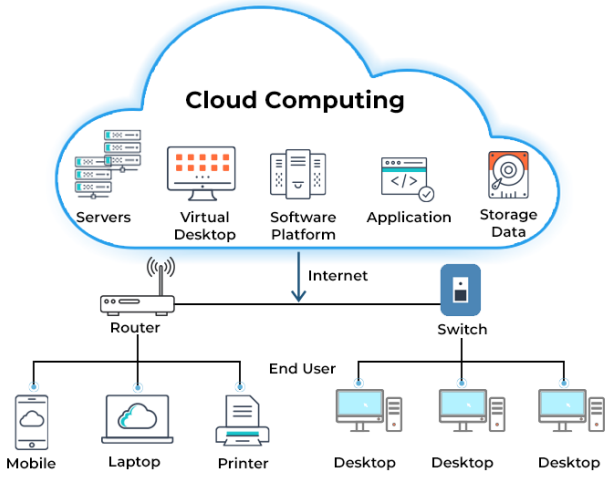


Рисунок 1.1 – Структурна схема хмарних обчислень

Конфіденційність даних є однією з головних проблем, пов’язаних із хмарним зберіганням даних, яка суттєво перешкоджає розповсюдженню хмарних обчислень у всьому світі і вважається серйозною проблемою.

Хмарне сховище – це модель обслуговування, у якій дані передаються та зберігаються у віддалених системах зберігання, де вони обслуговуються, керуються, створюються резервні копії та стають доступними для користувачів через мережу – як правило, Інтернет.

Хмарне сховище базується на віртуалізованій інфраструктурі зберігання з доступними інтерфейсами, майже миттєвою еластичністю та масштабованістю, можливістю багатокористувацького доступу та вимірюваними ресурсами. Хмарні дані зберігаються в логічних пулах на різних серверах зберігання, розташованих на території або в центрі обробки даних, яким керує сторонній хмарний постачальник. Існує три основні варіанти хмарного сховища на основі різних моделей доступу: загальнодоступне, приватне та гібридне.

Публічна хмара. Ці служби зберігання забезпечують середовище зберігання з кількома клієнтами, яке найбільше підходить для неструктурованих даних на основі передплати. Дані зберігаються в центрах обробки даних постачальника послуг із сховищем даних, розподілених по кількох регіонах або континентах. Клієнти зазвичай платять за використання, подібно до моделі комунальних платежів. У багатьох випадках також стягується плата за транзакції залежно від частоти та обсягу даних, до яких здійснюється доступ. У цьому секторі ринку домінують такі послуги:

- Amazon Simple Storage Service (S3);
- Amazon Glacier для глибокого архівування або холодного зберігання;
- Google Cloud Storage;
- Google Cloud Storage Nearline для холодних даних;
- Microsoft Azure.

Приватна хмара. Служба приватного хмарного сховища – це власний ресурс зберігання, який розгортається як спеціальне середовище, захищене брандмауером. Реалізації внутрішнього розміщення приватних хмарних сховищ емулюють деякі функції комерційних публічних хмарних служб, забезпечуючи легкий доступ і розподіл ресурсів зберігання для бізнес-користувачів, а також протоколи зберігання об’єктів. Приватні хмари підходять для користувачів, які потребують налаштування та більшого контролю над своїми даними або які мають суворі вимоги щодо безпеки даних чи нормативні вимоги.

Гібридна хмара. Цей варіант хмарного сховища є сумішшю приватного хмарного сховища та сторонніх публічних хмарних служб зберігання з рівнем оркестровки для оперативної інтеграції двох платформ.

Модель пропонує підприємствам гнучкість і більше варіантів розгортання даних. Організація може, наприклад, зберігати активно використовувані та структуровані дані в локальній приватній хмарі, а неструктуровані та архівні дані

– у загальнодоступній хмарі. Гібридне середовище також полегшує роботу з сезонними чи непередбаченими сплесками у створенні даних або доступі до зовнішньої служби зберігання даних через хмару, уникаючи необхідності додавати власні ресурси зберігання.

Впровадження моделі гібридної хмари зросло в останні роки. Незважаючи на переваги гібридних хмар, вони створюють технічні, бізнес та управлінські проблеми. Наприклад, приватні робочі навантаження повинні отримувати доступ і взаємодіяти з постачальниками загальнодоступних хмарних сховищ, тому сумісність і надійне та широке підключення до мережі є важливими факторами. Хмарна система зберігання корпоративного рівня має бути масштабованою відповідно до поточних і майбутніх потреб, доступною звідусіль і незалежною від додатків.

В табл. 1.1, наведемо порівняльну характеристику різних варіантів хмарного сховища.

Таблиця 1.1 – Порівняльна таблиця основних варіантів хмарного сховища

	Загальнодоступне хмарне сховище	Приватне хмарне сховище	Гібридне хмарне сховище
Масштабованість	Дуже висока	Обмежена	Дуже висока
Захист	Гарний, але залежить від провайдеру Інтернет послуг	Найбільш захищений	Прийнятний рівень захищеності
Продуктивність	Середня	Дуже добра	Добра, оскільки активний вміст кешується локально
Надійність	Середня; залежить від підключення до Інтернету	Висока, оскільки все обладнання знаходиться на місці	Висока, оскільки кешований вміст зберігається на місці, але також залежить від Інтернету
Ціна	Найнижча; модель оплати за використання й відсутність потреби в локальній інфраструктурі зберігання	Найдорожча, бо потрібні локальні ресурси, наприклад простір центру обробки даних. Електрика та охолодження	Середня, оскільки деякі дані зберігаються на місці і для цього також необхідна інфраструктура

Постачальники хмарних послуг керують і обслуговують дані, передані в хмару. Послуги зберігання надаються за запитом у хмарі, ємність збільшується та зменшується за потреби. Організації, які обирають хмарне сховище, усувають необхідність купувати, керувати та підтримувати власну інфраструктуру зберігання. Хмарне сховище радикально знизило вартість зберігання за гігабайт, але постачальники хмарних сховищ додали операційні витрати, що може зробити технологію значно дорожчою, залежно від того, як вона використовується.

Постачальники хмарних послуг обслуговують великі центри обробки даних у багатьох місцях по всьому світу. Коли клієнти купують хмарне сховище у постачальника, вони передають більшість аспектів зберігання даних постачальнику, включаючи безпеку, ємність, сервери зберігання та обчислювальні ресурси, доступність даних і доставку через мережу. Додатки клієнтів можуть отримувати доступ до збережених хмарних даних за допомогою традиційних протоколів зберігання або індикаторів прикладного програмування (API).

Принцип роботи хмарного сховища залежить від типу використовуваного сховища. Три основні типи це блочне сховище, сховище файлів і сховище об’єктів

Блокове зберігання, яке зображене на рис. 1.2, розділяє великі обсяги даних на менші одиниці, які називаються блоками. Кожен блок пов'язаний з унікальним ідентифікатором і розміщений на одному з системних накопичувачів. Блокове зберігання є швидким, ефективним і забезпечує низьку затримку, необхідну для таких програм, як бази даних і високопродуктивні робочі навантаження.

Рисунок 1.2 – Блочна система збереження

Файлове сховище організовує дані в ієрархічній системі файлів і папок; він зазвичай використовується з накопичувачами персональних комп'ютерів і мережевими накопичувачами (NAS). Дані в системі зберігання файлів зберігаються у файлах, а файли зберігаються в папках. Каталоги та підкаталоги використовуються для організації папок і пошуку файлів і даних. Хмара на основі сховища файлів може спростити доступ до даних і їх пошук, оскільки цей ієрархічний формат знайомий користувачам і потрібний для деяких програм.

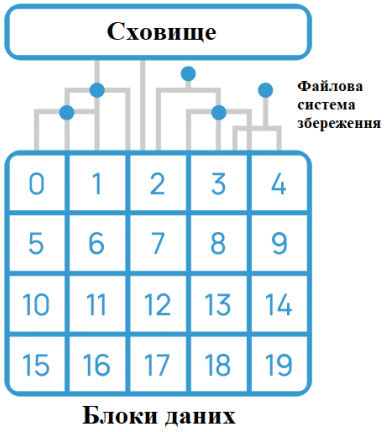


Рисунок 1.3 – Файлова система збереження

Об'єктне сховище зберігає дані як об'єкти, які складаються з трьох компонентів: дані, що зберігаються у файлі, метадані, пов'язані з файлом даних, і унікальний ідентифікатор. Використовуючи RESTful API, протокол зберігання об'єктів зберігає файл і пов'язані з ним метадані як єдиний об'єкт і призначає йому ідентифікаційний (ID) номер.

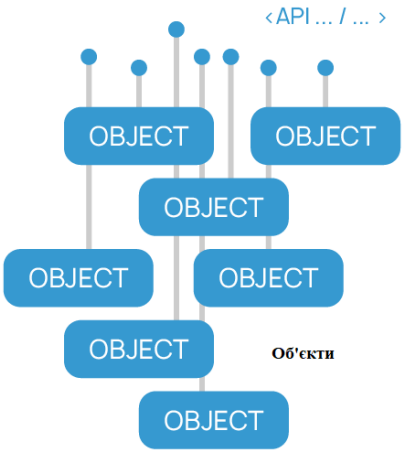


Рисунок 1.4 – Об'єктна система збереження

Щоб отримати вміст, користувач надає ідентифікатор системі, і вміст збирається з усіма метаданими, автентифікацією та безпекою. Об’єктні системи зберігання дозволяють налаштовувати метадані, що може оптимізувати доступ до даних і аналіз. За допомогою об’єктного сховища дані можна зберігати у рідному форматі з великою масштабованістю.

Останніми роками постачальники сховищ об’єктів додали функції та можливості файлової системи до свого програмного та апаратного забезпечення сховищ об’єктів головним чином тому, що сховище об’єктів не було прийнято досить швидко. Наприклад, шлюз хмарного сховища може надавати інтерфейс емуляції файлової системи для свого сховища об’єктів; таке розташування часто дозволяє програмам отримувати доступ до даних без підтримки протоколу зберігання об’єктів. Усі програми резервного копіювання використовують протокол зберігання об’єктів, що є однією з причин, чому резервне копіювання онлайн у хмарну службу було першим успішним додатком для хмарного зберігання.

Більшість комерційних хмарних служб зберігання даних використовують величезну кількість систем зберігання на жорстких дисках, встановлених на серверах, які об’єднані мережевою архітектурою, подібною до сітки. Постачальники послуг також додали високопродуктивні рівні до своїх пропозицій віртуальних сховищ, які зазвичай складаються з твердотільних накопичувачів (SSD). Високопродуктивне хмарне сховище зазвичай є найефективнішим, якщо сервери та програми, які отримують доступ до сховища, також знаходяться в хмарному середовищі.

Розглянемо найпопулярніших представників загальнодоступних хмарних сховищ і порівняємо їх основні характеристики. Основний параметр, який нас цікавить – обсяг простору для зберігання даних.

OneDrive, від компанії Microsoft**.** Пропонує майбутнім користувачам простий спосіб ділитися, зберігати та синхронізувати найпоширенішими типами файлів з іншими людьми робочої групи та пристроями в мережі через веб-інтерфейс або встановлену фірмову програму, яка чудово інтегрується в операційну систему Windows. Також є можливість синхронізувати користувацькі налаштування, такі як системні та візуальні налаштування. Навіть історія переглядів веб-сторінок та збережені паролі з веб-браузерів залишаються. Хмарний сервіс від Microsoft добре інтегрований у всі програмні продукти

Microsoft, тому для кожного тарифного плану були певні додаткові сервіси та програми. Ми наведемо в табл. 1.2 тарифні плани Microsoft [2].

Таблиця 1.2 – Тарифні плани Microsoft OneDrive

Тарифний план	Об'єм пам’яті, Гб	Ціна за місяць, грн	Ціна за рік, грн	Ціна 1 Гб, грн
OneDrive Standalone	100	71	759	7,59
Microsoft 365	1000	201	1956	19,56

Google drive, від компанії Google. Google величезна компанія, яка має в своїй власності велику кількість продуктів, саме хмарного типу. Це дозволяє їм інтегрувати та поєднувати продукти для більшого комфорту використання. Так Google drive, має онлайн-простір, з власною файловою системою в яку вже інтегровано наступне:

- Google Docs, інструмент для роботи з документами;
- Google Sheets, інструмент для роботи з таблицями;
- Google Presentations, інструмент для роботи з презентаціями;
- Google Keep, інструмент для роботи з нотатками.

З всіх розглянутих постачальників такого роду послуг, Google Drive має найбільший можливий обсяг простору – 30 ТБ. Функція редагування файлів доступна навіть в автономному режимі роботи, при під’єднанні до мережі відбудеться миттєва синхронізація з хмарою. Також Google drive має налаштування рівнів доступу, які полегшують роботу в компанії та запобігають зникненню важливих документів. Наведемо в табл. 1.3 тарифні плани Google. [3]

Таблиця 1.3 – Тарифні плани Google диск

Тарифний план	Об'єм пам’яті, Гб	Ціна за місяць, грн	Ціна за рік, грн	Ціна 1 Гб, грн
«Free»	15	0	0	0
«Business Starter»	100	54,96	659,56	6,60
« Business Standard»	200	77,83	933,93	4,67
« Business Plus»	2000	237,87	2854,47	1,43

Mega Cloud, від компанії Mega. Продукт від цього постачальника, можна назвати самим простим, адже користувач отримає лише те що хоче, а саме вільний простір в хмарному сховищі, без додаткових служб. Безкоштовний пакет, має найбільший простір серед конкурентів, але з неприємним обмеженням – фіксованим завантаженням файлів на 24 години. Наведемо тарифні плани Mega Cloud в табл. 1.4 [4].

Таблиця 1.4 – Тарифні плани Mega Cloud

Тарифний план	Об'єм памяті, gb	Ціна за місяць, грн	Ціна за рік, грн	Середня ціна 1 gb
Pro Standart	50	0	0	0
Pro Lite	400	148,99	1787,96	4,47
Pro I	2000	288,77	3465,35	1,73
Pro II	8000	568,33	6819,93	0,85
Pro III	16000	847,88	10174,51	0,63

NAS. Це автономний пристрій для зберігання користувацької інформації, який підключений до мережі за допомогою протоколу Ethernet. Він може зберігати файли та видавати їх за запитом з центрального розташування для авторизованих користувачів, зобразимо приклад такої системи на рис. 1.5 Зручність NAS полягає в гнучкому налаштуванні та контрольованою масштабованість.

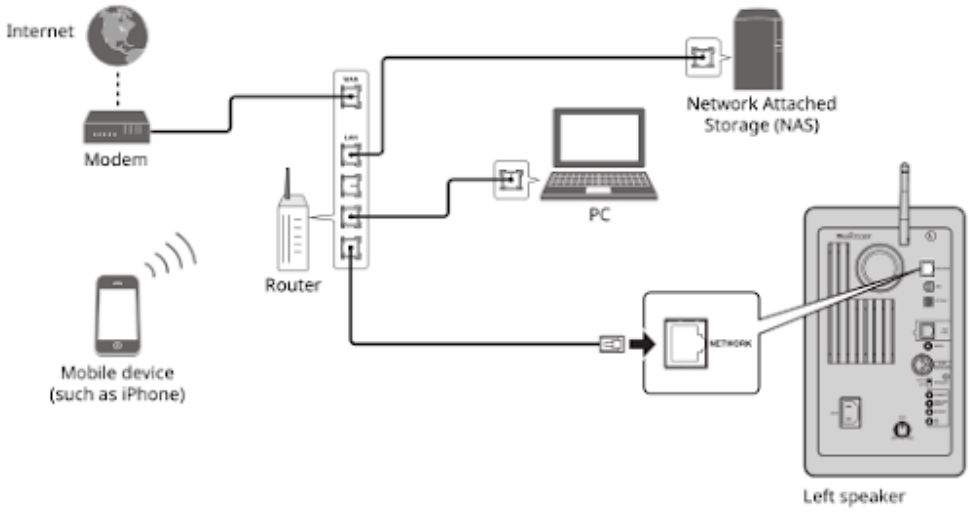


Рисунок 1.5 – NAS в системі

Тома NAS відображаються для користувача як том, підключений до мережі. Файли, які обслуговуються, зазвичай, містяться на одному або декількох дисках зберігання, часто у вигляді логічних надлишкових контейнерів зберігання або як RAID масив.

Сам пристрій являє собою мережевий вузол, дуже схожий на комп'ютер і інші пристрої TCP/IP, які підтримують свої власні IP-адреса і можуть ефективно взаємодіяти з іншими мережевими пристроями.

Хоча NAS зазвичай не призначений для використання в якості сервера загального призначення, постачальники NAS і незалежні постачальники все частіше пропонують програмне забезпечення для забезпечення серверної функціональності на NAS.

Пристрої NAS дозволяють кільком користувачам у різних місцях легко отримувати доступ до даних, що корисно під час спільної роботи над проектами або обміну інформацією. NAS забезпечує хороший контроль доступу та безпеку для підтримки співпраці та дозволяє неспеціалістам IT керувати доступом до даних. Він також забезпечує хорошу базову безпеку даних завдяки використанню надлишкових структур даних (зазвичай RAID).

NAS часто є наступним кроком для домашніх офісів або малих підприємств, які використовують DAS. Перехід до NAS був зумовлений бажанням обмінюватися файлами локально та віддалено, доступністю файлів 24/7, резервуванням даних, можливістю заміни та оновлення жорстких дисків у системі та доступністю інших послуг, таких як автоматичне резервне копіювання.

Перерахуємо переваги NAS:

- відносно дешевий;
- невеликі габарити, відносно серверу;
- цілодобовий доступ до даних;
- віддалений доступ до даних;
- можливість розширення простору;
- можливий перехід на твердотільні накопичувачі;
- обіг інформації без сторонніх сервісів-буферів;
- архітектура резервованого сховища;
- гнучкість конфігурації.

Слабкі сторони випливають з його переваг. Насамперед безпека, захищеність апаратури від стороннього доступу лягає на плечі власника. Він сам повинен забезпечити відповідну шафу з захистом, розеткою мережі та джерелом живлення. Також слабка сторона NAS пов'язана і з продуктивністю, оскільки звертатися до нього може одночасно велика кількість користувачів компанії, може не вистачити пропускної здатної порту Ethernet або ширини каналу наданого постачальником Internet.

Ще один недолік фундаментально пов'язаний з самим стандартом Ethernet. За своїм протоколом Ethernet, розділяє корисне навантаження на відповідні сегменти, формує пакети та передає дані з одного місця в інше, структура Ethernet фрейму представлена на рис. 1.6.

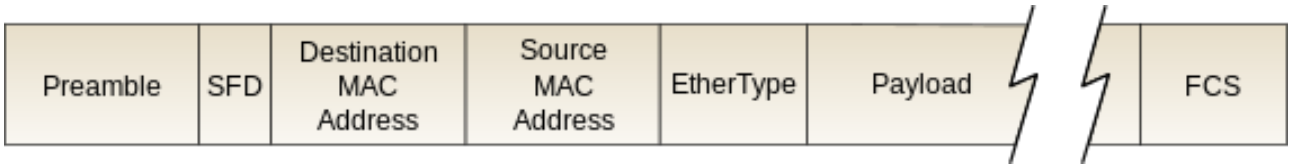


Рисунок 1.6 – Ethernet фрейм

Будь-який, з вище показаних, пакетів може бути затриманий або відправлений не в своєму порядку. В такому випадку, відправлена інформація не буде доступна користувачеві, до поки всі пакети не будуть прийняті відповідно порядку, записаному в стандарті.

Затримки або перезапити пакетів (повільні або повторні підключення) зазвичай проходять не поміченими для користувача, за умови передачі невеликих файлів. Але існують середовища, де часті перезапити або втрата пакетів може сильно вплинути на швидкодію роботи або ж зовсім зупинити її. Для прикладу – рендеринг, затримка більш декількох мілісекунд може зруйнувати весь накопичений прогрес рендерингу та прийдеться починати спочатку..

NAS WD My Cloud Duo. My Cloud Duo, який зображений на рис. 1.7, в декількох проекціях, має в своїй конструкції два порти для роботи жорстких дисків або SSD. Підключення до мережі Internet, відбувається за допомогою витої пари через порт Ethernet маршрутизатора. Власний додаток та візуальна оболонка робить максимально простою взаємодію користувача з хмарною системою для централізованого зберігання цифрових матеріалів.

На відміну від мережевих систем зберігання даних, це просте рішення для зберігання зарезервованих файлів, завдяки фірмовому додатку My Cloud Home можна копіювати, відкривати файли, а також надавати до них спільний доступ через мережу Internet.

Рисунок 1.7 – My Cloud Duo

Будь-який пристрій My Cloud Home містить у своїй будові два порти, де стоять жорсткі диски, які налаштовані в режим дзеркального запису (режим RAID 1). Це гарантує, що всі файли, які користувач завантажує у NAS автоматично копіюються на два диска. Якщо збереження даних не є настільки пріоритетним, в налаштуваннях можна вимкнути функцію RAID та збільшити внутрішній простір в 2 рази. Основні параметри наведемо в табл. 1.5 [5].

Таблиця 1.5 – Основні параметри NAS WD My Cloud Duo

NAS WD My Cloud Duo	
HDD	2 x 3,5 SATA
Raid, рівень	JBOD, Raid 1
Мережеві інтерфейси	1 × Gigabit Ethernet RJ-45
Додаткові інтерфейси	2 × USB Host 3.0
Габарити, мм	179 × 160 × 102
Ціна пристрою, грн	9536
Ціна HDD на 2 tb, грн	2519
Ціна за 1 gb, грн	6,03

NAS Synology DS220+. DS220+, який зображений на рис. 1.8, в декількох проекціях, має в своїй конструкції два порти для роботи жорстких дисків або SSD. Підключення до мережі Internet, відбувається за допомогою витої пари через порт Ethernet маршрутизатора [5].

Рисунок 1.8 – NAS Synology DS220+

Базою для побудови цього NAS лежить однокристальна платформа Intel Gemini Lake Refresh з двоядерний процесором Celeron J4025 з робочою частотою від 2,0 до 2,9 ГГц. Процесор Celeron J4025 має внутрішню графічну карту Intel UHD 600 безпосередньо на кристалі, з можливістю відтворення відеосигналу з роздільною здатністю 4096 × 2160 з частотою 60 Гц. Основні параметри пристрою наведемо в табл. 1.6 [6].

Таблиця 1.6 – Основні параметри NAS Synology DS220+

NAS Synology DS220+	
HDD	2 x 3,5 або 2 x 2,5 SATA
Raid, рівень	JBOD, Raid 0, Raid 1, Basic
Мережеві інтерфейси	2 × Gigabit Ethernet RJ-45
Додаткові інтерфейси	2 × USB Host 3.0
Габарити, мм	165 × 108 × 233
Ціна пристрою, грн	12043
Ціна HDD на 2 tb	2519
Ціна за 1 gb, грн	7,28

NAS має в своїй конструкції два гігабітних мережевих інтерфейса, представленими контролерами RTL8111HS виробництва Realtek Semiconductor Corp. Це рішення дозволяє об'єднувати мережеві підключення завдяки функції Link Aggregation з налаштуванням типу підключення [7].

P2P мережі. Історія піринговых мереж почалася з далекого 1999 року. Цього року стартував проект Napster, який замислювався як всесвітня база обміну файлами. І вже за рік Napster став найпопулярнішим брендом в Інтернеті. Цим сервісом користувалися близько 40 млн. користувачів. І головна причина такої популярності – користувачам за допомогою цієї мережі надавалася можливість безкоштовно завантажувати музичні файли.

Робота пірингової мережі кардинально відрізняється від мереж, побудованих за принципом «клієнт-сервер», ця мережа працює за принципом «клієнт – клієнт», як показано на рис. 1.9. Тобто, якщо говорити правильно, технологія називається Peer to Peer (P2P) – одноранговий вузол мережі до однорангового вузла мережі. Тут комп'ютер кожного користувача виступає одночасно як клієнт і як сервер.

Загальний принцип роботи таких мереж можна описати так, клієнтська програма після підключення до мережі посилає в мережу список файлів, які комп'ютер може надати для скачування іншими клієнтськими програмами і перелік файлів, які клієнтська програма бажає отримати з мережі. Далі відбувається пошук клієнтських програм, яким потрібні пропоновані файли та клієнтських програм, які мають шукані файли. Після цього, клієнтськими програмами встановлюється прямий зв'язок і відбувається обмін даними.

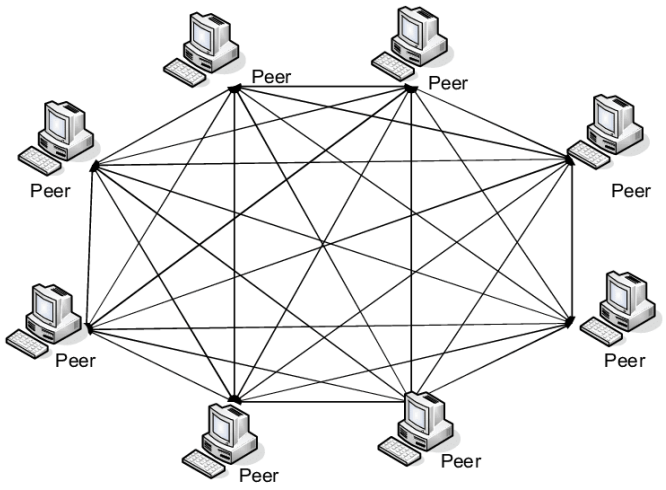


Рисунок 1.9 – структурна схема мережі P2P

Усі файли розбиваються на безліч окремих елементів і клієнтська програма, яка в цей момент часу виконує завантаження потрібних файлів на свій комп'ютер, отримує ці файли частинами від декількох комп'ютерів відразу. І найчастіше комп'ютери-донори, що дають вам частини файлу, знаходяться в різних кінцях світу. Слід врахувати, що клієнтська програма одночасно не тільки отримує частини файлу, але відразу і роздає щойно отримані частини іншим зацікавленим клієнтським програмам.

Існують дві моделі пірінгових мереж – централізовані та децентралізовані мережі. У централізованих мережах пошук відповідних партнерів для обміну даними здійснюється за допомогою центрального сервера, а обмін даними відбувається після того, як між клієнтськими машинами встановиться прямий зв'язок. Якщо будь-які вузли, тобто клієнтські комп'ютери, виконують одночасно функції сервера та клієнта, такі мережі називаються децентралізованими. У таких мережах не використовується центральний сервер для обробки запитів та координації клієнтських машин. Звичайно, централізовані мережі працюють швидше, ніж децентралізовані, але надійність децентралізованих мереж вища, оскільки вони можуть працювати без координуючих серверів.

Ще один позитивний момент – кожен користувач мережі може виставляти на роздачу (так називається процес розповсюдження файлу за протоколом Bittorrent) власні файли. Наприклад, один з працівників компанії, закінчив роботу, яка має складну структуру даних та великий об’єм. Він у своїй програмі-клієнті вказує папку, де на жорсткому диску комп'ютера розташований проект та запускає роздачу. Звичайно, спочатку тільки він буде віддаватимете цей файл в мережу, створюючи навантаження на власну систему, але в міру завантаження файлу на комп'ютери інших працівників компанії, воно буде падати, а швидкість завантаження зростатиме. Файл буде доступний для користувачів, навіть якщо першоджерело вимкне комп’ютер або відімкне його від мережі інтернет.

У пірингових мережах є своя термінологія. Звичайно, користувач, який віддає файл у мережу, буде називатися інакше, ніж користувач, який споживає цей файл. Отже, умовно всіх користувачів можна розділити на сидерів, бенкетів та лікерів.

Сидер або сид (від англ. seeder або seed - сіяч або зерно) це користувач, який має всі частини файлу. Це може бути користувач, який розпочав роздачу, або користувач, який повністю завантажив файл і не вийшов з мережі.

Бенкет (від англ. peer - рівний, співучасник) це користувач, який повністю не скачав весь файл. Але, качаючи на свій комп'ютер усі частини файлу, цей користувач одночасно і бере участь у роздачі, поділяючись з рештою учасників мережі тими частинами файлу, які вже закачав на свій комп'ютер. В принципі, бенкет - будь-який учасник роздачі.

Лічер (від англ. lecher, яке у свою чергу походить від слова leech - п'явка) це користувач, який тільки почав завантажувати файл і не встиг почати роздачу вже отриманих частин.

Природно, весь поділ користувачів на сидерів, бенкетів та лікерів умовний, тому що кожен користувач на початку закачування є лічером, поступово перетворюється на бенкету і в результаті стає сидером. Повернемося до термінології, прийнятої у піринговых мережах.

Доступність – кількість повних копій файлу, доступних клієнту. Кожен сид додає значення 1,0 до цього числа; Лічери збільшують доступність в залежності від кількості скачаного, якого немає в інших лічерів. Наприклад, якщо на роздачі є один сид і два лічера, що завантажили по 50% файлу (завантажені частини рівні між собою), то доступність дорівнює 1,50.

Рой (від англ. swart - рій) - всі користувачі і сидери, що беруть участь у роздачі.

У процесі роботи, скачування та роздачі даних є ще два терміни – анонсування та DHT. Анонсування – процес звернення клієнта до трекера. При кожному анонсі програма-клієнт передає на трекер інформацію про обсяги завантаженого та відданого, а трекер передає клієнту список адрес інших клієнтів. Звернення клієнта до трекера відбувається через певні інтервали часу, які визначаються налаштуваннями клієнта та трекера.

Якщо двома словами, то DHT це протокол, що дозволяє програмам-клієнтам знаходити одне одного без допомоги трекера. Можливість працювати з цими протоколами може бути як дозволена, так і заборонена. Тут нескладно здогадатися, що якщо для певного торрент-файлу DHT заборонено, то без відповідного рівня рейтингу (якщо він нижчий від порогового значення) завантажити файл буде неможливо. Якщо DHT дозволено, то, по суті, рейтинг на трекері значення не має і файл буде доступний до завантаження в будь-якому випадку.

Ethernet. Ethernet - це традиційна технологія, що використовується для підключення пристроїв у провідній локальній мережі (LAN) або глобальній мережі (WAN), що дозволяє їм обмінюватися даними один з одним через протокол (набір правил або спільна мережна мова).

Ethernet описує, як мережеві пристрої форматують та передають дані, щоб інші пристрої в тій же локальній мережі або сегменті мережі могли ідентифікувати, отримувати та обробляти інформацію. Кабель Ethernet - це фізична замкнута проводка, якою проходять дані.

Стандарт Fast Ethernet (IEEE 802.3u) був створений для мереж Ethernet, яким потрібні вищі швидкості передачі. Цей стандарт підвищує обмеження швидкості Ethernet з 10 Мбіт/с до 100 Мбіт/с із мінімальними змінами в існуючій структурі кабелю. Fast Ethernet забезпечує більш високу пропускну здатність для відео, мультимедіа, графіки, перегляду веб-сторінок та більш надійне виявлення та виправлення помилок. Існує три типи Fast Ethernet: 100BASE-TX для використання з кабелем UTP рівня 5; 100BASE-FX для використання з оптоволоконним кабелем; та 100BASE-T4, в якому використовуються два додаткові дроти для використання з кабелем UTP 3-го рівня. Стандарт 100BASE-TX став найпопулярнішим через його тісну сумісність із стандартом 10BASE-T Ethernet.

Gigabit Ethernet, цей тип мережі передає дані з вищою швидкістю, близько 1000 Мбіт/с або 1 Гбіт/с. Гігабітна швидкість – це модернізація Fast Ethernet, від якої поступово відмовляються. У цьому типі мережі всі чотири пари в кабелі скрученої пари роблять свій внесок у швидкість передачі даних. Він знаходить широке застосування у системах відеозв'язку, у яких використовуються кабелі CAT5e чи інші сучасні кабелі. Для розширених мереж на відстані до 500 м можна використовувати оптоволоконні кабелі 1000Base SX для багатомодових систем, а також 1000Base LX для одномодових систем. Найбільш важливі відмінності між Gigabit Ethernet та Fast Ethernet включають додаткову підтримку повнодуплексного режиму на рівні MAC та швидкості передачі даних.

10 Gigabit Ethernet - це найшвидший і останній стандарт Ethernet. IEEE 802.3ae визначає версію Ethernet з номінальною швидкістю 10 Гбіт/с, що робить його у 10 разів швидше, ніж Gigabit Ethernet. На відміну від інших систем Ethernet, 10 Gigabit Ethernet повністю ґрунтується на використанні оптоволоконних з'єднань. У цьому стандарті відбувається перехід від дизайну

ЛОМ, який здійснює широкомовну розсилку на всі вузли, до системи, яка включає деякі елементи глобальної маршрутизації. Він підтримується кабелями скрученої пари CAT6a або CAT7, а також оптоволоконними кабелями. Використовуючи оптоволоконний кабель, ця мережева зона може бути збільшена приблизно до 10 000 метрів.

Комутатор Ethernet: для цього типу мережі потрібний комутатор або концентратор. Також замість кабелю скрученої пари у разі використовується звичайний мережевий кабель. Мережеві комутатори використовуються для передачі даних від одного пристрою до іншого, не перериваючи роботу інших пристроїв у мережі.

Wi-Fi. Це стандарт бездротового підключення LAN для комунікації різних пристроїв, що стосується набору стандартів IEEE 802.11. Бездротова мережа використовує радіохвилі, як це роблять мобільні телефони, телевізори та радіо. Насправді зв’язок через бездротову мережу дуже схожий на двосторонній радіозв’язок [8]. Ось що відбувається:

- бездротовий адаптер комп'ютера перетворює дані в радіосигнал і передає його за допомогою антени;
- бездротовий маршрутизатор приймає сигнал і декодує його. Маршрутизатор надсилає інформацію в Інтернет за допомогою фізичного дротового з’єднання Ethernet;

Процес також працює у зворотному напрямку: маршрутизатор отримує інформацію з Інтернету, перетворює її на радіосигнал і надсилає на бездротовий адаптер комп’ютера.

Радіостанції, які використовуються для зв’язку по Wi-Fi, дуже схожі на радіостанції, які використовуються для рацій, мобільних телефонів та інших пристроїв. Вони можуть передавати і приймати радіохвилі, а також можуть перетворювати 1s і 0s в радіохвилі і перетворювати радіохвилі назад в 1s і 0s. Але радіостанції WiFi мають кілька помітних відмінностей від інших радіостанцій:

- вони передають на частотах 2,4 ГГц або 5 ГГц. Ця частота значно вища за частоти мобільних телефонів, рацій і телевізорів. Вища частота дозволяє сигналу передавати більше даних.
- З’єднання на частоті 2,4 ГГц зараз вважаються дещо застарілими, оскільки вони передають дані на нижчій швидкості, ніж 5 ГГц. Однак діапазон 2.4 продовжує використовуватися, оскільки нижча частота може

переноситись на 30 метрів. В ідеальних умовах діапазон 5 ГГц має максимальний радіус дії приблизно 61 метр, але в реальному світі він набагато більш схильний до перешкод від стін, дверей та інших об’єктів. Діапазон 2,4 може бути швидшим для користувача, який підключається до маршрутизатора за кілька кімнат, тоді як 5 ГГц точно буде швидшим для тісного з’єднання.

Wi-Fi використовує мережеві стандарти 802.11, які мають кілька варіантів і розвивалися протягом десятиліть:

802.11b (введений у 1999 році) є найповільнішим і найменш дорогим стандартом. Деякий час його вартість зробила його популярним, але зараз він менш поширений, оскільки швидші стандарти стають дешевшими. 802.11b передає в діапазоні частот 2,4 ГГц радіоспектру. Він може обробляти до 11 мегабіт даних за секунду та використовує модуляцію додаткового коду (CCK) для підвищення швидкості.

802.11a (введений після 802.11b) передає на частоті 5 ГГц і може передавати до 54 мегабіт даних на секунду. Він використовує мультиплексування з ортогональним частотним поділом (OFDM), більш ефективну техніку кодування, яка розділяє цей радіосигнал на кілька підсигналів, перш ніж вони досягнуть приймача. Це значно зменшує перешкоди.

802.11g передає на частоті 2,4 ГГц, як і 802.11b, але він набагато швидший – він може обробляти до 54 мегабіт даних на секунду. 802.11g є швидшим, оскільки він використовує те саме кодування OFDM, що й 802.11a.

802.11n (введений у 2009 році) має зворотну сумісність з a, b і g. Він значно покращив швидкість і радіус дії порівняно зі своїми попередниками. Наприклад, хоча 802.11g теоретично передає 54 мегабіта даних за секунду, він досягає реальної швидкості лише близько 24 мегабіт даних за секунду через перевантаження мережі. Однак, як повідомляється, 802.11n може досягати швидкості до 140 мегабіт на секунду. 802.11n може передавати до чотирьох потоків даних, кожен із максимальною швидкістю 150 мегабіт на секунду, але більшість маршрутизаторів дозволяють лише два або три потоки.

802.11ac з’явився приблизно в 2014 році і працює виключно на частоті 5 ГГц. 802.11ac зворотно сумісний з 802.11n (і, отже, з іншими також), з n на діапазоні 2,4 ГГц і ac на діапазоні 5 ГГц. Він менш схильний до перешкод і набагато швидший за своїх попередників, пропускаючи максимум 450 мегабіт на

секунду в одному потоці, хоча в реальних умовах швидкість може бути нижчою. Як і 802.11n, він дозволяє передавати кілька просторових потоків — до восьми, за бажанням. Іноді його називають 5G через його частотний діапазон, іноді Gigabit WiFi через його потенціал перевищувати гігабіт за секунду на кількох потоках, а іноді дуже високу пропускну здатність (VHT) з тієї ж причини.

802.11ax, також відомий як WiFi 6, з’явився в галузі в 2019 році. Цей стандарт розширює можливості 802.11ac кількома ключовими способами. Перш за все, нові маршрутизатори забезпечують ще вищу швидкість потоку даних до 9,2 Гбіт/с (гігабіт на секунду). WiFi 6 також дозволяє виробникам встановлювати більше антен на один маршрутизатор, приймаючи кілька з’єднань одночасно, не турбуючись про перешкоди та сповільнення. Деякі нові пристрої також підключаються до вищого діапазону 6 ГГц, що приблизно на 20 відсотків швидше, ніж 5 ГГц в ідеальних умовах.

Очікується, що 802.11be (або WiFi 7) стане стандартом до 2024 року та має запропонувати ще кращий радіус дії, більше з’єднань і вищу швидкість передачі даних, ніж будь-яка з попередніх версій.

Інші стандарти 802.11 зосереджені на конкретних застосуваннях бездротових мереж, як-от глобальні мережі у транспортних засобах або технологіях, які дозволяють легко переходити від однієї бездротової мережі до іншої.

Радіостанції WiFi можуть передавати на будь-якому діапазоні частот. Або вони можуть швидко «стрибати» між різними діапазонами. Стрибкова зміна частоти допомагає зменшити перешкоди та дозволяє кільком пристроям використовувати одне бездротове з’єднання одночасно.

Якщо всі вони мають бездротові адаптери, кілька пристроїв можуть використовувати один маршрутизатор для підключення до Інтернету. Це з'єднання зручне, практично непомітне і досить надійне; однак, якщо маршрутизатор виходить з ладу або якщо занадто багато людей намагаються одночасно використовувати програми з високою пропускною здатністю, користувачі можуть відчувати перешкоди або втрачати з’єднання, хоча нові, швидші стандарти, такі як 802.11ax, допоможуть у цьому.

Радіо-модеми. З розвитком мереж передачі даних, зростали вимоги до пропускної спроможності каналів зв'язку, зокрема і радіоканалів. Радіообладнання розвивалося в основному у двох напрямках: адаптація

радіотелефонних каналів до передачі даних та збільшення розмірів локальних обчислювальних мереж (за рахунок використання радіосистем), які принципово відрізняються типами використовуваних каналів – синхронним та асинхронним відповідно [9].

Останнім часом при створенні бездротових мереж передачі даних найбільшого поширення набули пристрої на базі технології RadioEthernet, яка орієнтована на забезпечення радіо-доступу до локальних обчислювальних мереж. Зазвичай, таке завдання виникає при підключенні до комп'ютерної мережі офісу мобільних абонентів. Зараз ця технологія дуже популярна при розгортанні великомасштабних корпоративних мереж та організації доступу до Інтернету. Причина цього – простота та дешевизна стикування обладнання RadioEthernet із звичайними комп'ютерними мережами, оскільки для цього не потрібно купувати та встановлювати дорогі мережні узгоджувальні пристрої (перетворювачі інтерфейсів та протоколів, потужні маршрутизатори тощо). Однак через труднощі, пов'язані із забезпеченням необхідної якості обслуговування, та нестачі пропускної спроможності мережевих каналів технологія Ethernet не завжди підходить для передачі трафіку мультимедіа. Крім того, вона є неефективною для організації великомасштабних мереж.

Вирішенням проблеми побудови таких мереж сьогодні є технологія Frame Relay, яка ґрунтується на використанні синхронних високопродуктивних каналів, до яких належать і радіо-модемні канали.

На відміну від радіорелейних систем, що працюють у різних діапазонах - від дециметрового до міліметрового, практично всі існуючі радіо-модеми функціонують у діапазонах, що не ліцензуються в більшості країн світу, виділених для промислового, наукового та медичного обладнання (Industrial, Scientific and Medical bands — ISM): 902—928 МГц, 2,4—2,4835 ГГц і 5,725—5,85 ГГц. Можливість вільного використання діапазонів ISM багато в чому визначила широку популярність радіомодемів у всьому світі.

Провести межу між радіо-модемами, пристроями RadioEthernet та радіорелейними системами досить складно. І все ж, спираючись на підходи, що вже склалися, можна сказати, що радіо-модем це радіотехнічний пристрій, призначений для передачі синхронних цифрових потоків даних по радіоканалу ISM-діапазону без використання спеціалізованих протоколів доступу до середовища передачі даних.

На відміну від радіо-модемів, професійні радіорелейні системи зазвичай не працюють у діапазонах ISM, а радіо-мости та інші пристрої RadioEthernet (точки доступу, бездротові мережеві адаптери та ін.) забезпечують пакетну передачу даних з використанням спеціально розроблених протоколів доступу до середовища передачі, що необхідно для запобігання колізіям пакетів.

Рисунок 1.10 – Радіо-модем на 435МГц

Крім перерахованих у визначенні характеристик, радіо-модеми мають інші важливі властивості, а саме: підтримка технології розширення спектра сигналу; високе значення показника ефективність/вартість; простота установки, що не вимагає від особливих професійних навичок; малі габаритні розміри, маса та енергоспоживання; переважне використання для створення однопрогонових радіоліній топології «P2P».

1.2 Постановка завдання на проектування

Метою кваліфікаційної роботи є розробка системи мультисервісного доступу до корпоративного хмарного сховища.

Система корпоративного хмарного сховища повинна мати невеликі габарити та складатися з мінімально можливої кількості модулів та без декоративного, пластикового корпусу, за для здешевлення конструкції та покращення охолодження, але без втрати функціональності. Умова, про невеликі

розміри вимушена тим, що пристрої такого призначення в компаніях, зазвичай монтують в захищених шафах, до якої має доступ обмежена кількість людей. Пристрій повинен мати мінімум дротів для підключення.

Система повинна виконувати наступні функції:

- завантаження через мережу файлів різних форматів;
- доступ до раніше завантажених файлів через локальну мережу;
- доступ до раніше завантажених файлів через мережу інтернет;
- редагування завантажених файлів в локальної хмарі;
- реалізація різних прав доступу до файлів;
- забезпечення збереження даних за технологією RAID 1;
- захист даних в сховищі;

Для виконання системою всіх покладених на неї завдань, необхідно:

- розглянути теперішній рівень техніки та конкурентів;
- розглянути методи захисту даних в пристроях цієї області;
- розробити алгоритм функціонування пристрою;
- розробити схему електричну структурну;
- розробити схему електричну функціональну пристрою;
- розробити схему електричну принципову;
- виконати підбір елементної бази;
- розрахувати економічну частину проекту.

2 НАУКОВО-ДОСЛІДНА ЧАСТИНА

Зараз багато користувачів використовують зашифрований канал для зв'язку з віддаленим сервером ресурсів. Такий канал забезпечує високий ступінь приватності та конфіденційності. Secure Shell (SSH) є одним із найпоширеніших методів віддаленого підключення до сервера. SSH забезпечує приватність і конфіденційність шляхом шифрування мережевого трафіку між клієнтом і сервером. Завдяки шифруванню процес вивчення зловмисних дій через SSH є складним, особливо завдяки простому аналізу мережевого трафіку. Щоб подолати проблему, ми можемо використати інтроспекцію віртуальної машини. VMI дозволяє прямий доступ до пам'яті віртуальної машини, включаючи доступ до даних процесу SSH.

Однак поточний прототип страждає від великих накладних витрат, оскільки він витягує кожне корисне навантаження мережі SSH у вигляді звичайного тексту з пам'яті, а процес вилучення вимагає миттєвої призупинення віртуальної машини. У науково-дослідній частині, ми розглянемо інструмент SSHkex, який також використовує VMI для вилучення ключів сеансу SSH із пам'яті сервера, за авторством Stewart Sentanoe та Hans P. Reiser. Їх підхід вимагає лише двічі призупиняти віртуальну машину, щоб отримати ключі сеансу для кожного сеансу SSH та виконує пасивний моніторинг мережі, де це не має помітного впливу на поточне з'єднання. Щоб використовувати SSHkex, на сервері не потрібно вносити жодних змін. Таким чином, він підходить для систем виявлення вторгнень і приманки з високою взаємодією, де сервер не потрібно змінювати.

Вступ. Зашифроване спілкування стало де-факто стандартом в інтернеті. Ця розробка має значні переваги з точки зору IT-безпеки, але створює нові проблеми для цифрової криміналістики. SSH це мережевий протокол, який забезпечує безпечний віддалений доступ до комп'ютерів. Він підтримує такі програми, як вхід до командного рядка, віддалене виконання команд, безпечна передача файлів, а також довільне тунелювання мережевого трафіку. Цей широкий спектр застосування робить його універсальним інструментом, який часто використовується в інтернеті.

Однак SSH також відіграє важливу роль у контексті шкідливих дій в інтернеті. Служба SSH може бути ціллю та точкою входу для зловмисників.

Погано налаштовані сервери можуть стати жертвами атак підбору пароля. Часто навіть можна знайти пристрої, розгорнуті в загальнодоступному інтернеті, із стандартними обліковими даними для адміністративного входу. Зловмисники також можуть використовувати SSH для передачі корисного навантаження другого етапу атаки або взаємодії з віддаленими серверами керування. Він також може бути важливою частиною високо взаємодіючої приманки та потенційно корисним у автоматизованій системі виявлення вторгнень.

Існує кілька потенційних підходів для отримання розшифрованого трафіку SSH. Найбільш відомими методами є man-in-the-middle (MITM) та бінарна маніпуляція. Проте кожен із цих двох підходів має певні недоліки та обмеження.

MITM базується на проксі-сервері між клієнтом і сервером [10].Сервер MITM має розшифрувати та повторно зашифрувати кожен пакет SSH, який передається між клієнтом і сервером. Його легко розгорнути, але він також має серйозний недолік, користувач може легко визначити вторгнення. SSH містить механізм для перевірки ключів хоста і якщо хтось підключається до проксі-сервера MITM замість реального віддаленого сервера, інший ключ хоста легко виявити.

Підхід бінарних маніпуляцій базується на модифікаціях використовуваної реалізації SSH-сервера (та/або клієнта). З додаванням коду, який вилучає всі розшифровані пакети даних, можна створити повний дамп зв’язку. Хоча таку модифікацію можна виконати лише за допомогою кількох рядків коду у випадку реалізації SSH з відкритим кодом, складною частиною є розгортання. Змінений двійковий файл SSH потрібно розгорнути на досліджуваній машині, що є нав’язливою операцією та потребує привілейованого доступу до цільової системи.

Прототип SSHkex виявляє встановлення з’єднань SSH, витягує інформацію про те, які криптографічні алгоритми використовує SSH, а потім використовує VMI для читання пам’яті віртуальної машини. Щоб отримати ключі сеансу SSH, SSHkex використовує знання про реалізацію SSH і може безпосередньо визначити розташування ключів SSH у пам’яті та ефективно їх витягувати.

Цей підхід має низку переваг, він не перехоплює комунікацію потенційно виявленим способом. Він також не вимагає двійкових модифікацій служби SSH у цільовій системі. Цей підхід створює невеликі накладні витрати на сеанс SSH.

Оцінка продуктивності показує, що SSHkex створює додаткову затримку менше ніж 100 мс підключення SSH і займає невелику кількість часу для розшифровки мережевого пакета. Цей підхід підходить для розширених систем виявлення вторгнень і SSH-приманок із високою взаємодією, де не потрібно змінювати сервер.

SSH. Це мережевий протокол для встановлення безпечних віддалених з'єднань через незахищену мережу. SSH був розроблений для заміни telnet і незахищених віддалених протоколів оболонки, таких як rlogin, rsh і rexec.

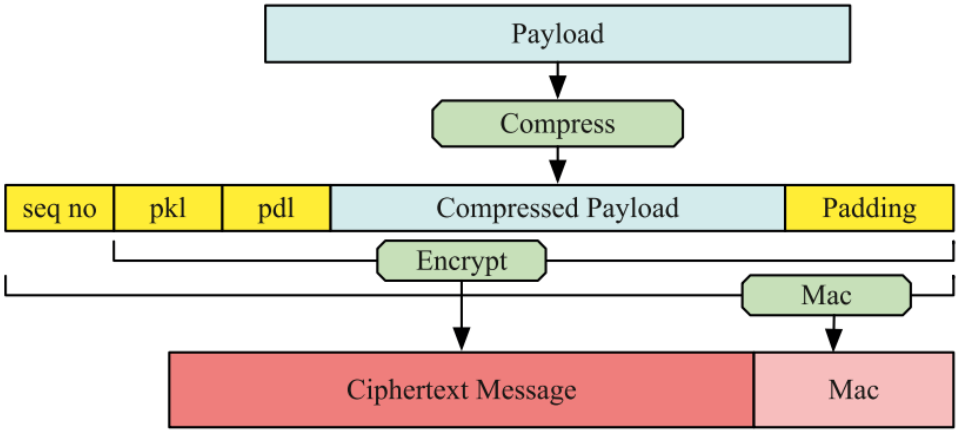


Рисунок 2.1 – Макет пакетів транспортного протоколу SSH

Коли з'єднання SSH встановлено, клієнт і сервер почнуть обмінюватися даними через з'єднання TCP. Макет пакетів транспортного протоколу SSH, показаний на рис. 2.1. Кожне корисне повідомлення спочатку стискається, а потім об'єднується з такими елементами:

- packet length (PKL) містить довжину самого пакета, не включаючи поле довжини пакета або поле коду автентифікації повідомлення (MAC);
- padding length (PDL) містить довжину поля заповнення;
- стиснуте корисне навантаження містить фактичні дані корисного навантаження пакета;
- містить випадкові байти доповнення, щоб загальна довжина пакета (за винятком MAC) була кратною розміру блоку шифру або 8 байтів для потокового шифру;
- message authentication code (MAC) містить значення, якщо його користування домовлено. Він обчислюється на основі неявного порядкового номеру та всіх інших незашифрованих елементів.

Обмін ключами SSH і ключами сесії. Процедура обміну ключами SSH визначена в RFC 4253 [11]. В результаті обміну ключами сервер і клієнт отримали головний ключ «K» і хеш-значення «h». Клієнт може перевірити автентичність сервера за допомогою відкритого ключа сервера, оскільки обмін повідомленнями підписується сервером за допомогою його закритого ключа.

Однак сервер не може зробити висновок про автентичність клієнта. Щоб подолати цю проблему, використовується протокол автентифікації SSH, який підтримує кілька методів автентифікації, таких як автентифікація за паролем і автентифікація за відкритим ключем.

Значення «h» першого типу обміну ключами служить ідентифікатором сеансу для цього підключення. Пізніші операції повторного введення ключа можуть оновити матеріал ключа, але збережуть ідентифікатор сеансу. На етапі обміну ключами значення «K», «h» та ідентифікатор сеансу (позначений session_id нижче) використовуються для отримання ключів сеансу SSH. Ці ключі використовуватимуться для шифрування мережевого трафіку. Ключі обчислюються наступним чином [11]:

1) ключ A (вектор ініціалізації, клієнт-сервер) $IV_{client2server}$ = Hash (K, h, “A”, session_id);
2) ключ B (вектор ініціалізації, клієнт-сервер) $IV_{client2server}$ = Hash (K, h, “B”, session_id);
3) ключ C (ключ шифрування, клієнт-сервер) $EK_{client2server}$ = Hash (K, h, “C”, session_id);
4) ключ D (ключ шифрування, клієнт-сервер) $EK_{client2server}$ = Hash (K, h, “D”, session_id);
5) ключ E (ключ цілісності, клієнт-сервер) $IK_{client2server}$ = Hash (K, h, “E”, session_id);
6) ключ F (ключ цілісності, клієнт-сервер) $IK_{client2server}$ = Hash (K, h, “F”, session_id).

Таким чином, для розшифровки зашифрованого зв’язку SSH необхідно дізнатися про значення, які використовуються в обчисленні ключа сеансу (K, h та session_id), або безпосередньо ключі шифрування для вхідних та вихідних даних.

OpenSSH. Це вільно-доступна та широко використовувана програмна реалізація протоколу SSH. OpenSSH включає серверний компонент SSHD та

клієнтські інструменти, зокрема, віддалений клієнт оболонки SSH і безпечну передачу файлів SCP.

OpenSSH використовував розділення привілеїв, щоб помилка в непривілейованому дочірньому процесі не призводила до злому системи [12].

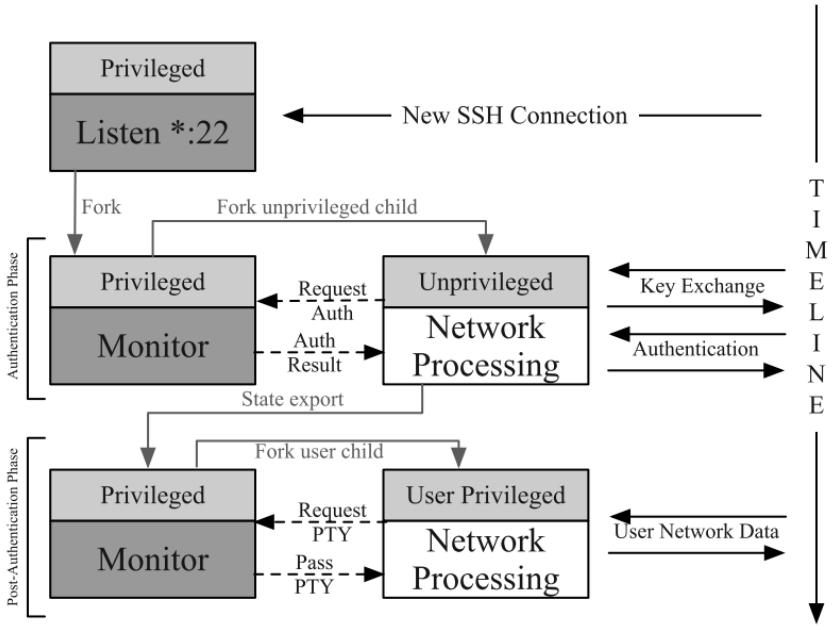


Рисунок 2.2 – Потік поділу привілеїв OpenSSH.

Розділення привілеїв використовує два процеси:

- привілейований батьківський процес стежить за перебігом непривілейованого дочірнього процесу, як показано на рис. 2.2 та рис. 2.3(A);
- непривілейований дочірній процес має обмежений доступ до файлової системи та обробляє всі мережеві дані, що надходять від клієнта під час автентифікації.

А

```
root  /usr/sbin/sshd -D
root          \_ sshd: user1 [priv]
sshd          \_ sshd: user1 [net]
```

Б

```
root /usr/sbin/sshd -D
root  \_ sshd: user1 [priv]
user1       \_ sshd: user1@pts/2
user1             \_ -zsh
```

Рисунок 2.3 – Процеси OpenSSH, (A) під час фази автентифікації, (Б) після успішної фази автентифікації

Батьківський процес вирішить, успішна автентифікація чи ні. Коли користувач автентифікований, привілейований батьківський пристрій (монітор) запускає новий процес, який змінює його ідентифікатор користувача та привілеї на ідентифікатор автентифікованого користувача [9], як показано на рис. 2.3(Б). Ці знання є важливими для підходу, представленого в цій статті, оскільки ми витягуємо дані з пам’яті, які можуть існувати лише всередині привілейованого процесу, а не в непривілейованому процесі або навпаки.

VMI. Самоаналіз віртуальної машини це процес перевірки та моніторингу віртуальної машини з точки зору гіпервізора. Гіпервізор дозволяє гостьовій операційній системі запускати та підтримувати контроль над ресурсами [13]. Така архітектура дозволяє гіпервізору мати повний і чистий перегляд усієї інформації про стан гостьової віртуальної машини. Завдяки такій можливості гіпервізор може спостерігати та аналізувати гостьову операційну систему ззовні [14]. Існує кілька способів виявити дії на віртуальній машині за допомогою VMI. Два з них це моніторинг виконання системних викликів [15] і моніторинг викликів функцій. Перехоплюючи системні виклики або виклики функцій і аналізуючи параметри, ми можемо отримати огляд того, що відбувається всередині віртуальної машини.

Цілі та припущення SSHkex. Основною метою SSHkex є використання стандартних підходів для захоплення мережевого трафіку та їх поєднання з динамічним вилученням ключа сеансу SSH з основної пам’яті за допомогою інтроспекції віртуальної машини.

Перше припущення полягає в тому, що існують засоби для пасивного моніторингу мережевого трафіку. Весь мережевий трафік зв'язку з сервером SSH може бути захоплений у стандартному форматі PCAP для подальшої обробки. Це припущення може легко виконати практично будь-яка система.

Друге припущення суворіше, припускаємо, що знаємо, яка реалізація SSH запущена на сервері. Зокрема, припускаємо, що маємо інформацію про те, яка версія реалізації SSH використовується в нашій цільовій системі, і ми припускаємо, що у нас є доступна інформація про системи налагодження цієї реалізації. Друге припущення зазвичай задовольняється, якщо цільова система використовує готову версію програмного забезпечення SSH зі стандартного дистрибутива Linux. У випадку спеціального, можливо модифікованого програмного забезпечення SSH у цільовій системі, потрібен певний процес для

передачі необхідної інформації символів налагодження нашому програмному забезпеченню. Однак основна увага зосереджена на цільових системах, які використовують добре відоме стандартне програмне забезпечення. Таким чином, це дійсне припущення в більшості реалістичних випадків використання.

Третє припущення полягає в тому, що маємо функціональні можливості VMI для доступу до цільових систем. Прототип реалізації базується на відомому LibVMI, який, підтримує самоаналіз на віртуальних машинах на гіпервізорі Xen і KVM.

На рис. 2.4, зображений високорівневий дизайн SSHkex.

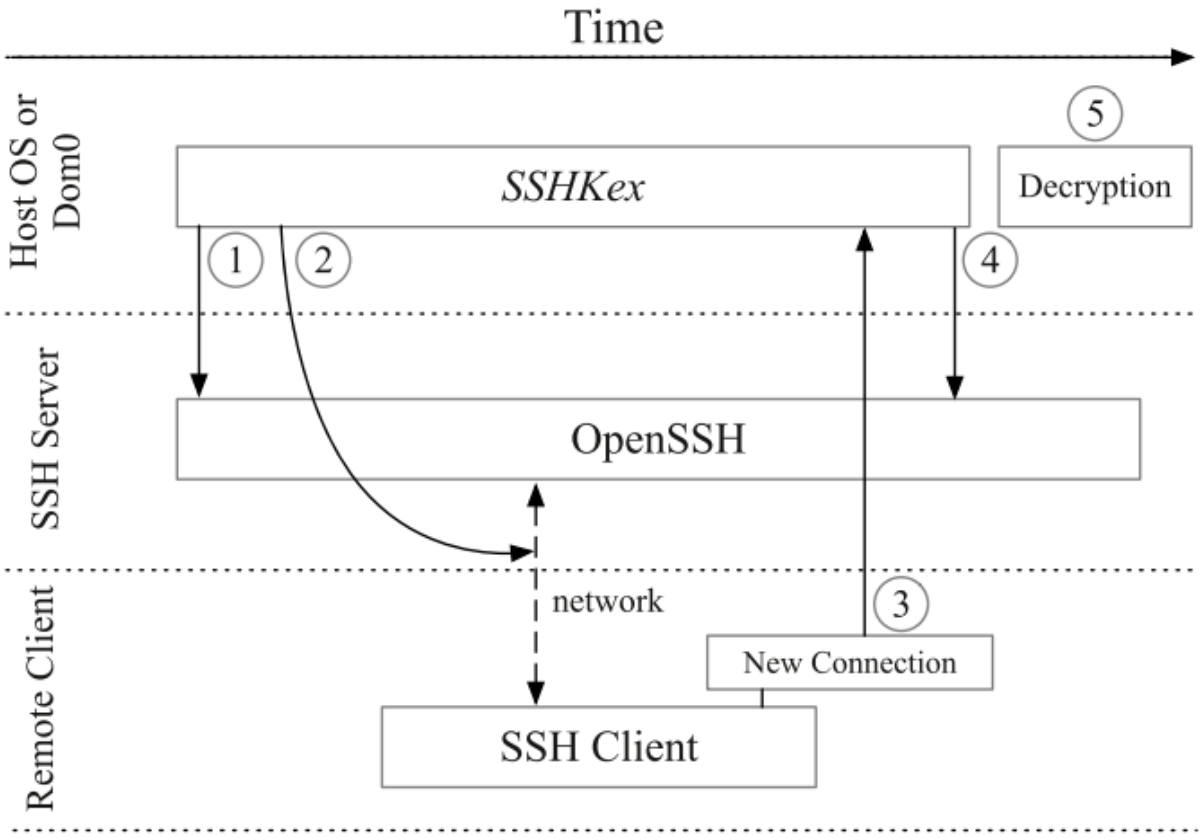


Рисунок 2.4 – Високорівневий дизайн SSHkex

Основними компонентами SSHkex є :

- setup. Відповідає за налаштування моніторингу процесів SSH у цільовій віртуальній машині за допомогою активного VMI;
- network logging. Основою аналізу є пряме захоплення мережевого трафіку. ми можемо використовувати будь-які стандартні інструменти, які записують трафік у файли PCAP;

– key capture trigger. Ми повинні витягти ключі в потрібний момент часу. Лише через короткий проміжок часу після початкового з'єднання мережевого рівня матеріал ключа буде згенерований і буде присутній в основній пам'яті, а після закриття з'єднання матеріал ключа може бути швидко очищений з основної пам'яті. У SSHkex використовується трасування потоку керування процесом SSH, щоб отримати інформацію про розташування пам'яті, яка використовуються для зберігання ключів сеансу SSH і визначити, коли було завершено створення ключа;
– key extraction. На основі інформації, зібраної на попередньому кроці, ми можемо безпосередньо витягнути матеріал ключа SSH з основної пам'яті та зберегти його у файлі для подальшого розшифровки мережевого трафіку;
– Decryption. Розшифровка виконується окремо.

Мета SSHkex полягає не тільки в тому, щоб знайти ключовий матеріал в основній пам'яті, але й зробити це швидко та ефективно. Для цього використовуються знання про структури даних, які використовуються для зберігання ключових матеріалів. Припускається, що є доступна інформація про символи налагодження, яка відповідає версії реалізації SSH у цільовій системі. Використовується libdwarfparser [16], щоб отримати інформацію про структури даних і отримати значення зміщення конкретних елементів структури даних у структурі.

Прототип SSHkex був розроблений і протестований з OpenSSH. Для цієї реалізації наступні структури даних представляють особливий інтерес:

1) struct ssh оголошується у файлі packet.h і містить віддалену IP-адресу та порт (ім'я змінної: remote_ipaddr і remote_port), локальну IP-адресу та порт (ім'я змінної: local_ipaddr і local_port), адреса пам'яті session_state і sshbuf;
2) struct session_state оголошено у файлі packet.c і містить ключі сеансу (ім'я змінної: newkeys) для конкретного сеансу SSH;
3) структура newkeys оголошена у файлі kex.h і містить структуру sshenc, яка містить інформацію про ключі шифрування;
4) Структура sshenc оголошена у файлі kex.h і містить метод шифрування (ім'я змінної: name), вектор ініціалізації (ім'я змінної: iv) і ключі шифрування (ім'я змінної: key).

Зробимо зауваження, що інформація про тип, структуру даних не повідомляє нам безпосередньо, де в основній пам'яті знайти потрібну інформацію. Він повідомляє нам лише про те, де розташовані зацікавлені елементи відносно початку структури даних.

Відстеження функцій OpenSSH. Для ефективного визначення місцезнаходження структур даних і вилучення ключів у потрібний момент часу ми використовуємо відстеження двох функцій OpenSSH, які використовуються непривілейованим процесом sshd (рис. 2.5), який обробляє автентифікацію клієнта та генерацію ключів:

- kex_derive_keys. Функція оголошена у файлі kex.c. Функція викликається, коли починається процес генерації сеансових ключів. Вхідними параметрами є адреса пам’яті структури ssh, хеш «h», довжина хешу та адреса sshbuf, які містять спільний секрет «K»;
- do_authentication2. Функція оголошена у файлі auth2.c. Функція викликається, коли незабаром розпочнеться фаза автентифікації користувача. Вхідним параметром є адреса пам’яті struct Authctxt.

Впровадження точок зупину та вилучення ключів. Точка зупину програмного забезпечення це навмисне призупинення виконання програми в певній точці потоку виконання, яке можна використовувати для дослідження стану програми з метою налагодження [17]. Підхід розробників, використовувати точки зупину, розміщені на початку функцій, щоб реалізувати трасування функцій. Встановлюється контрольні точки за допомогою VMI і для цього потрібна інформація про те, де в пам’яті розташовані дві відповідні функції. Цю інформацію можна отримати з символів налагодження OpenSSH, отриманих із сховища пакетів використовуваного дистрибутива Linux.

Вилучення ключа. Коли викликається функція kex_derive_keys, звідси витягується та зберігається адреса структури ssh. Потім процес продовжуватиметься, доки не буде викликана функція do_authentication2. Далі витягуються ключі з пам’яті, дотримуючись структури, як показано на рис. 2.6. OpenSSH зберігає ключі client2server і server2client за різними індексами нових ключів (індекси 0 і 1 відповідно). Ключі A та C зберігаються в індексі 0 у полях iv та key відповідно. Ключі B і D зберігаються в індексі 1, також у полях iv і key. Прототип також витягує ключі E і F, але наразі ці ключі не використовуються для подальших кроків. OpenSSH також має функцію повторного введення

ключів, коли новий набір ключів генерується та використовується в середині тривалого сеансу. Ця функція детально не розглядається цю функцію, але щоб впоратися з цим, потрібно перехопити функцію, яка відповідає за повторне введення ключа. Щоб обробляти кілька з'єднань, кожен ключ витягується на основі власного ідентифікатора процесу, оскільки OpenSSH створює дочірній процес для обробки кожного з'єднання.

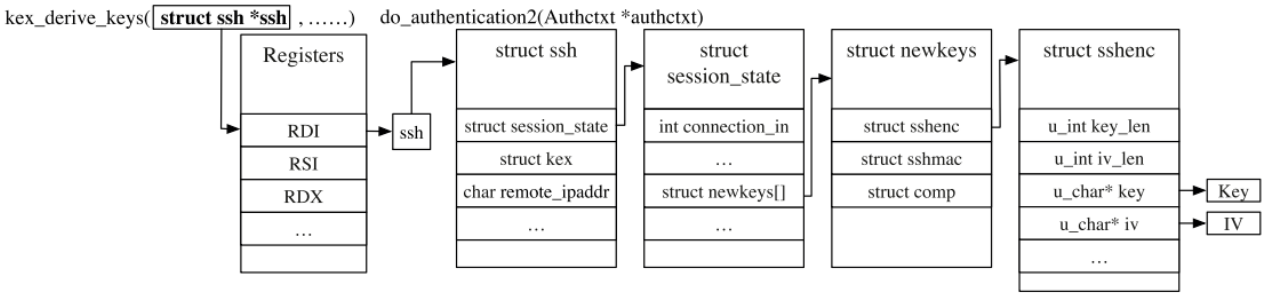


Рисунок 2.6 – Потік вилучення ключів OpenSSH

Дешифрування. Створено два інструменти дешифрування для цього дослідження. Перший на основі Python з бібліотекою Pycryptodome, а другий є плагіном Wireshark.

Інструмент на основі Python служить доказом концепції та перевіряє правильні ключі. Для його використання користувачеві необхідно вручну ввести ключі та зашифрований текст. Потім програма надрукує звичайний текст.

Для плагіна Wireshark спочатку користувач має встановити плагін. По-друге, завантажте файл PCAP або виконати пошук мережі в реальному часі. По-третє, вставити ключі сеансу SSH. Нарешті, плагін розшифрує та надрукує звичайний текст. Wireshark використовує джерела даних для кожного захопленого пакета, у якому зберігаються відповідні дані (зашифровані дані пакета, довжина, інформація заголовка).

Експериментальна установка. Для цього експеремента був використаний Xen 4.11 як гіпервізор. В процесі експеремента було використано дві віртуальні машини Ubuntu з 256 МБ оперативної пам'яті та 1 vCPU, під керуванням OpenSSH 7.2p2. Одна віртуальна машина діяла як сервер SSH, який активно контролюється SSHkex (як показано на рис. 2.7). Моніторинг мережі виконується на хост-системі, яка має доступ до мережевого інтерфейсу віртуальної машини.

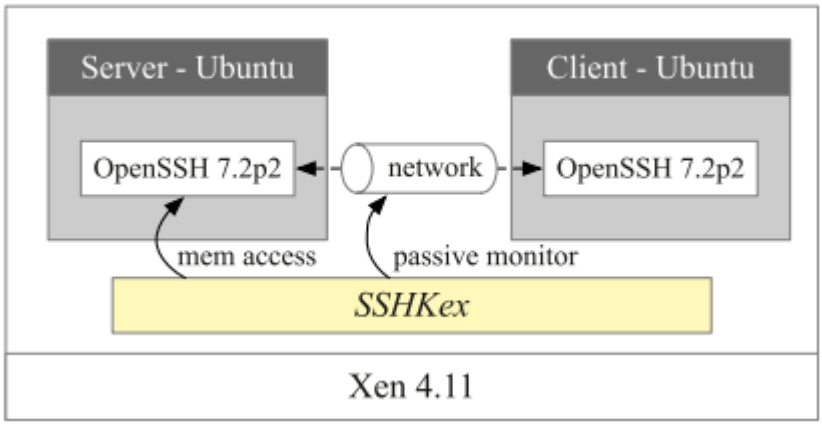


Рисунок 2.7 – Експериментальне встановлення

Лістинг, який зображений на рис. 2.7, показує витягнуті ключі ChaCha20-Poly1305. Система SSHkex витягує назву алгоритму шифрування та всі ключі (A-F) у шістнадцятковому форматі.

```
Key 1 name : chacha20-poly1305@openssh.com
Key 2 name : chacha20-poly1305@openssh.com

key : A -- addr : 5588ac9fcde0 len : 64
93D20734C5F83B5BC81DD1816A042BB4624D1296BC1F8E
93FF02EA133CEC2F6B07922928ABEE64EC59DFC4BB11301
7284D6CAC9548BE9A8C6A9AC4EFB0D0035B
key : B -- addr : 5588ac9f8ac0 len : 64
A196F1B4A72308A8EC41AC72A84FA222CD699EF83F3CDE
7789AB37B47113CED57A67396EF07ACA52FE2B5D96F2E3
D5E3AD982B6F32850472AA4B29C1221CDCD4
key : C -- addr : 5588ac9fed90 len : 64
0B77E10E49CAB165B93CF0C861FEB38C097F3F8F8E6AA7
FFF4D979ABC5F3164A8136FE3C48DFE86CF59710BA7998
FA996B40AC21B50E5B2F6A799F08C7C3E183
key : D -- addr : 5588ac9fa810 len : 64
8F84E1EEFD3E09A4F8A338DDF7E47A59808B0C96020595
DA70F073A5830A7ABE666C4FB6BE2F60DD837C63E2DC8
766B6BF62DFBD77BDA98BD1FCEB791E9442B8
key : E -- addr : 5588ac9fce30 len : 64
D446F576112F2B09C6758D1A8ABD70ED4B945A8298CD6
7F8102ABDED23779132C4AC8CB1483C2D22533AC4BE1B
33F838B97FBBA624900A13D19063C99F8E8693
key : F -- addr : 5588ac9fbd00 len : 64
4FFE294F171DBAD11CA45B289BD2E27C3E48BEF73E1D1
C503CDF726BA0071998C71AD9A2066D2613B130CD3122
6F6FDF927BA38202BCCDFE6D25ED380644186C
```

Рисунок 2.8 – Витягнуті ключі Chacha20-Poly1305

Розшифрований пакет tcpdump. Використовуючи tcpdump для захоплення мережевого трафіку та інструмент дешифрування на основі Python, можна розшифрувати мережеві пакети SSH за допомогою вилучених ключів сеансу. Лістинг, який зображений на рис. 2.9, показує зашифрований (за допомогою Chacha20-Poly1305) пакет SSH, а лістинг, який зображений на рис. 2.10, показує розшифрований пакет, який все ще має значення довжини заповнення.

```
09:55:19 IP 192.168.12.46.ssh > 192.168.12.29.60660:
tcp 76
0x0000:  4548 0080 bc22 4000 4006 e471 c0a8 0c2e
0x0010:  c0a8 0c1d 0016 ecf4 5a30 7e13 4bac 0d86
0x0020:  8018 0487 9a0e 0000 0101 080a 22c4 d660
0x0030:  8cb4 0c8a 1a00 37ab f298 07d5 661e 26ed
0x0040:  e007 2618 0c96 0bc6 35c6 2d06 9a0f 546e
0x0050:  4b7a ae5d 4440 90c2 445c 8da9 c998 0138
0x0060:  bf0b 9220 9256 e3c7 bb18 0ddf 9591 6bcd
0x0070:  8dab 2f73 07ef 0916 1c33 298e 9eb8 1985
```

Рисунок 2.9 – Розшифрований пакет SSH з Chacha20-Poly1305

```
b'\x04^\x00\x00\x00\x00\x00\x00\x00*This is a
test. Just a test. One two three\x8btf\xc8'
```

Рисунок 2.10 – Розшифрований пакет із значенням довжини заповнення

Розшифрований пакет Wireshark. На рис. 2.11 та рис. 2.12 показаний інтерфейс користувача Wireshark з модулем дешифрування;

1. Показує, що ми фільтруємо з'єднання SSH від клієнта до нашого сервера;
2. Показує список пакетів;
3. Показує список пакетів підключення до сервера;
4. Показує деталі пакетів у шістнадцятковому представленні, де ми можемо обачити це з'єднання SSH;
5. Показує розшифрований пакет;
6. Показує номер кадру та розмір розшифрованого пакету.

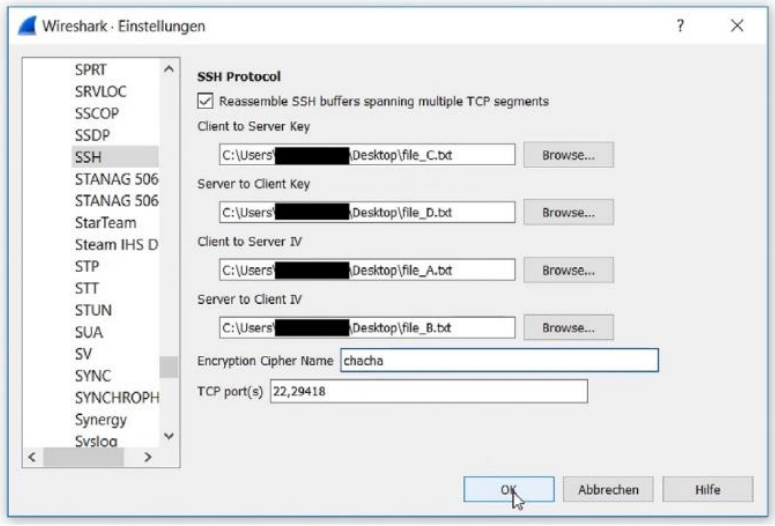


Рисунок 2.11 – Інтерфейс користувача Wireshark (налаштування ключа)

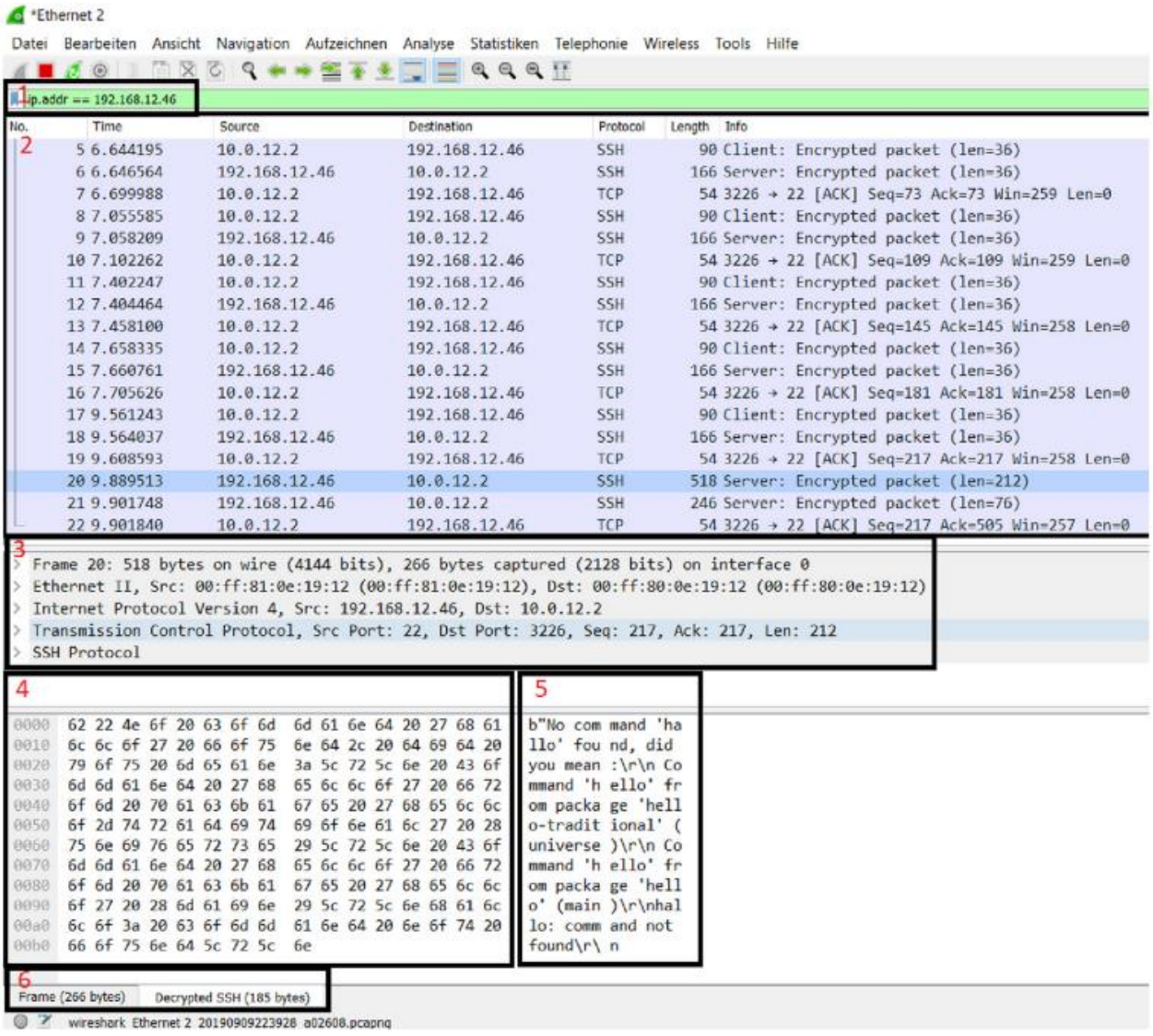


Рисунок 2.12 – Інтерфейс користувача Wireshark (дешифрування пакетів).

Тест продуктивності. Вимірювання продуктивності системи визначається, обчислюючи час підключення з боку клієнта за допомогою методу автентифікації з відкритим ключем. Є кілька сценаріїв фіксації результатів:

1. Встановити сеанс ssh (з AES128-CTR) і негайно його перервати;
2. На SSH-сервері паралельно виконується команду git clone https://github.com/wireshark/wireshark.git і клієнтське ssh-з'єднання. Метою є створення навантаження на моніторинговий сервер.
3. Подібно до 1 варіанту, але з AES256-CTR.
4. Подібно до 2 варіанту, але з AES256-CTR.

Кожен з сценаріїв виконується рівно 100 разів і середній результат відображений на рис. 2.13. У найкращому випадку система збільшує час підключення на 50 мс. З іншого боку, коли сервер має більше навантажень, система створює затримку в 85,8 мс. Цей результат показує, що система має незначний і непомітний вплив на сервер. Результат вимірювання часу дешифрування за допомогою Pycryptodome з кількома шифрами, показано в табл. 2.1. pkl і pyl позначають довжину пакета та корисного навантаження в байтах відповідно. Розшифровка кожного пакета відбувається швидко і оскільки вона виконується паралельно, це не впливає на продуктивність з'єднання.

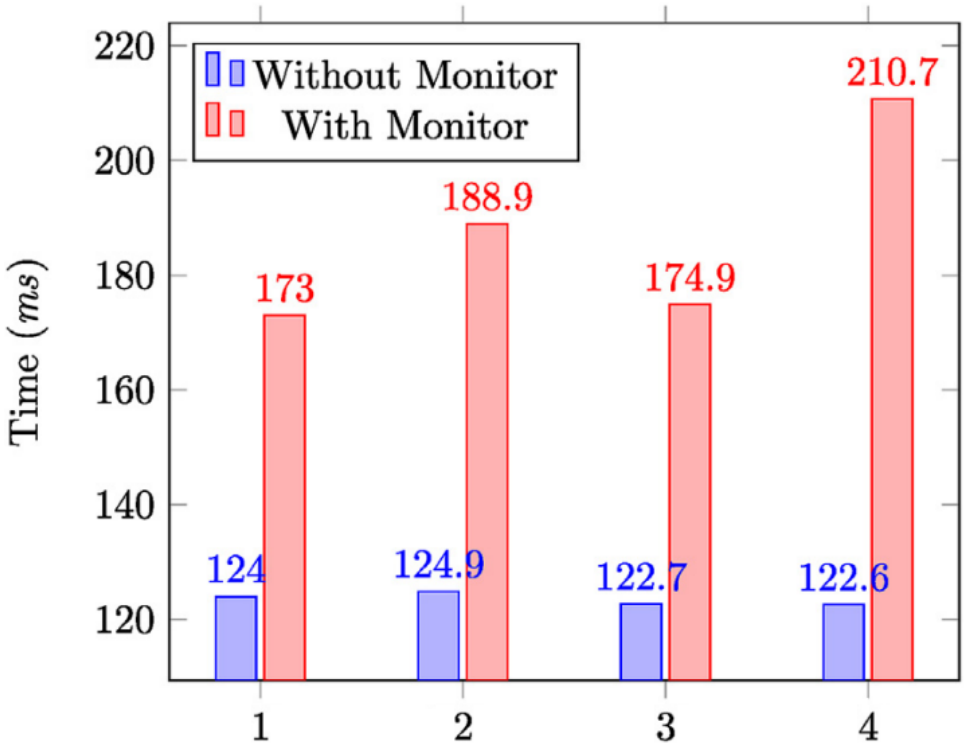


Рисунок 2.13 – Результат тесту продуктивності SSHkex

Таблиця 2.1 – Час розшифровки одного мережевого пакета SSH

AES 128-CTR			AES 256-CTR			ChaCha20-Poly1305		
Pkl, Б	Pyl, Б	Час, мс	Pkl, Б	Pyl, Б	Час, мс	Pkl, Б	Pyl, Б	Час, мс
28	16	0,030	28	16	0,031	28	16	0,066
44	32	0,036	60	48	0,037	56	16	0,070
76	64	0,043	76	64	0,041	76	56	0,069
124	96	0,052	132	32	0,058	132	32	0,079

Висновок. Моніторинг зашифрованих комунікацій SSH є важливим для таких цілей, як аналіз атак та цифрова захищеність даних. Існуючі підходи, які базуються на перехопленні або модифікації двійкових файлів SSH за принципом MITM мають недолік, оскільки вони видимі в цільовій системі, а отже можуть бути виявлені зловмисником. Існуючі підходи на основі VMI пропонують кращу прихованість, але є або неефективними (наприклад, вилучення ключа грубою силою), або мають сильний вплив на цільову систему (наприклад, відстеження всіх мережевих операцій вводу-виводу та вилучення відкритого тексту з пам’яті за допомогою VMI).

Представлений варіант Stewart Sentanoe та Hans P. Reiser демонструє, що новий підхід пропонує чудові властивості, зберігаючи прихованість моніторингу на основі VMI. Застосовуючи трасування до вибраних функцій SSH, які використовуються під час встановлення з’єднання, накладні витрати VMI обмежується коротким моментом на початку сеансу SSH.

Перехоплення дозволяє безпосередньо витягувати ключі сеансу SSH, які потім можна використовувати для дешифрування. За допомогою реалізації прототипу SSHkex, який містить інструмент вилучення ключів, автономний інструмент дешифрування та плагін Wireshark для дешифрування, було продемонстровано, що такий підхід можливий із LibVMI та цільовою системою Linux.

Базуючись на мережевому трафіку, зафіксованому як файли PCAP, їх підхід ефективний і результативний, SSHkex здатний видобувати ключі точно з мінімальними витратами – менше 100 мс. Але їх поточний підхід можливий, лише якщо схема пам’яті структур даних, які використовуються реалізацією SSH, відома екстрактору ключів. Тому для збереження даних, потрібно врахувати цей момент.

3 РОЗРОБЛЕННЯ АЛГОРИТМУ ФУНКЦІОНУВАННЯ ТА СТРУКТУРНОЇ СХЕМИ

3.1 Розроблення алгоритму функціонування пристрою

Для розроблення конкретного алгоритму функціонування системи мультисервісного доступу до корпоративного хмарного сховища, нам потрібно поділити результуючу роботу на менші складові частини [18]. Для цього, перечислимо всі фундаментальні функції, які має виконувати система:

- авторизація користувача за допомогою логіна та пароля;
- прийом користувацьких даних в поширених форматах;
- безпечна передача даних;
- зберігання користувацьких даних на локальному сховищі;
- резервне копіювання за технологією Raid 1;
- редагування користувацьких даних на локальному сховищі;
- видалення користувацьких даних з локального сховища;
- контроль вільного простору в сховищі;
- контроль придатності сховища.

На основі списку функцій, наведених вище побудуємо блок-схему алгоритму роботи системи мультисервісного доступу до корпоративного хмарного сховища, сама схема наведена на рис. 3.1.

Першим пунктом нашої блок схеми є підключення пристрою до маршрутизатора. Рекомендується це робити через порт Ethernet, так як через Wi-Fi набагато простіше перехопити пакети та розшифрувати їх методом SSHkex. Система не має в своїй конструкції дисплеїв та інших індикаторів, які б допомогли налаштувати доступ до сховища, тому при надходженні живлення на систему, автоматично розгортається web-сервер [9].

Підключення для самого корпоративного хмарного сховища повинно бути високошвидкісним, бажано щоб роутер мав гігабітний порт.

Третім пунктом блок схеми є запит на авторизацію користувача. Це потрібно для застосування захисту даних та сегрегації користувачів за правами доступу до файлів, приклад інтерфейсу авторизації наведений на рис. 3.2.

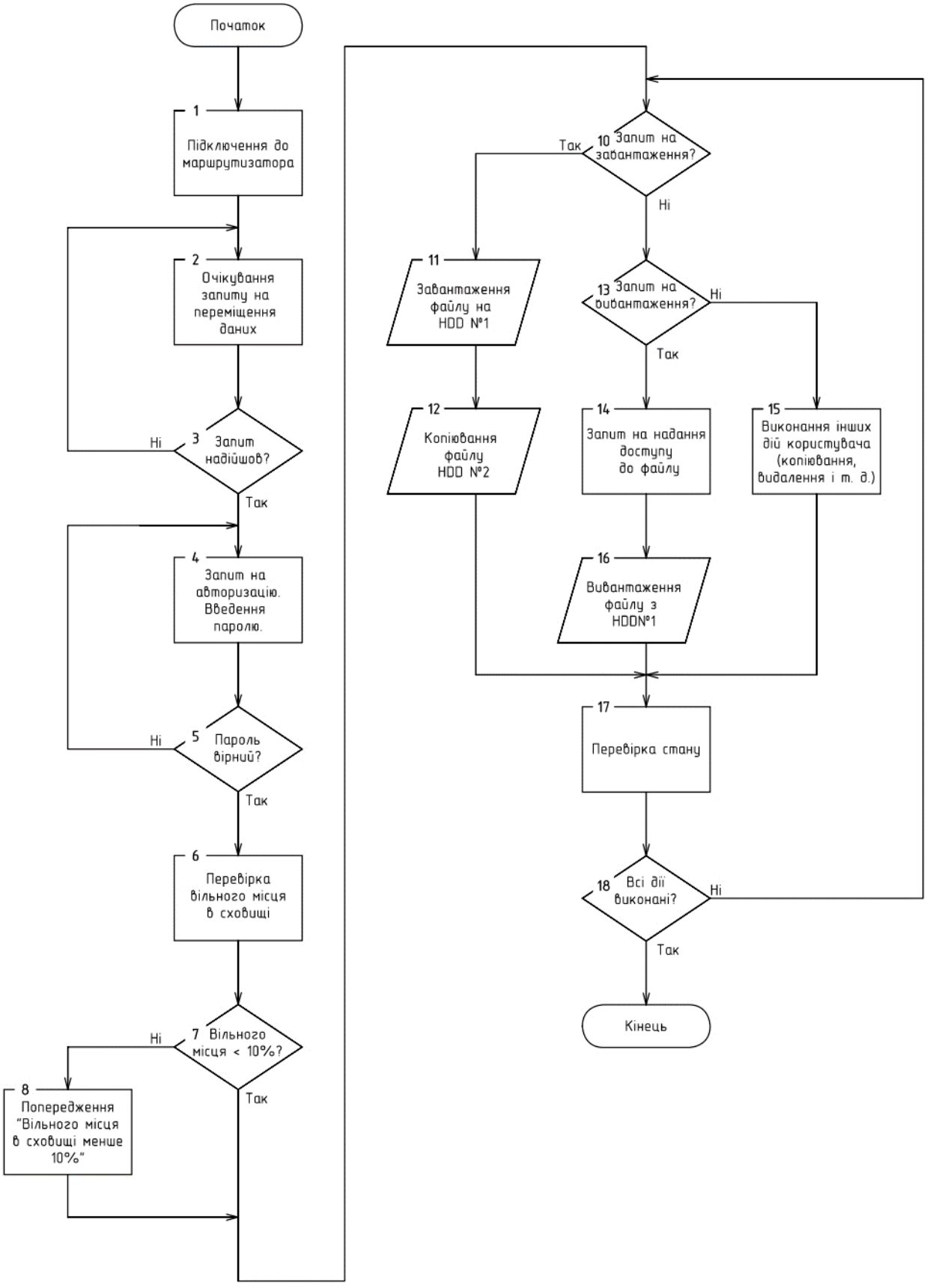


Рисунок 3.1 – Блок схема системи мультисервісного доступу до корпоративного хмарного сховища

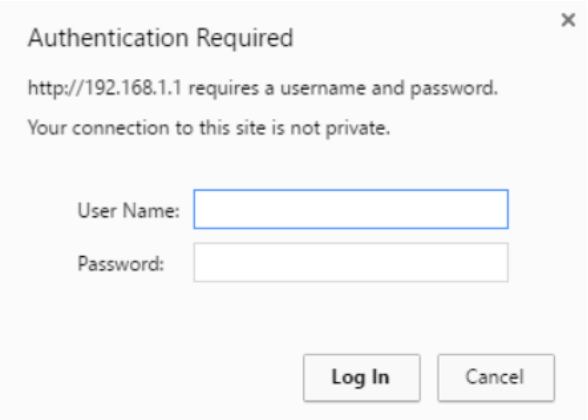


Рисунок 3.2 – Приклад інтерфейсу для автентифікації користувача за

Абонент, який намагається отримати доступ до простору сховища повинна мати власний логін та пароль. Можливий варіант, коли загальні дані для авторизації доступні групі абонентів. Це дозволяє розділити обов'язки, запобігти втрати важливих користувацький даних та організувати безпечний доступ до файлів відслідковуючи час та місце авторизації.

Шостим пунктом блок схеми, являється перевірка вільного дискового простору в сховищі. На сьогоднішній день, всі користувацькі та промислові пристрої збереження інформації, незалежно від форми запису, будь-то твердотільний накопичувач або жорсткий диск використовують технологію самодіагностики self-monitoring, analysis and reporting technology (S.M.A.R.T.). При увімкненні системи пристрій збереження інформації починає нормально функціонувати, відстежуючи власний стан за допомогою спеціальних параметрів і властивостей. Ці параметри контролюються окремою областю в керуючій платі накопичувача і недоступні за нормальних умов.

Сьомим пунктом блок схеми є перевірка, чи не залишилося вільного місця менше 10% від загального простору. Після того, як користувач правильно введе логін і пароль, він безпосередньо входить в графічний інтерфейс і може бачити свій власний простір. Цей пункт необхідний, бо при майже повній завантаженості вільного простору, робота накопичувача значно уповільнюється. Завдяки технології S.M.A.R.T., програмне забезпечення отримує інформацію з накопичувача та подає її у зручному відсотковому виді, для розуміння людиною. Якщо ж в локальному сховищі залишатиметься менше 10% вільного місця, користувач буде отримувати відповідне повідомлення під час завантаження файлів.

Десятим пунктом блок схеми є запит користувача на дозвіл завантаження файлу в сховищі. Для звичайних користувачів, які не мають досвіду роботи з linux-терміналом, зручніше використовувати графічний інтерфейс, тому наш система мультисервісного доступу до корпоративного хмарного сховища матиме програмне забезпечення з графічним інтерфейсом, як показано на рис. 3.3. Взаємодія з файлом запитується клацанням миші на символ «...» у випадку з комп'ютером або довгим натисканням у випадку зі смартфоном. Після цього користувач має вибрати в контекстному меню, що має статися з файлом. Доступні дії користувача: «скопіювати», «видалити», «переглянути», «завантажити», «перейменувати».

Важливо відмітити, за поставленим завданням проектування, система матиме технологію RAID 1 резервного копіювання. Вона вимагає мінімум двох фізичних дисків, оскільки дані записуються одночасно в два місця. Диски фактично є дзеркальним відображенням один одного, як показано на рис. 3.3, тому якщо один диск виходить з ладу, інший може взяти на себе контроль і надати доступ до даних, які зберігаються на цьому диску. Дзеркалювання дисків добре для дуже швидких операцій читання, але повільніше під час запису на диски, оскільки дані потрібно записувати в два місця.

Рисунок 3.3 – Візуалізація технології RAID 1

Тринадцятий крок блок схеми це запит на переміщення файлів. Коли надходить запит на передачу файлу, програмне забезпечення обчислює приблизний час, який знадобиться для передачі файлу та відображає цю інформацію користувачеві. Час передачі залежить від багатьох факторів, основними з яких є: розміру переданих файлів, кількості цих файлів, швидкості порту роутера, якість каналу зв'язку, швидкість мережі інтернет та швидкості

приймача. Потім відповідна інформація буде надіслана на IP-адресу користувача, який ініціював передачу файлу.

Вісімнадцятий крок, це фінальний етап роботи системи, тут йде перевірка виконання поставлених задач перед системою і якщо щось було не завершено, то весь цикл програми повториться знову. Якщо ж вся робота була виконана коректно, так користувач закрив web-інтерфейс хмари – відбудеться розлогінування з системи.

Всі вище описані етапи легітимні лише для користувацького облікового запису, але в системі присутній також акаунт адміністратора. Він має більше прав, по відношенню до користувацького та існує в єдиному екземплярі, його ціль – внесення глобальних налаштувань в систему та правка спірних моментів при використанні.

3.2 Розробка структурної схеми пристрою

Згідно з завдання та початкових даних, необхідно розробити систему, яка реалізує функцію мультисервісного доступу до корпоративного хмарного сховища. Система повинна мати невеликі габарити, не мати декоративного пластикового корпусу, який би заважав охолодженню, бути швидкою, та бути захищеною.

Хмарне сховище повинно забезпечувати виконання наступних функцій:

- завантаження через мережу файлів різних форматів;
- доступ до системи через мережу інтернет;
- доступ до системи через технологію Ethernet;
- доступ до системи через вбудований Wi-Fi модуль;
- редагування завантажених файлів в локальної хмарі;
- видалення завантажених файлів в локальної хмарі;
- реалізація різних прав доступу до файлів;
- захист користувацьких даних, які розташовані в сховищі;
- контроль рівня вільного місця в сховищі;
- контроль фізичного стану сховища;
- забезпечення збереження даних за технологією RAID 1;
- графічний інтерфейс програми сумісний зі смартфонами.

Маршрутизатор, який виконує роль зв’язуючої ланки системи не буде входити в комплект системи та не буде врахований в техніко-економічній частині, але його наявність критична для коректного функціонування всієї системи. За структурною схемою, яка наведена на рис. 2.4, весь потік даних проходить саме через маршрутизатор та саме він може бути вузьким місцем всієї системи.

Таке рішення, не включати роутер у загальний кошторис, було прийняте за декількох причин. Насамперед, це дозволяє кінцевому споживачеві самому обрати пристрій, без нав’язування конкретної моделі та виробника розробником. Також, велика вірогідність, що маршрутизатор у користувача вже є, адже розвиток мережі інтернет на 2022 рік в Україні доволі великий, навіть у селах є оптоволоконне підключення. Ну і звісно, відсутність маршрутизатора в кошторисі позитивно вплине на ціну фінального пристрою.

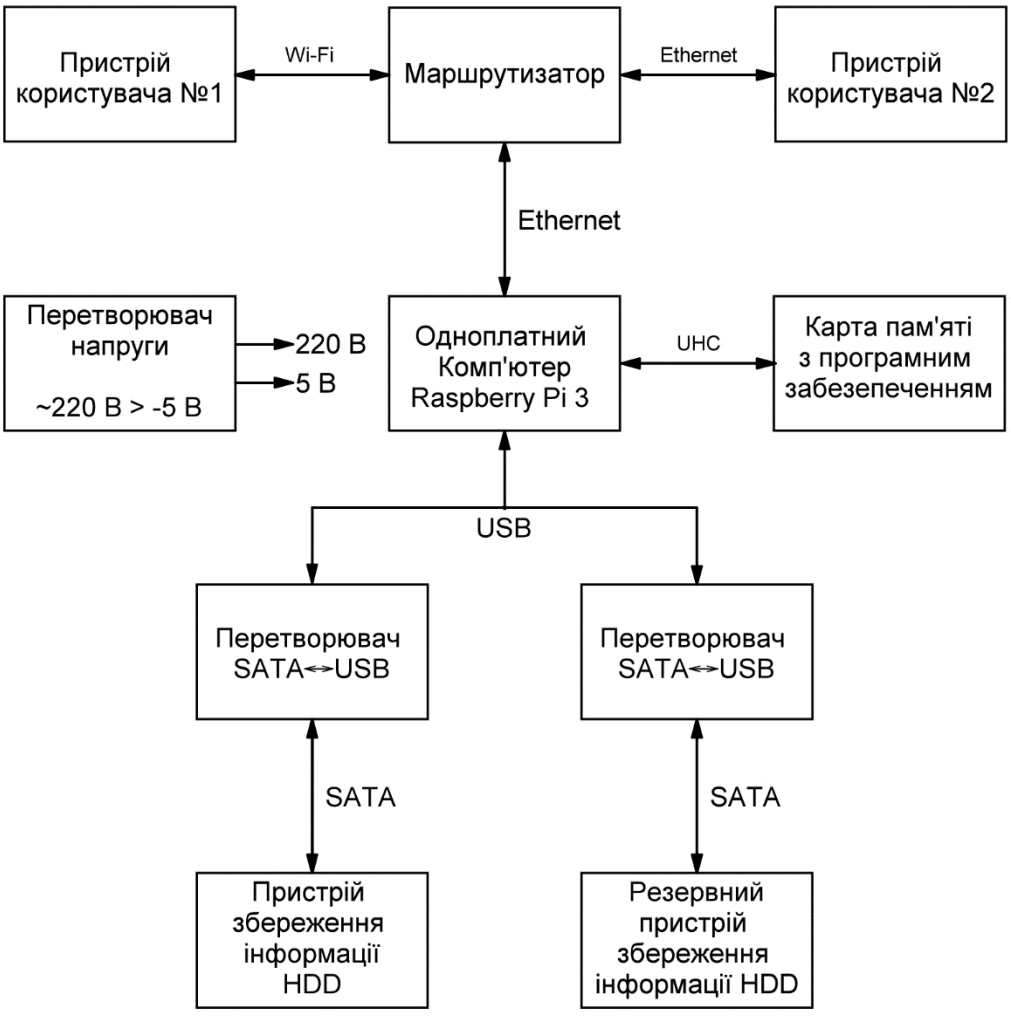


Рисунок 3.4 – Структурна схема мережевого сховища даних

Система виконує багато корисного навантаження, тому її не можна назвати простою, вона повинна «вміти» працювати з поширеними форматами даних, серед яких є вимогливі до ресурсів формати відео, розгортати на собі сервер з графічним інтерфейсом, працювати з доступом до інтернету і робити це безпечно для інформації, яка зберігається на підключеному сховищі. Для такого роду завдання мікроконтролер з декількома десятками оперативної пам’яті та ядром в мегагерцовому діапазоні уже не підійде, доцільно застосувати потужніші плати, які називаються одноплатними комп’ютерами на ARM архітектурі.

Жорсткі диски, які ми будемо використовувати в якості простору для сховища, використовують SATA-інтерфейс, який зображений на рис. 3.5, для прийомо-передачі даних. Для одноплатних комп’ютерів наявність повноцінного SATA-порту, скоріше виключення ніж правило, з огляду на їх фізичні розміри. Тому для поєднання цих двох структур будемо застосувати двонаправлені схеми перехідники SATA-USB.

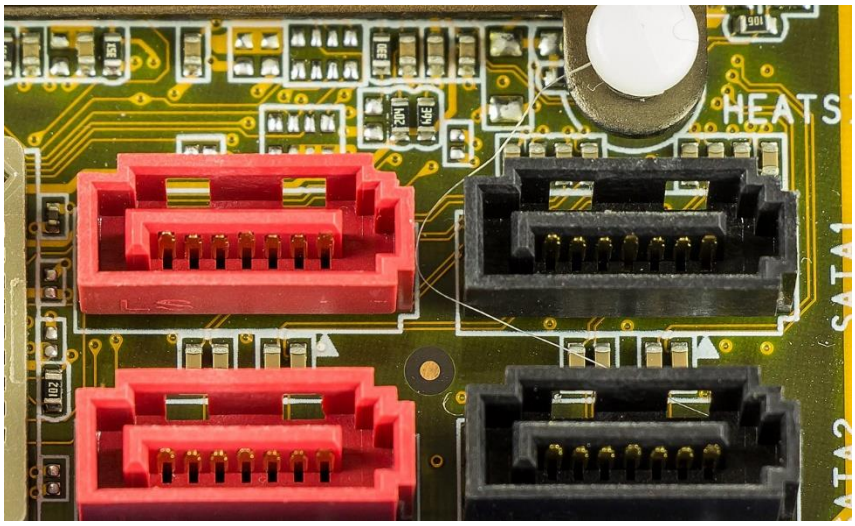


Рисунок 3.5 – Зовнішній вигляд SATA порту

Більшість одноплатних комп’ютерів мають в своїй будові розпаяну оперативну пам’ять, але не мають постійної, для розташування користувацької програми, для таких цілей в них розпаяний порт для встановлення SD-карти.

Найкращим варіантом постійної пам’яті для одноплатного комп’ютера є SD-карта [20] на 16 Гб, зі швидкодією 10-го класу. Об’єму пам’яті в 16 ГБ цілком вистачить для програмного забезпечення з графічним інтерфейсом, а 10 клас швидкодії забезпечить стабільну роботу системи.

4 РОЗРОБЛЕННЯ ФУНКЦІОНАЛЬНОЇ СХЕМИ ПРИСТРОЮ

Людині для користування системою мультисервісного, корпоративного хмарного сховища необхідний мати будь-який мобільний пристрій під керуванням ОС Android, IOS або ж персональний комп'ютер з ОС Windows, Linux. Принципові різниця між цими двома типами пристроїв – метод підключення до хмарного сховища, якщо смартфон буде з'єднуватися через менш безпечний Wi-Fi, то персональний комп'ютер буде підключений через Ethernet. Ці два варіанти розглянуті та зображені на структурній схемі, і записані як «Пристрій користувача №1» та «Пристрій користувача №2». Ці пункти можуть динамічно змінюватись, але на структурній схемі зображений максимально різноманітний варіант.

Коли користувач робить запит у браузері свого пристрою, незважаючи з якого пристрою, запит надсилається на маршрутизатор, приклад якого зображений на рис. 4.1. На структурній схемі цей пристрій записується як «Маршрутизатор».

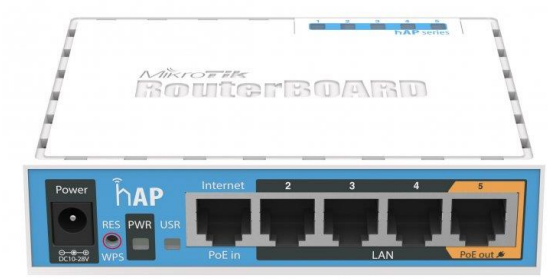


Рисунок 4.1 – Зовнішній вигляд маршрутизатору

Маршрутизатор отримує дані запиту з пристрою користувача, він намагається знайти співпадіння IP-адрес в своїй таблиці маршрутизації, якщо ж користувацький пристрій вперше з'єднується, то відповідно і потрібна адреса не буде знайдена. При такому співпадінні обставин, спрацює відпрацьований, автоматичний алгоритм. Маршрутизатор починає сканувати та збирати певну інформацію про всі нові пристрої, які підключені та авторизовані з ним. Після цього процесу, на всі знайдені IP-адреси буде відправлений широкомовний запит. Всі пристрої, що отримали широкомовний запит, зобов'язані відправити всю запрошену інформацію маршрутизатором, для побудови оновленої таблиці маршрутизації. Якщо все пройшло вдало та система була підключена до цього

маршрутизатора, запит користувача буде доставлено до комп’ютера. Цей процес зазвичай проходить дуже швидко і майже не помітний для користувача. Тепер, коли запит надійшов, користувачу необхідно увести свій логін та пароль, який йому видав адміністратор сховища.

Для досягнення максимальної швидкості передачі даних і зменшення циклів повторного запиту файлів необхідний одноплатний комп’ютер, підключений до маршрутизатора через кабель Ethernet. На структурній схемі, він записаний як «Одноплатний комп’ютер raspberry pi 3 b+».. Він відіграє важливу роль у нашій системі, обмінюючись пакетами даних з користувачем, захищаючи особисту інформацію та забезпечує потужністю відображення графічного інтерфейсу для взаємодії з жорстким диском. Одноплатний комп’ютер, показаний на рис. 3.2, з нього видно, що він дуже компактний і не займе багато місця в шафі безпеки. Він відіграє ключову роль для нашої системи, оскільки він формує пакети для надсилання користувачеві, захищає особисту інформацію та розгортає графічний інтерфейс.

Рисунок 4.2 – Одноплатний комп’ютер

Одноплатний комп’ютер, технічно має мінімальні розбіжності зі звичайним, він також має свій процесор, відео ядро, мережу карту та оперативну пам’ять, але для зменшення габаритів вони розпаяні на платі. Постійної пам’яті одноплатний комп’ютер не має, свою операційну систему та системні файли він зберігає на SD-картці, яку користувач вставляє у відповідний слот. Слот зв’язаний з центральним процесором за допомогою протоколу UHS, це дозволяє їм працювати з картами формату SD. На структурній схемі, цей пристрій записаний як «Карта пам’яті з програмним забезпеченням».

Живлення системи реалізовано послідовно, джерело живлення на 3 А та робочою напругою 5 В, з’єднується з одноплатним комп’ютером через інтерфейс

microUSB, на самому комп’ютері є лінійний стабілізатор напруги, який здатен живити всю периферію підключену до нього. Враховуючи, що комп’ютер має велику кількість незадіяних в нашому проектів енергоємких портів для переферії, таких як наприклад шлейф для 10 дюймового сенсорного дисплею, то всього два жорстких диска підключених до порту USB він буде живити без жодних проблем. На структурній схемі, цей пристрій записаний як «Перетворювач напруги».

Після успішної авторизації користувача в системі, формується команда на взаємодію з файлом розташованим на жорсткому диску. Операційна система надсилає запит на керуючу плату HDD через адаптер USB-SATA. Одноплатні комп'ютери не мають у своїй конструкції роз'ємів SATA, а всі сучасні жорсткі диски використовують для обміну даними лише інтерфейси SATA. Тому для координації їх роботи використовується схема конвертеру USB-SATA, який формує запити для керуючої плати HDD, а той в свою чергу, відправляє запрошену інформацію з жорсткого диска на одноплатний комп'ютер, а потім і на пристрій користувача. На блок-схемі цей пристрій записується як «SATA ←→USB Converter» [21].

Всі дані користувачів зберігається на основному та резервному жорстких дисках, які підключені через відповідно конвертори рівнів до одноплатного комп'ютера. Керуюча плата жорсткого диску надсилає на комп’ютер всі запрошені дані користувача та свої діагностичні дані. На структурній схемі пристрій записується як «пристрій для зберігання інформації». На основі вищесказаного побудуємо функціональний схему і зобразимо її на рис. 3.3.

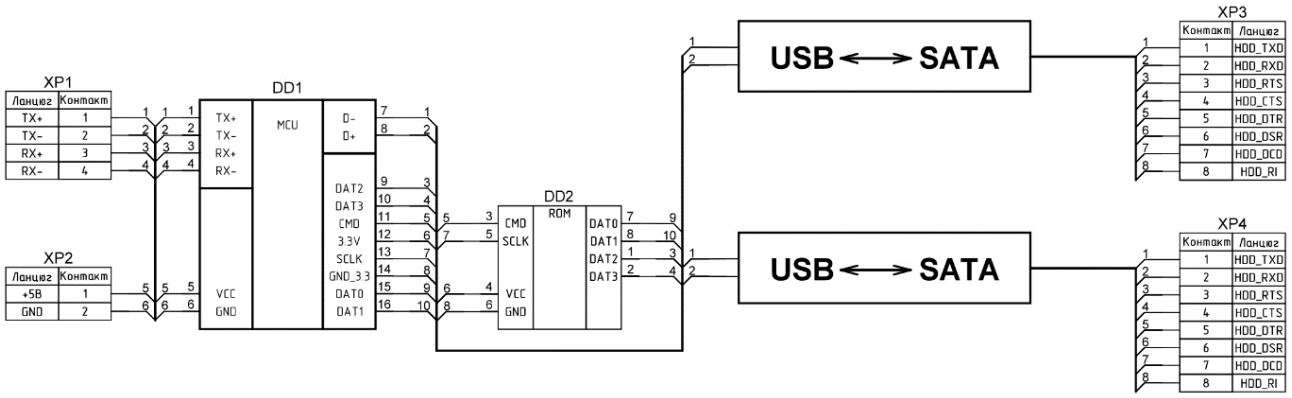


Рисунок 4.3 – Функціональна схема пристрою

5 РОЗРОБКА ПРИНЦИПОВОЇ СХЕМИ ТА ВИБІР ЕЛЕМЕНТНОЇ БАЗИ

5.1 Raspberry Pi 3 B

Raspberry Pi перетворився на дуже потужний маленький комп'ютер, який протягом 6 років повільно захоплював світ. Загалом було продано близько 19 мільйонів одиниць Raspberry Pis, з них 9 мільйонів одиниць попередньої версії було продано лише за 2 роки; вражаючі цифри для комп'ютера, враховуючи, що він коштує лише 35 доларів США.

Новий модуль Raspberry Pi 3 Model B+ є останньою ітерацією верхньої лінії Raspberry Pi, розробленою Raspberry Pi Foundation. Він забезпечує підвищену продуктивність завдяки швидшому процесору та 5 ГГц Wi-Fi. Для деяких завдань він здається значно швидшим, ніж його попередня ітерація Raspberry Pi 3 Model B, для деяких інших завдань він виглядає приблизно так само.

Обчислювальний модуль Raspberry Pi 3 B+ є механічно сумісною системою на модулях DDR2-SODIMM, що містить процесор, пам'ять, eMMC Flash і допоміжні схеми живлення. Ці модулі дозволяють розробнику використовувати стек обладнання та програмного забезпечення Raspberry Pi у власних системах і форм-факторах. Наведемо на рис. 5.1, зовнішній вигляд та основні інтерфейси Raspberry Pi 3 B+.

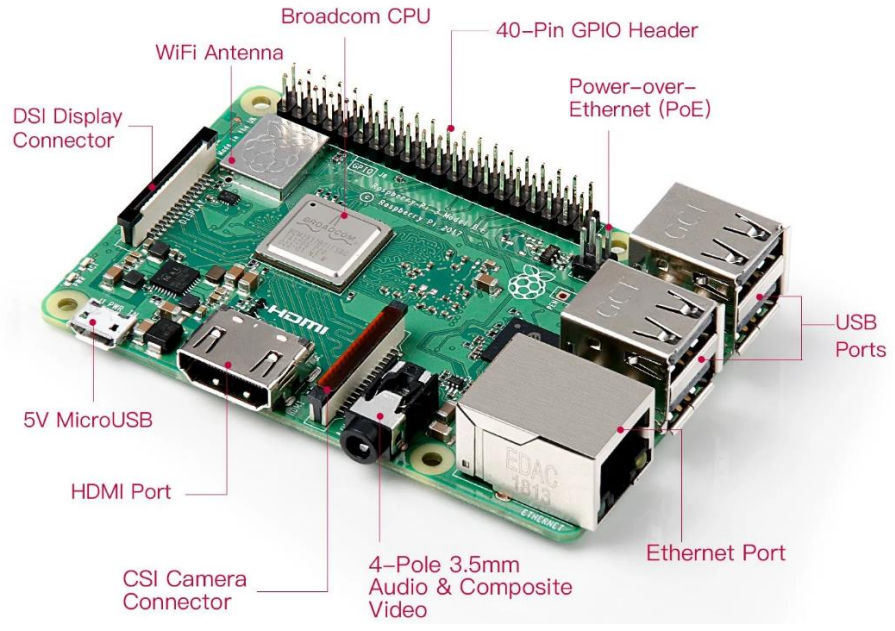


Рисунок 5.1 – Основні інтерфейси Raspberry Pi 3 B+

Raspberry Pi 3 B+, структурна схема якого зображена на рис. 5.2, містить процесор BCM2837, 1 Гбайт оперативної пам'яті LPDDR2, але не має вбудованої Flash-пам'яті. Натомість плата має контакти інтерфейсу SD/eMMC доступні для користувача, щоб підключити власний пристрій SD/eMMC. Звернемо увагу, що процесор BCM2837 є еволюцією процесора BCM2835, який використовувався в попередніх ітераціях модуля Raspberry. Єдині реальні відмінності полягають у тому, що BCM2837 може використовувати більше оперативної пам'яті (до 1 Гбайт), а комплекс центрального процесору ARM було оновлено з одноядерного ARM11 у BCM2835 до чотирьохядерного Cortex A53 із виділеним 512 Кбайт кеш-пам'яті L2 у BCM2837. BCM2837 має в своїй будові двох-ядерний графічний співпроцесор Video Core IV® Multimedia, який забезпечує Open GL ES 2.0, апаратне прискорення Open VG і 1080p30 H.264 декодування.

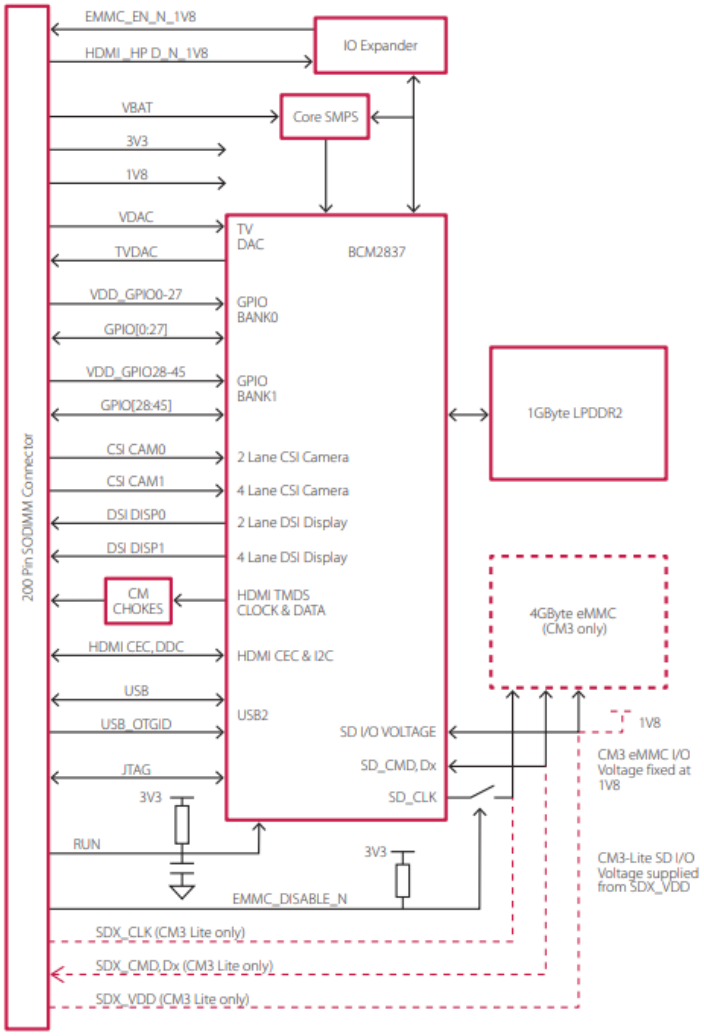


Рисунок 5.2 – Структурна схема Raspberry Pi 3 B+

Cortex-A53 – процесор середнього класу з низьким енергоспоживанням, який реалізує архітектуру ARMv8-A. Процесор Cortex-A53 має від одного до чотирьох ядер, кожне з яких має систему пам'яті L1 і один спільний кеш L2. На рис. 5.2 показано приклад конфігурації Cortex-A53 MPCore з чотирма ядрами та інтерфейсом ACE або CHI.

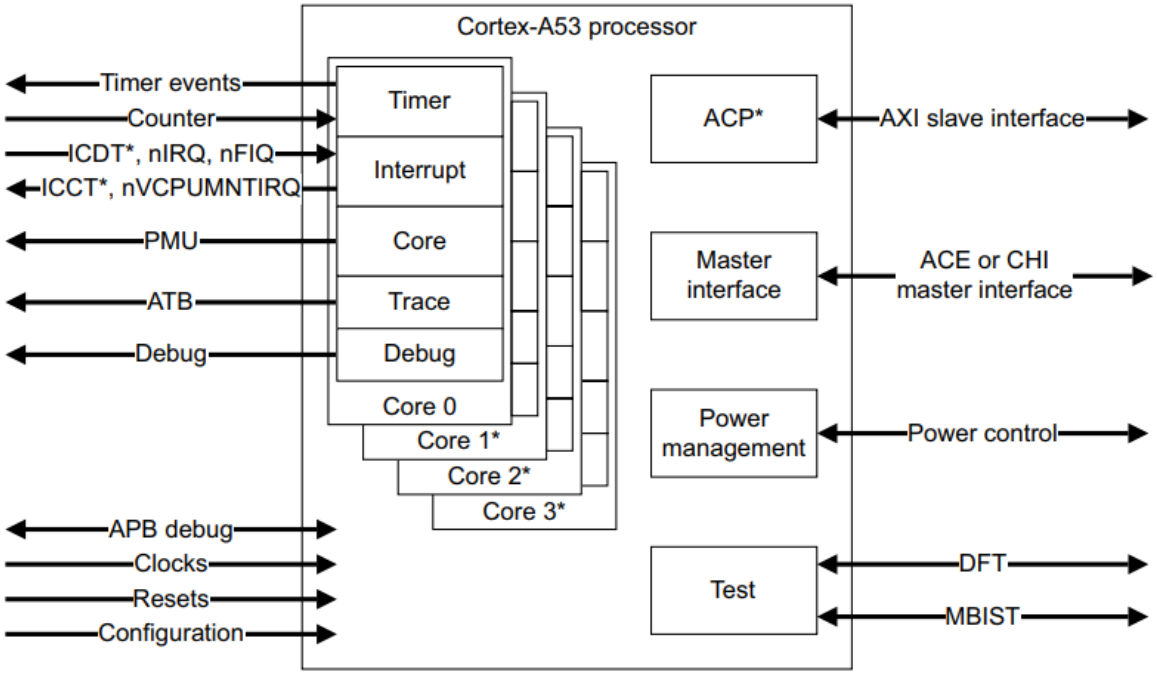


Рисунок 5.3 – Конфігурація процесора Cortex-A53

Додаткове розширення шифрування Cortex-A53 MPCore підтримує криптографію ARMv8 Розширення. Розширення Cryptography додає нові інструкції A64, A32 і T32 в розширений SIMD, який прискорює:

- шифрування та дешифрування Advanced Encryption Standard (AES);
- алгоритм безпечного хешування (SHA) виконує функції SHA-1, SHA-224 і SHA-256;
- арифметика кінцевого поля, яка використовується в таких алгоритмах, як режим Галуа/Лічильник і еліптична крива.

Усі інтерфейси вводу-виводу та периферійні пристрої залишаються незмінними, тому обидва чіпи в основному програмно та апаратно сумісні.

Розпіновка Raspberry Pi 3 B+ практично не змінилася з моменту появи 40-контактного роз'єму, додано лише 4-контактний роз'єм (2x2) для підтримки Power over Ethernet (PoE), зобразимо розпіновку на рис. 5.3.

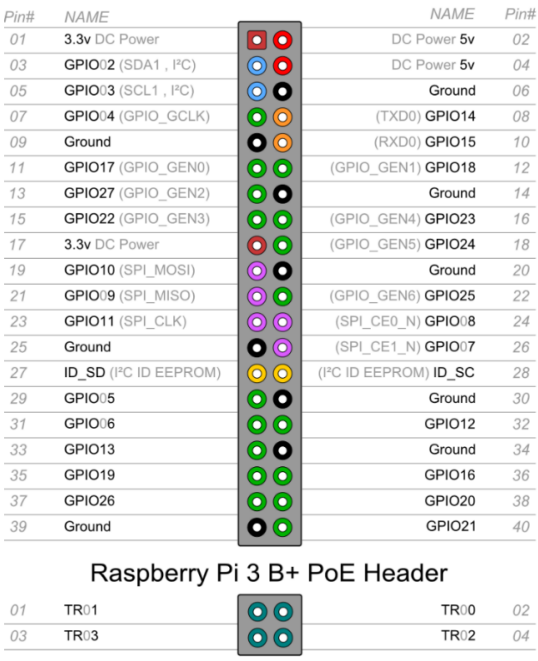


Рисунок 5.4 – Розпіновка 40-пінового порту

Однією зі змін версії B+, було значне вдосконалення вбудованої локальної мережі. Замість контролера 10/100, чіпсет Pi містить контролер 10/100/1000 Gigabit LAN.

Як показує порівняльна характеристика мережі різних версій плат Raspberry, яка наведена на рис. 5.4, ми отримуємо набагато більшу пропускну здатність від моделі B+, і це позбавляє необхідності використовувати зовнішній адаптер Gigabit USB 3.0 для отримання швидшого дротового доступу до локальної мережі, як це потрібно було робити на попередніх моделях.

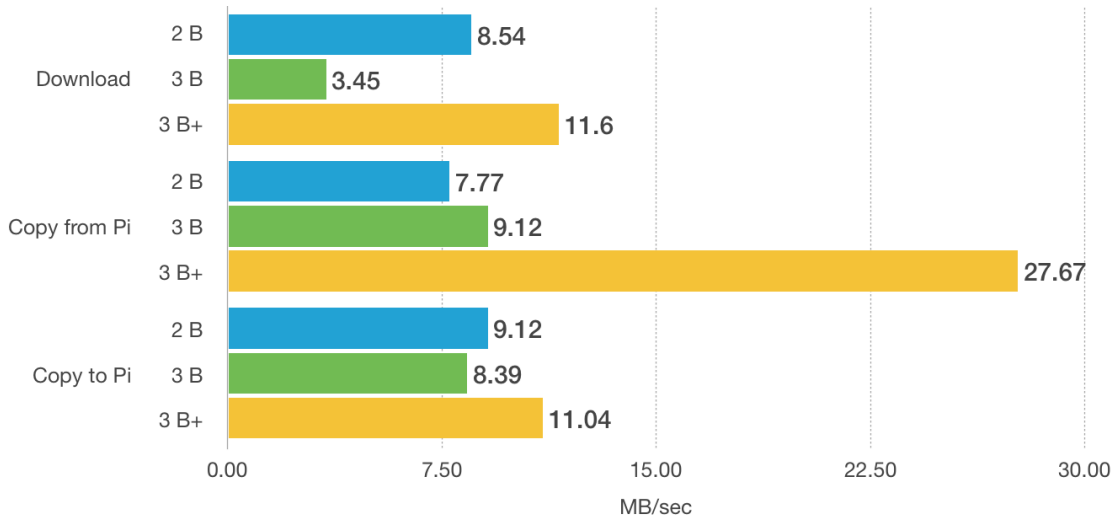


Рисунок 5.5 – Швидкість LAN різних версій плат Raspberry

Модель Pi 3 B+ значно швидша, ніж моделі 2 B і 3 B, особливо коли немає жодної іншої активності на спільній шині, через яку він працює (як у випадку з мережевою операцією читання, яка є кешованою та дуже швидкою). Однак навіть 3 B+ має серйозні перешкоди порівняно з іншими SBC із спеціальною мережевою шиною, яка може пропускати справжній Ethernet 1 Гбіт/с незалежно від операцій з карткою microSD.

Продуктивність Wi-Fi моделі 3 B+ знищує 3 B. Це видно з порівняльної характеристики, наведеної на рис. 5.5. Крім того, на відміну від вибагливих налаштувань WiFi на 3 B, де доводилося мати справу з дивними помилками. Налаштування зводилось до редагування всередині утиліти raspi-config з командного рядка, і навіть не потрібно було вручну редагувати будь-які мережеві файли або перезапускати будь-які служби чи інтерфейси.

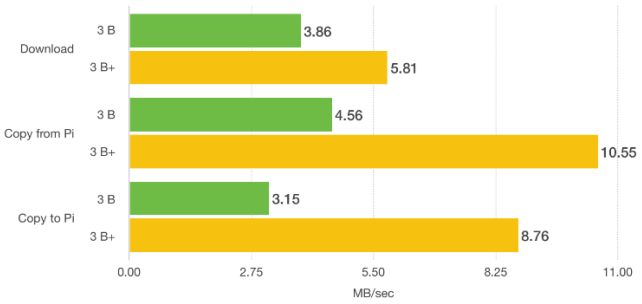


Рисунок 5.6 – Швидкість Wi-Fi різних версій плат Raspberry

Кожне покоління Raspberry має процесор з вищою тактовою частотою, а також кілька нових схем, які потребують постійного живлення просто для роботи в режимі очікування і 3 B+ не є винятком. Результати проведених тестів з енергоспоживанням різних версій Raspberry наведені на рис. 5.5.

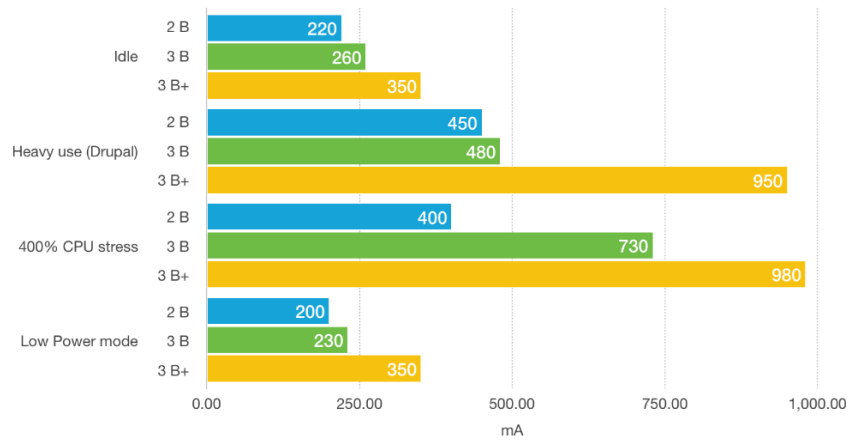


Рисунок 5.7 – Порівняльна характеристика енергоспоживання Raspberry

Для трохи більшої обчислювальної потужності та значно більшої пропускної здатності мережі Pi model 3 B+ споживає майже вдвічі більше електроенергії, ніж попереднє покоління Pi model 3. Інші SBC, такі як Tinker Board, споживають ще вдвічі більше, ніж використовує 3 B+, але це також більш ніж подвоює показники продуктивності.

Кожне покоління Raspberry Pi підвищує вимоги до джерела живлення. Перші кілька Raspberry Pi можна було живити від будь-якого маленького USB-адаптеру живлення, і окрім мало імовірних збоїв – більше ніякої небезпеки не було.

ARM процесори мали перевагу серед конкурентів в сфері вбудованих систем, таких як смартфони, планшети та розумні годинники, бо мають гарне співвідношення потужність до тепловиділення. Але для використання в задачах, де потрібно більше ресурсів, вони не підходили, головним чином це було через 32-х бітну адресацію.

Архітектура процесора Cortex визначає набір команд керування низького рівня, алгоритми керування пам'яттю та кешем. Дуже поширеними є архітектура ARMv7 – для 32 біт та ARMv8 – для 32/64 розрядних систем, серед яких ядро Cortex-A53, що використовується в Raspberry Pi 3 B+ . Архітектура ARMv8 це не найновіша архітектура, світ побачив її в 2011, (найновіша має назву ARMv9 і має перевагу лише в сфері Machine Learning) але її потужності до сих пір вистачає для енергоємних задач, різниця між версіями архітектур до V8, наведена на рис. 5.7.

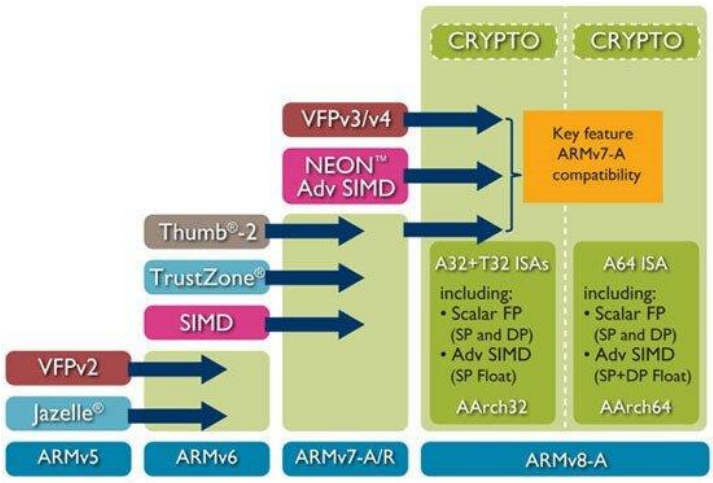


Рисунок 5.8 – Версії архітектури ARM

В архітектурі ARMv8, так як і в новітній ARMv9, реалізований принцип зворотної програмної сумісності попередніх версій архітектур. Це дозволяє застосовувати попередні напрацювання на нових версіях процесорів.

ARMv8, в порівнянні з ARMv7, більш повно відображає ідеологію RISC-архітектури, яка створювалася не для вбудованих систем, а для вирішення класичних завдань обчислень і обробки даних. Архітектура ARMv7 та її попередні версії спочатку містили властивості, які були вигідні для вимог вбудованих систем і диктували необхідність використовувати регістри особливим чином для керування перериваннями в режимі реального часу. Ці функції не відображаються в режимі AArch64, тому обидві командні системи були об'єднані в архітектурі ARMv8. Набір команд режиму AArch64 адаптований для роботи в одному з двох режимів залежно від операційної системи. Якщо операційною системою є Linux або інші поширені клони UNIX, тоді вибирається режим LP64, де цілі числа мають довжину в 32 біти, а довгі цілі числа мають довжину 64 біти, а якщо операційною системою є Windows, тоді вибирається режим LLP64, де цілі числа та довгі цілі числа мають довжину 32 біта, а дуже довгі цілі числа (long long integers) – 64 біти.

Режим AArch64 використовує новий набір інструкцій A64 із 32-розрядними інструкціями та 5-розрядними вказівниками на регістри для доступу до 32/64/128-розрядних регістрів. Спеціальний нульовий регістр доступний, коли виконується більшість команд. Покажчик стека та лічильник команд не входять до групи регістрів, нові команди підтримують 32-розрядні та 64-розрядні дані. У наборі A64 відсутні кілька команд завантаження або збереження довільної довжини. Підтримка архітектури SIMD і операцій з плаваючою комою аналогічна підтримці попередньої версії архітектури – ARMv7A. В архітектурі з'явився спеціалізований інструмент Embedded Trace Mode 4, за допомогою якого адреса розширюється до 64 біт. До основних особливостей AArch64:

- новий набір команд A64;
- регістр загального призначення довжиною в 64 біта;
- окремі регістри SP і PC;
- більшість команд працюють сумісні з 32 та 64-бітними аргументами;
- реалізовано розширення NEON;
- з 16 до 32 збільшено кількість 128-розрядних регістрів, доступних через співпроцесори NEON, VFPv4, крипто-команди AES, SHA.

5.2 Flash-накопичувач для ОС

Флеш-пам'ять це різновид напівпровідникової технології пам'яті EEPROM, що електрично перепрограмовується. Це ж слово використовується в електронній схемотехніці для позначення технологічно закінчених рішень постійних пристроїв у вигляді мікросхем на базі цієї напівпровідникової технології. У побуті це словосполучення закріпилося за широким класом твердотільних пристроїв для зберігання інформації.

Завдяки компактності, дешевизні, механічній міцності, великому об'єму, швидкості роботи та низькому енергоспоживання, флеш-пам'ять широко використовується у цифрових портативних пристроях та носіях інформації. Серйозним недоліком даної технології є обмежений термін експлуатації носіїв, а також чутливість до електростатичного розряду [22].

Елементарний осередок зберігання даних флеш-пам'яті є транзистором з плаваючим затвором. Особливість такого транзистора полягає в тому, що він уміє утримувати електрони (заряд). Ось на його основі і розроблено основні типи флеш-пам'яті NAND та NOR. Конкуренції між ними немає, тому що кожен з типів має свою перевагу і недолік. До речі, на їх основі будують гібридні версії, такі як DiNOR та superAND. У флеш-пам'яті виробники використовують два типи осередків пам'яті MLC та SLC.

Флеш-пам'ять з MLC (Multi-level cell - багаторівневі осередки пам'яті) осередки більш ємні та дешеві, але вони з більшим часом доступу та меншою кількістю циклів запису/прання (близько 10000).

Флеш-пам'ять, що містить у собі SLC комірки, має максимальну кількість циклів запису/стирання 100000 і мають менший час доступу.

Стирання, запис та читання флеш-пам'яті завжди відбувається відносно великими блоками різного розміру, при цьому розмір блоку стирання завжди більше ніж блок запису, а розмір блоку запису не менше ніж розмір блоку читання. Власне це характерна відмітна ознака флеш-пам'яті по відношенню до класичної пам'яті EEPROM.

Як наслідок – всі мікросхеми флеш-пам'яті мають яскраво виражену ієрархічну структуру. Пам'ять розбивається на блоки, блоки складаються із секторів, сектори зі сторінок. Залежно від призначення конкретної мікросхеми глибина ієрархії та розмір елементів може змінюватись.

Осередок пам'яті з одним транзистором. Якщо на керуючий затвор подати позитивну напругу (ініціалізація осередку пам'яті) то він перебуватиме у відкритому стані, що відповідатиме логічному нулю. Наведемо структурну будову польового транзистора з плаваючим затвором на рис ЧЧЧ.

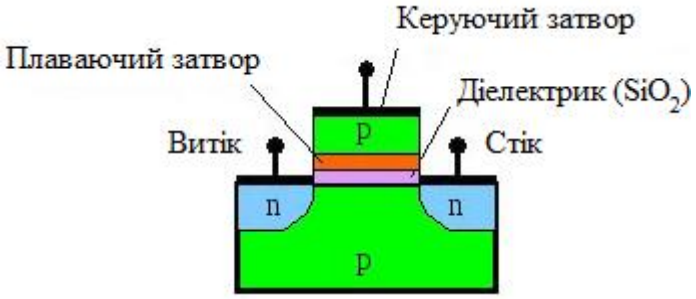


Рисунок 5.9 – Структура польового транзистора

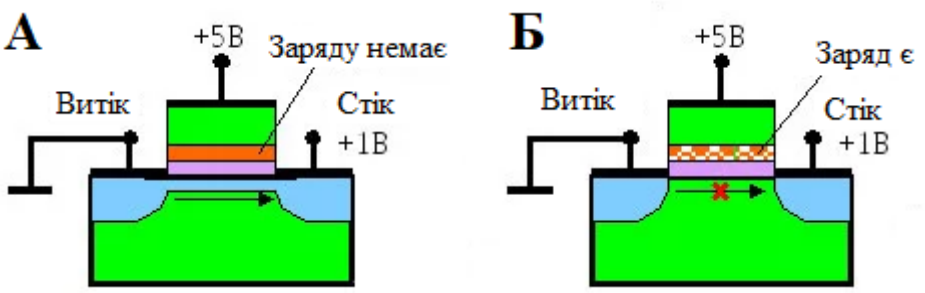


Рисунок 5.10 – читання логічного нуля (А) і логічної одиниці (Б) з комірки пам’яті на базі транзистора з плаваючим затвором

А якщо на плаваючий затвор помістити надлишковий негативний заряд (електрон) і подати позитивну напругу на затвор керуючий, то він компенсує створюване керуючим затвором електричне поле і не дасть утворюватися каналу провідності, а значить транзистор буде знаходитися в закритому стані. Ось так, наявність або відсутність заряду на плаваючому затворі точно визначає стан транзистору. Якщо ми будемо розглядати подачу напруги на затвор, що керує, як ініціалізацію осередку пам'яті, то по тому, яка напруга між витоком і стоком можна судити про наявність або відсутність заряду на плаваючому затворі. Таким чином виходить своєрідний елементарний осередок пам'яті, здатний зберігати один інформаційний біт. До цього дуже важливо, щоб заряд на плаваючому затворі міг зберігатися там довго, як при ініціалізації осередку пам'яті, так і при відсутності напруги на затворі, тільки в цьому випадку осередок пам'яті буде енергонезалежним.

Якщо використовується метод інжекції гарячих електронів, то на стік і затвор, подається висока напруга це додає електронам у каналі достатньої енергії, щоб подолати потенційний бар'єр, який створюється тонким шаром діелектрика і тунелювати в область плаваючого затвора. Під час читання на керуючий затвор подається менша напруга та ефект тунелювання не відбувається.

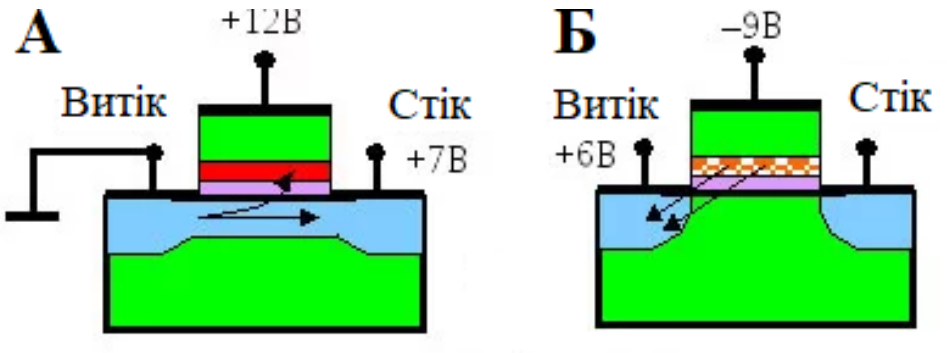


Рисунок 5.11 – Елементарна комірка пам’яті, процес запису (А), процес стирання (Б)

Щоб видалити заряд з плаваючого затвора (виконати стирання комірки пам'яті) на затвор, подається висока негативна напруга, близько 9 В, а на область витоку подається позитивна напруга. Це призводить до того, що електрони тунелюють з області затвора в область витоку. Таким чином відбувається квантове тунелювання Фаулера – Нордхейма. Структурна схема процесів зображена на рис. 5.11.

Комірки з одним транзистором мають деякі недоліки, основним з яких є погана масштабованість. Так як при створенні масиву пам'яті, кожен осередок пам'яті підключається до двох перпендикулярних шин. Затвори підключаються до шини, яку називають Word Line, а стоки з'єднують з шиною, яку називають Bit Line. Тому в схемі знаходиться висока напруга і при записі шляхом інжекції гарячих електронів всі Word Line та Bit Line необхідно розмістити на великій відстані один від одного. Це дасть необхідний рівень ізоляції, але вплине на обмеження обсягу флеш-пам'яті.

Ще одним недоліком такого осередку пам'яті є присутність ефекту надлишкового видалення заряду з затвора, а він не може компенсуватися процесом запису. Тому на плаваючому затворі утворюється позитивний заряд, що робить постійним стан транзистора і він завжди залишається відкритим.

Двотранзисторний осередок пам'яті, це модифікований однотранзисторний осередок, в якому знаходиться звичайний КМОП-транзистор і транзистор з плаваючим затвором. У цій структурі звичайний транзистор виконує роль ізолятора транзистора з плаваючим затвором від Bit Line .

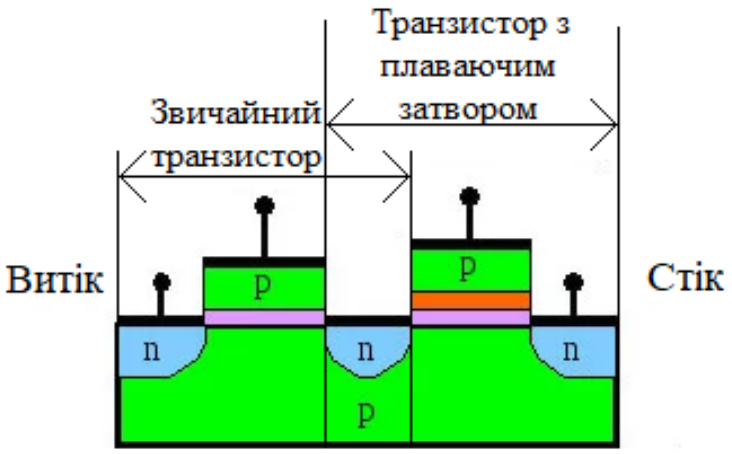


Рисунок 5.12 – Структура двотранзисторної комірки пам’яті

Основною перевагою двотранзисторного осередку пам'яті, є усунення головного недоліка схеми з одним транзистором. З її допомогою можна створювати компактніші і добре масштабовані мікросхеми пам'яті, тому що тут транзистор із плаваючим затвором ізолюється від бітової лінії. До того ж, на відміну від однотранзисторного осередку пам'яті, де інформація записується методом інжекції гарячих електронів, у двотранзисторного осередку пам'яті для запису та стирання інформації використовується метод квантового тунелювання Фаулера-Нордхейма. Такий підхід дає можливість знизити напругу, яка потрібна для операції запису.

Влаштування флеш-пам'яті з архітектурою NOR. Тип цієї пам'яті є джерелом і певним поштовхом у розвитку всієї EEPROM. Її архітектура була розроблена компанією Intel у далекому 1988 році. Як було написано раніше, щоб отримати доступ до вмісту комірки пам'яті (ініціалізувати комірку), потрібно подати напругу на затвор. Тому розробники компанії всі затвори приєднали до лінії керування, яка називається Word Line. Аналіз інформації осередку пам'яті виконується за рівнем сигналу стоку транзистора. Тому розробники всі стоки транзисторів приєднали до лінії, яка називається Bit Line.

Архітектура NOR отримала назву завдяки логічній операції АБО-НЕ (у перекладі з англійської NOR). Принцип логічної операції NOR у тому, що вона з

кількома операндами дає одиничне значення, якщо ці операнди рівні нулю, і нульове значення у всіх інших операціях. Під операндами мається на увазі значення осередків пам'яті, отже у цій архітектурі одиничне значення на бітової лінії спостерігається лише тому випадку, коли значення всіх осередків, які підключені до бітової лінії, дорівнюватимуть нулю (всі транзистори закриті).

У цій архітектурі добре організований довільний доступ до пам'яті, але процес запису та стирання даних виконується відносно повільно. У процесі запису та стирання застосовується метод інжекції гарячих електронів. До того ж мікросхема флеш-пам'яті з архітектурою NOR і розмір її осередку виходить більшим, тому ця пам'ять погано масштабується. Флеш-пам'ять з архітектурою NOR зазвичай використовують у пристроях зберігання програмного коду, його структурна схема зображена на рис. 5.13.

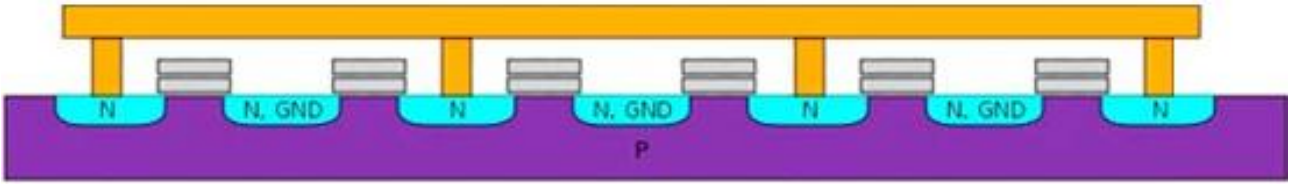


Рисунок 5.13 – Структура 6 комірок NOR-Flash

Пристрій флеш-пам'яті з архітектурою NAND. Цей тип пам'яті було розроблено компанією Toshiba, мікросхеми завдяки своїй архітектурі застосовують у маленьких накопичувачах, які отримали ім'я NAND (логічна операція І-НЕ). При виконанні операція NAND дає значення нуль тільки коли всі операнди рівні нулю та одиничне значення у всіх інших випадках. Як було написано раніше, нульове значення – це відкритий стан транзистора. В архітектурі NAND мається на увазі, що бітова лінія має нульове значення в тому випадку, коли всі підключені до неї транзистори відкриті і одиничне значення, коли хоча б один з транзисторів закритий. Таку архітектуру можна побудувати, якщо приєднати транзистори з бітовою лінією не по одному, а послідовними серіями, як показано на рис. 5.14.

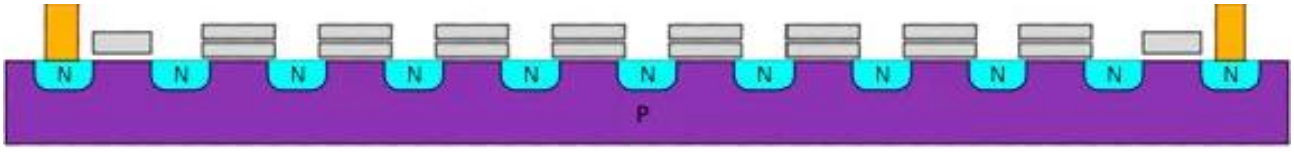


Рисунок 5.14 – Структура 6 комірок NAND-Flash

Згідно завдання та розробленої структурної схеми, flash-пам’ять, яка буде вміщати в себе операційну систему, оболонку та службову інформацію для коректної роботи системи. До пам’яті висувається такі умови:

– формат пам’яті microSD;
– об’єм пам’яті не менше 16 Гбайт;
– мінімальна швидкість запису 6 Мб/с;
– мінімальна швидкість читання 35 Мб/с.

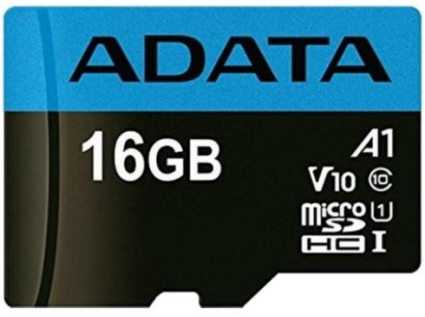
Розглянемо зведену табл. 5.1 основних характеристик, в якій опишемо декілька карт розширення.

Таблиця 5.1 – Порівняння характеристик flash-накопичувачів

Назва	Тип	Об’єм, Гб	Швидкість, Мб/с		Ціна
			Читання	Запис	
SanDisk Ultra A1	microSDHC	16	98	10	207
Apacer R85	microSDHC	16	85	85	215
A-Data Premier	microSDHC	16	100	25	198
Kingston	microSD	16	80	10	231
Transcend	microSD	16	10	10	298

Згідно табл. 5.1, нас задовольняє flash-пам’ять Premier, компанії A-Data. Вона має підходящі характеристики, а швидкість читання взагалі має найвище значення з представлених популярних брендів при мінімальній ціні.

Карта пам'яті Premier, від компанії A-Data, наведена на рис. 4.12, має повну сумісність з пристроями Android, а отже бездоганно буде працювати на Linux подібній системі в Raspberry Pi.



[

Рисунок 5.15 – Карта пам'яті A-Data Premier

5.3 Блок живлення

Згідно завдання та розробленої структурної схеми, для функціонування системи, нам необхідно підібрати блок живлення. Для стабільної роботи самого мікрокомп'ютера нам необхідно джерело струму приблизно 10 Вт, отже для всієї системи потрібно блок живлення з потужністю близько 15 Вт. Місце збереження користувацької інформації буде отримувати живлення через USB-порт змонтований на платі. Перерахуємо необхідні характеристики блока живлення для стабільної роботи системи:

- вихідна напруга 5В;
- вихідний струм 3А;
- пульсації та шум близько 150 мВ;
- вхідна напруга від 96 до 264 В;
- захист від короткого замикання;
- захист від перевантаження по струму;
- захист від перегріву.

Наведемо декілька популярних блоків живлення в табл. 5.2.

Таблиця 5.2 – Вибір блоку живлення

Характеристика	Raspberry Pi 15W	XOKO QC-100	BASEUS CCALL-BX02
Вихідна напруга, В	5,1	3,6 – 8	3,6 – 8
Вихідний струм, А	3	3,5	3
Потужність	15	18	15
Пульсації та шум, мВ	120	Не вказано	Не вказано
Вхідна напруга, В	96-264	200-260	100 – 240
Захист від КЗ	Є	Немає	Є
Захист від перевантаження	Є	Немає	Є
Захист від перегріву	Є	Немає	Є
Ціна	178	169	289

Згідно порівняльної таблиці, нам ідеально підійде фірмовий блок живлення від компанії Raspberry [23], він підходить нам по головним

параметрам, він від того ж виробника, що й платформа на якій будемо будувати систему, а отже менше проблем з поставками. що у виборі блоків живлення відіграє не останню роль [24]. На рис. 5.16 наведемо зовнішній вигляд блока живлення Raspberry Pi 15W.

Рисунок 5.16 – Raspberry Pi 15W

5.4 Конвертор SATA в USB

Конвертор SATA в USB. Для обміну інформацією між одноплатним комп’ютером та осередком пам’яті необхідно налагодити канал зв’язку, так як в Raspberry Pi 3 B+ немає стандартизованого порту SATA. В одноплатному комп’ютері є швидкі порти USB, ними й скористаємося. Спроектуємо спеціальний проміжний конвертор інформації, для цього будемо опиратись на спеціалізовану мікросхемі FT232BM.

Обрана спеціалізована мікросхема FT232BM це представник другого покоління перетворень від компанії FTDI Chip, її структурна схема зображена на рис. 5.17. Друге покоління розроблялося на основі першого покоління з блоком FT8U232AM який реалізує двонаправлену передачу інформації з протоколу USB в UART і навпаки. Можливості функціонування були вдосконалені завдяки новому режиму Bit Bang, тепер інтерфейс виходу мікросхеми дозволяє реалізувати до восьми незалежних лінійних введення та виведення.

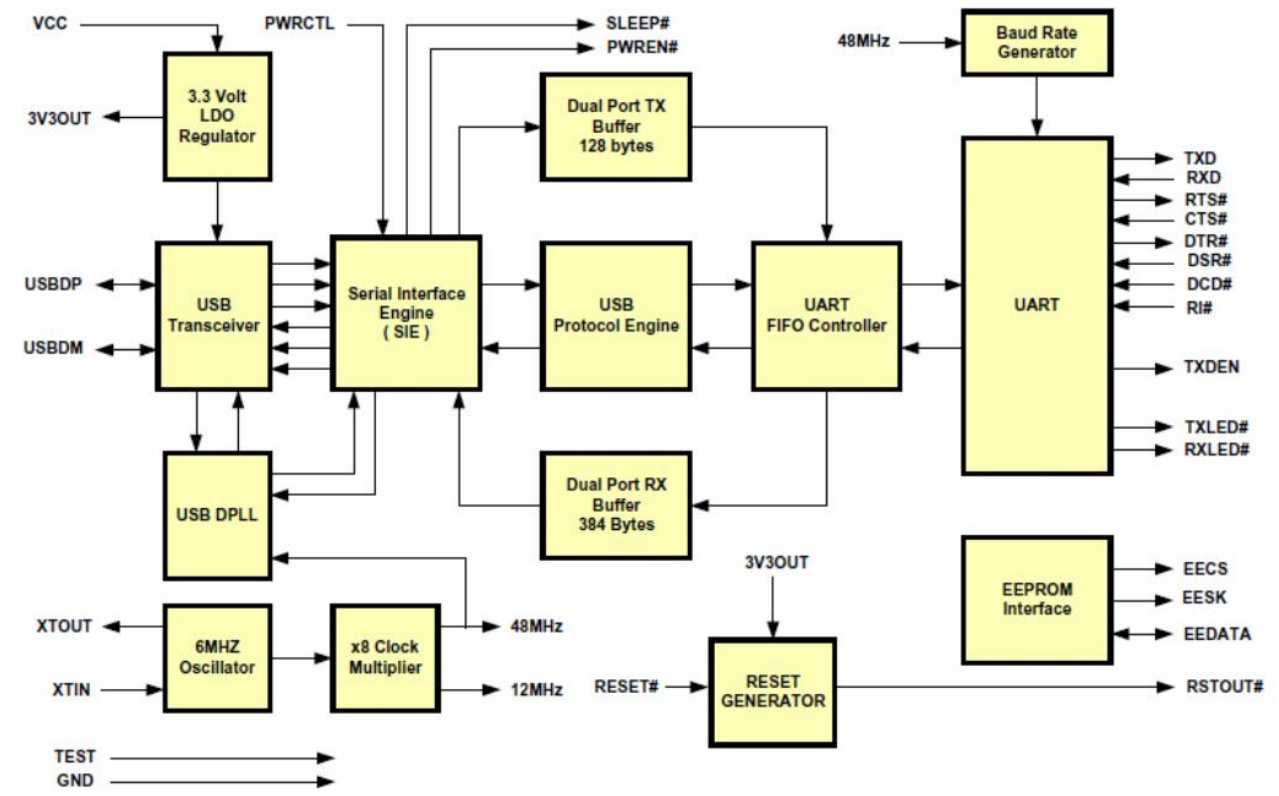


Рисунок 5.17 – Структурна схема FT232BM

Оновлений чіп FT232BM забезпечує передачу даних по одній із ліній з швидкістю до 2000 кБіт/с. Мікросхема має вбудований конвертер рівнів для обробки UART-сигналів на 5В логічних рівнях CMOS. Є також можливість підключати 3.3 вольтову логіку, для цього треба подати сигнал на відповідний контакт VCC-IO, таким чином використання будь-яких зовнішніх конверторів рівнів відпадає.

У новій ревізії чіпа, система управління живленням високовольтного обладнання була значно вдосконалена. У мікросхемах попереднього покоління контакт USBEN залишався активним при підключенні до пристрою USB. Для управління живленням сигнал повинен бути підключений до контактів SLEEP # і RESET # через зовнішній зв'язок. Завдяки такому вентилю, який тепер реалізовано в середині мікросхеми, сигнал PWREN# можна використовувати для безпосереднього керування транзистором або p-канальним МОП-транзистором у програмах, які вимагають перемикання живлення у зовнішньому ланцюзі. Завдяки функції EEPROM лінія інтерфейсу UART може бути повільно відключена при вимкненні живлення. У цьому режимі, коли живлення вимкнено, будь-яка напруга, що залишилася на зовнішньому ланцюзі, відводиться на землю,

забезпечуючи надійний перезапуск зовнішнього ланцюга через пін PWREN# після відновлення живлення. Час затримки буферу приймача в мікросхемі FT232BM може бути запрограмована в діапазоні від 1 до 255 мс з кроком 1 мс. Завдяки цьому, пристрій може бути оптимізовано для протоколів, що вимагають малого часу відклику для малого обсягу переданих даних.

Мікросхема виробляється тільки у форматі LQFP, який зображений на рис. 5. 18, що дещо ускладнює розробку і виготовлення друкованих плат, але при цьому готова схема буде займати найменше корисного простору, мікросхема показана на рис 5. 18.

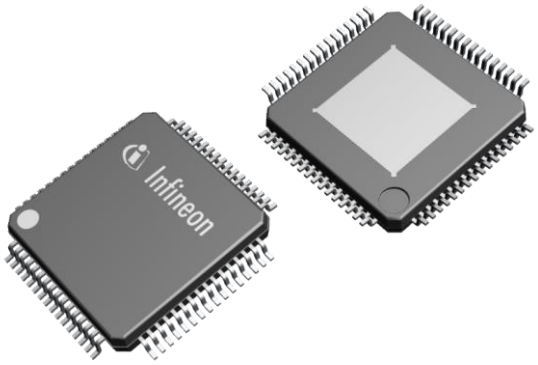


Рисунок 5.18 – Демонстрація чіпу формату LQFP

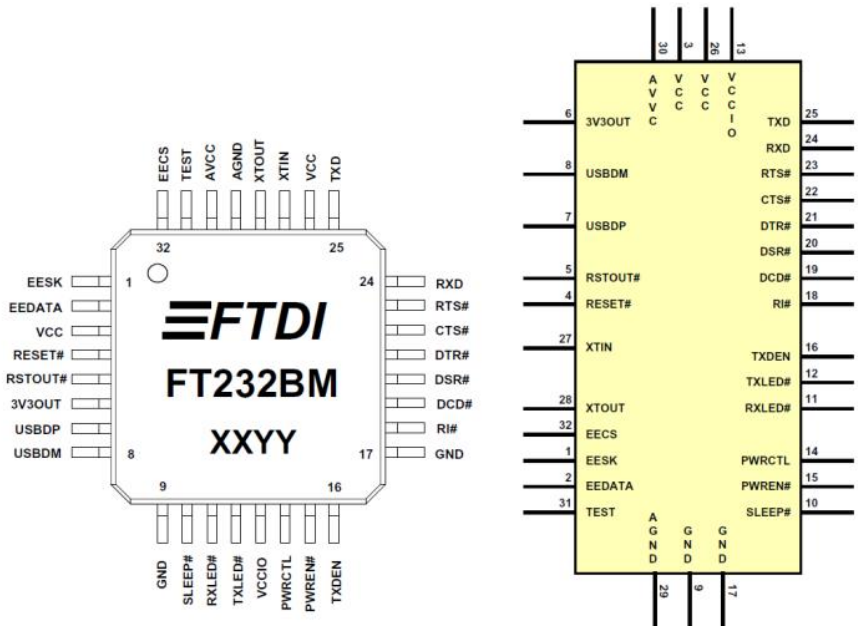


Рисунок 5.19 – Розпіновка корпусу та схематичне відображення FT232BM

За рис. 5.19 та технічною документацією, складемо табл. 5.3, де опишемо всі доступні піни мікросхеми FT232BM та коротко опишемо їх функції.

Таблиця 5.3 – Перелік та призначення портів FT232BM

№	Назва	Тип	Опис
1	EESK	Out	Сигнал лічильника на EEPROM. Додавання 10кОм резистора на EESK призведе до того, що FT232BM використовуватиме USB-ідентифікатор продукту 6004
2	EEDATA	I/O	EEPROM – введення та вивід даних безпосередньо підключається до входу даних EEPROM і до виходу даних EEPROM через резистор 2,2кОм.
3,26	VCC	pwr	Від +4,35В до +5,25В
4	RESET#	In	Активний пін скидання низьким сигналом.
5	RSTOUT	Out	Вихід внутрішнього генератора скидання. Залишається високим імпедансом протягом ~5 мс після VCC > 3,5 В
6	3V3OUT	Out	3,3 вольта на виході від вбудованого регулятора LDO. Його основна мета - забезпечити внутрішнє живлення 3,3 В до комірки USB-трансивера та виводу RSTOUT #.
7	USDP	I/O	USB Data Signal Plus
8	USDM	I/O	USB Data Signal Minus
9,17	GND	pwr	Пристрій заземлення пінів
10	SLEEP#	Out	Режим призупинення USB від низького рівня.
11	RXLED	O.C	Вхід для світлодіода – індикатор при отриманні даних через USB
12	TXLED	O.C	Вхід для світлодіода – індикатор при передачі даних через USB
13	VCCIO	pwr	+3,0 В до +5,25 В VCC до контактів інтерфейсу UART 10 ... 12, 14 ... 16 і 18 ... 25.
14	PWRCT	In	Шина з живленням
15	PWREN#	Out	Низький рівень після налаштування пристрою через USB, потім високий під час призупинення роботи USB.
16	TXDEN	Out	Увімкнення передачі даних для RS485.
18	RI#	In	Вхід управління кільцевим індикатором.
19	DCD#	In	Вхід управління виявлення носіїв даних.
20	DSR#	In	Набір даних Data Set Ready / сигнал рукостискання.
21	DTR#	Out	Data Terminal Ready Control / сигнал рукостискання.
22	CTS#	In	Clear To Send Control / сигнал рукостискання.
23	RTS#	Out	Request to Send Control/ сигнал рукостискання.
24	RXD	In	Отримання асинхронних даних.
25	TXD	Out	Передача асинхронних даних.
27	XTIN	In	Вхід для кристалічного генератора 6 МГц.
28	XOUT	Out	Вихід для кристалічного генератора 6 МГц. XTOUT перестає коливатися під час призупинення роботи USB.
29	AGND	pwr	Пристрій аналогового заземлення для внутрішнього тактового множника x8
30	AVCC	pwr	Пристрій аналогового джерела живлення для внутрішнього тактового множника x8
31	TEST	In	Переводить пристрій у режим перевірки мікросхеми. Повинен бути прив’язаний до GND для нормальної роботи.
32	EECS	I/O	EEPROM - вибір мікросхеми. Для роботи 48 МГц підтягніть EECS до GND за допомогою резистора 10 кОМ.

Уважніше розглянемо основні функції, які доступні в FT232BM.

+ 3.3V LDO регулятор. LDO-регулятор генерує опорну напругу 3,3 В для керування вихідними буферами трансиверного блоку USB. Для цього потрібен зовнішній розв'язувальний конденсатор, який необхідно підтягнути до вихідного контакту регулятора «3V3OUT». Він також забезпечує 3,3 В на контакті "RSTOUT #". Основною функцією цього пристрою є живлення USB-трансивера та скидання генератора, а не зовнішня логіка. Однак зовнішні схеми, які потребують номінальної напруги 3,3 В і не більше 5 мА, також можуть отримувати струм через пін «3V3OUT», якщо це необхідно.

USB трансивер. Блок трансивера USB забезпечує швидкісний фізичний інтерфейс USB 2.0/USB3.0 для каналу зв'язку. Вихідний драйвер забезпечує сигнал керування +3,3 В, а диференціальний вхідний приймач і два односторонніх вхідних приймача забезпечують USB-дані в односторонньому режимі 0 (SE0) і виявленні USB. Ця функція також включає внутрішні кінцеві резистори USB на лініях передачі даних USB і підтягуючий резистор 1,5 кОм на USBDP.

Внутрішній генератор 12 МГц. Внутрішній блок генератора 12 МГц генерує тактовий сигнал керування 12 МГц. Це забезпечує вхідні дані для функції множника тактової частоти x4. Генератор 12 МГц також використовується як опорний лічильник для блоків контролера SIE, протоколу USB і UART FIFO.

Дільники частоти та помножувачі частоти. Помножувач або дільник тактової частоти працює від входу внутрішнього генератора 12 МГц і генерує опорні тактові частоти 6 МГц, 12 МГц, 24 МГц та 48 МГц. Щоб генерувати тактову частоту 48 МГц, нам потрібен модуль USB DPLL і генератор швидкості передачі даних.

Осередок USB DPLL. USB DPLL підключається до входу USB NRZI. Потім він виводить відновлені дані та лічильники на блок послідовного інтерфейсу (SIE).

Модуль Serial Interface Engine або SIE може обробляти послідовну обробку даних USB паралельно. У специфікації USB 2.0 зазначено, що він виконує генерацію CRC5 і CRC16 і вставлення бітів. Він також перевіряє CRC потоку даних USB.

Механізм протоколу USB контролює потоком даних через кінцеву точку керування USB пристрою, обробляючи запити протоколу USB низького рівня,

які створені контролером хосту USB, а також розділ 9 функціональних команд специфікації USB 2.0 для керування параметрами UART.

Буфер FIFO RX (128 байт). 128-байтний буфер RX FIFO USB data OUT зберігає дані, надіслані з хост-контролера USB до UART через кінцеву точку USBdata OUT. Ці дані видаляються з буфера FIFO RX під керуванням контролера FIFO даних UART. Зберігаючи дані в цьому буфері, дані можна надсилати між двома пристроями через хост-контролер USB.

Буфер FIFO TX (256 байт). При отриманні даних через UART регістр прийому хост-контролера USB зберігає дані з буфера TX. Після отримання даних із кінцевої точки пристрою контролер хосту очищає буфер передачі та запитує дані з нього, надсилаючи запит USB. Це дозволяє контролеру хоста очистити дані з буфера передачі та представити чистий буфер для подальшої передачі даних. Дані, якими обмінюються буфери RX FIFO і TX FIFO і регістри прийому та передачі UART, обробляються контролером UART FIFO.

Контролер UART. Контролер UART обробляє передачу даних через буфери RX FIFO і TX FIFO, а також через регістри передачі та прийому UART. Він працює як програмований 8- або 7-бітний паралельно-послідовний перетворювач на інтерфейсі RS232, 422 або 486. Крім того, він підтримує інверсію сигналів даних, що надсилаються до та з регістрів. Апаратне підключення до конфігурації хоста забезпечує швидкий час відгуку. Режим UART підтримує керуючі сигнали Control, RTS, CTS, DSR, DTR, DCD і RI. Крім того, підтримуються параметри рукостискання XON / XOFF. Апаратне підключення забезпечує швидкий час відгуку. UART також підтримує параметри та індикатори BREAK RS232. Сигнали UART можна інвертувати індивідуально та надавати більшу потужність приводу за допомогою сконфігурованої опції, яка зберігається в EEPROM.

USB-пристрої з автономним живленням використовують власне джерело живлення та не підключаються до шини USB. На рис. 5.20 наведена мікросхема FT232R в базовій конфігурації з автономним живленням USB. Основні правила для цих пристроїв такі:

– щоб працювати з автономним живленням, пристрій не повинен пропускати струм із шини USB, коли контролер USB-хосту або концентратора від’єднується;

– пристрій з автономним живленням може природно використовувати будь-який необхідний струм без перерви. Під час нормальної роботи це також дозволяє пристрою призупиняти USB;
– пристрій з автономним живленням можна підключити до будь-якого USB-хосту, активного USB-концентратора або USB-концентратора.

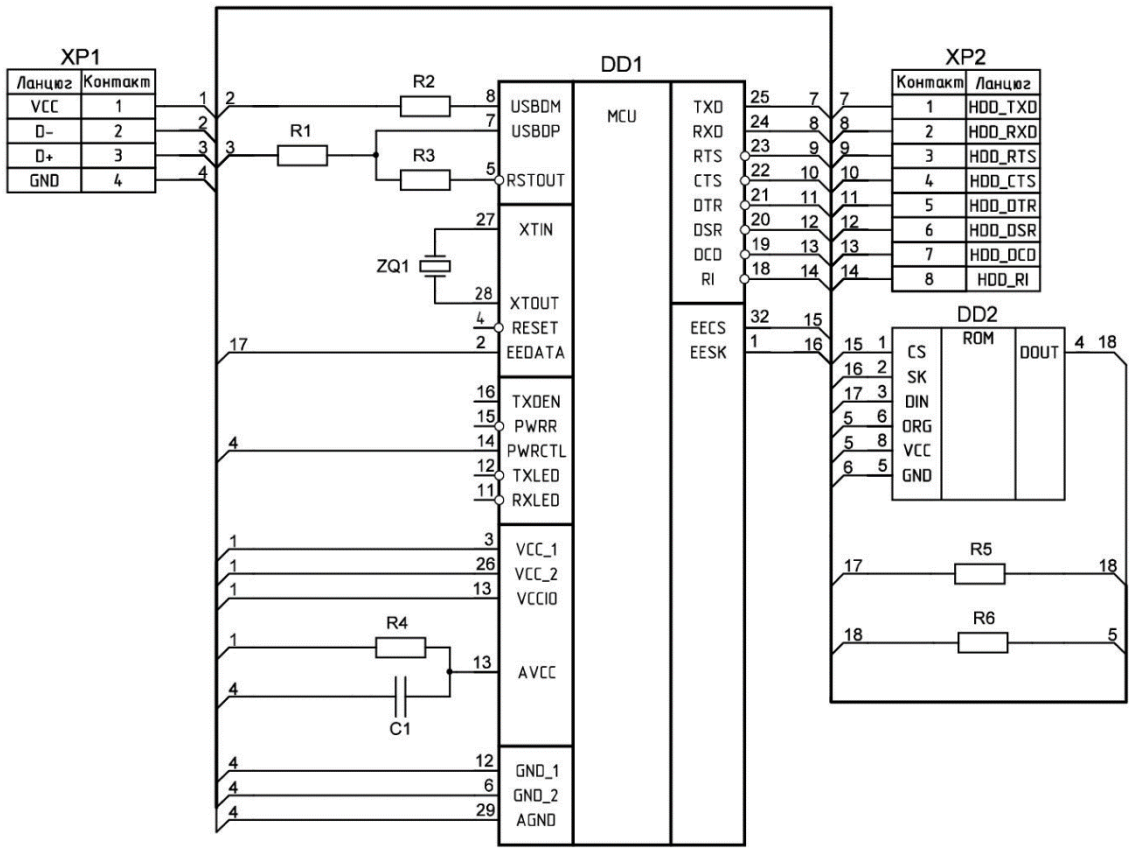


Рисунок 5.20 – Принципова схема конвертора

Дескриптор потужності у внутрішній EEPROM FT232R повинен бути запрограмований на нуль (з використанням автономного джерела живлення). Щоб задовольнити першу вимогу яка викладена вище, живлення шини USB (контакт 1) використовується для керування контактом RESET# FT232R. При підключенні до джерела живлення або концентратора USB USBDP бачитиме напругу до +3,3 В (генерується за допомогою резисторів 4700 Ом і 10 кОм), ідентифікуючи пристрій як швидкісний USB-хост або пристрій-концентратор. Коли USB-концентратор або концентратор вимкнено, RESET# буде низьким, а FT232R залишиться в скинутому стані. Оскільки RESET# низький, внутрішній резистор 1,5 кОм не підтягується до будь-якого джерела живлення, тому струм

через USBDP через підтягувальний резистор 1,5 кОм не проходить. Якщо цього не зробити, деякі хост-контролери або USB-концентратори не зможуть забезпечити живлення системи. На рис. 5.20 зображена конструкція з джерелом живлення від +4 В до +5,5 В.

5.5 Пристрій збереження інформації HDD

Жорсткий диск (скорочено HDD) – це тип накопичувача, який зазвичай використовується як основна система зберігання як для портативних, так і для настільних комп'ютерів. Він функціонує як будь-який інший тип цифрового запам'ятовуючого пристрою, записуючи біти даних, а потім викликаючи їх у відповідному порядку.

Варто зазначити, що жорсткий диск – це енергонезалежна пам'ять, що означає, він може зберігати дані без джерела живлення. Ця особливість у поєднанні з великою ємністю для зберігання даних і відносно низькою вартістю є причинами, чому жорсткі диски так часто використовуються як в домашніх комп'ютерах, серверних масивах та їх не витіснили сучасні SSD. Незважаючи на те, що жорсткі диски пройшли довгий шлях з моменту їх винайдення, основний спосіб їх роботи залишився незмінним. Фундаментально HDD вміщує в себе наступне:

- плата керування;
- шпиндельний двигун;
- блок магнітних головок;
- магнітний диск.

Плата керування, являє собою систему обслуговування магнітного диску, на якому і зберігається вся інформації. Вона й управляє роботою жорсткого диску, перетворює сигнали отримані з магнітних головок на зрозумілий центральному процесору корисний сигнал, передає цей сигнал за технологією SATA, керує швидкістю обертання шпиндельного двигуна, на якому розміщені магнітні пластини та керує позиціювання магнітної головки на площині магнітної пластини для запису, стирання або ж перегляду збереженої інформації. Для виконання всіх завдань, плата HDD має власний процесор, постійну та оперативну пам'ять, плата керування диска зображена на рис. 5.21 [25].

Рисунок 5.21 – Плата керування жорстким диском

Усередині корпусу є ряд дископодібних об’єктів, які називаються магнітними пластинами HDD, зображені на рис. 5.22. Це фундаментальний елемент жорсткого диску, саме тут зберігається вся користувацька інформація. Магнітна пластина має круглу форму, для зменшення вібрацій при роботі, зроблена з алюмінію або скла і покриту тонким шаром феромагнітного матеріалу, як правило, двоокисом хрому. Верхній магнітний шар, саме завдяки йому магнітна головка може записувати дані в двійковій формі.

Рисунок 5.22 – Магнітні пластини HDD

Для швидкого запису та читання, магнітні пластини обертає навколо своєї осі шпиндельний двигун, структурна схема якого зображена на рис. 5.23. За

керування цього двигуна відповідає плата керування, вона може динамічно змінювати швидкість обертання від 5400 до 10000 обертів на хвилину.

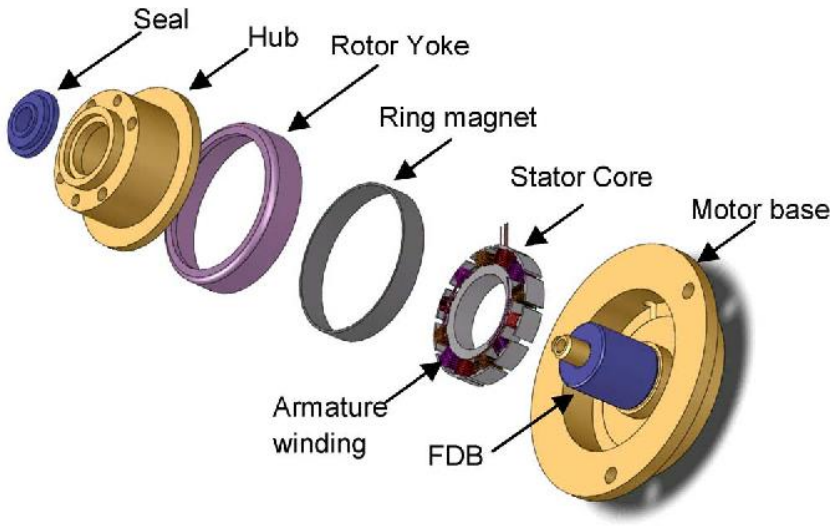


Рисунок 5.23 – Шпиндельний двигун HDD

Блок магнітних головок, який зображений на рис. 5.24, складається з мікросхеми підсилювача, комутатора та алюмінієвої направляючої, на кінці яких розташовані магнітні головки. За допомогою цих магнітних головок і відбувається зчитування та запис інформації на пластину.

Рисунок 5.24 – Блок магнітних головок

ЦП і материнська плата використовують програмне забезпечення, щоб повідомляти так званій блоку магнітних головок куди рухатися на пластині та де вона потім забезпечує електричний заряд для «сектору» на пластині. Кожен сектор є ізольованою частиною диска, що містить тисячі підрозділів, усі здатні приймати магнітний заряд. На рис. 5.25 зображена структурна схема яка

відображає процес збереження та відтворення інформації. Новіші жорсткі диски мають розмір сектора 4096 байт або 32768 біт, магнітний заряд кожного біта перетворюється на двійкові 1 або 0 даних.

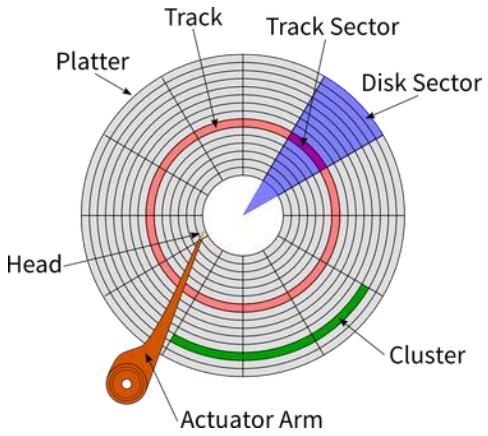


Рисунок 5.25 – Структурна схема блоку збереження/відтворення інформації

У міру розробки жорстких дисків одним із ключових факторів, який змінився, є орієнтація секторів на пластині. Жорсткі диски спочатку були розроблені для «поздовжнього запису», тобто більша сторона пластини орієнтована горизонтально, але з тих пір використовується інший метод під назвою «перпендикулярний запис», коли сектори складаються один на один. Цю зміну було внесено, оскільки виробники жорстких дисків досягли межі щодо того, наскільки маленьким вони могли зробити кожен сектор через «суперпарамагнітний ефект». Одним із недоліків перпендикулярного запису є підвищена чутливість до магнітних полів і помилка читання, що створює потребу в більш точних плечах читання/запису. Структурна схема процесу запису інформації на HDD зображена на рис. 5.26.

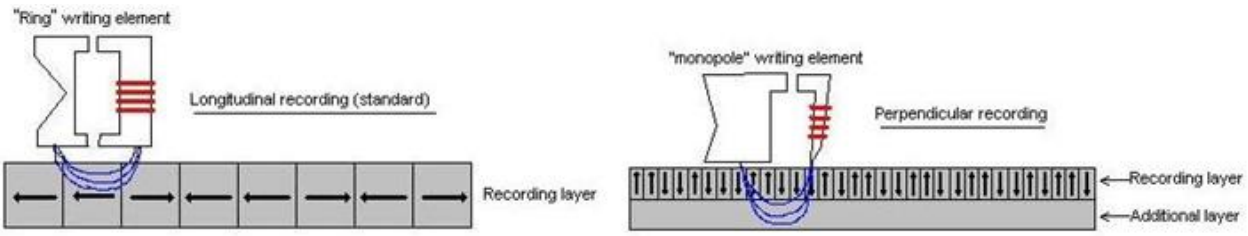


Рисунок 5.26 – Структурна схема запису на HDD

По суті, супер парамагнітний ефект означає, що сектори жорсткого диска, менші за певний розмір, змінюватимуть магнітний заряд випадковим чином

залежно від температури. Це явище може призвести до неточного зберігання даних, особливо враховуючи тепло, яке виділяє робочий жорсткий диск.

Тепер, коли ми обговорили фізичну роботу жорсткого диска, ми можемо поглянути на відмінності в тому, як операційні системи, такі як Windows або Linux, використовують диск. Однак заздалегідь важливо згадати загальну проблему зберігання даних, яка певною мірою виникає в усіх згаданих вище операційних системах.

Фрагментація диска відбувається після періоду зберігання та оновлення даних на диску. Наприклад, якщо оновлення не зберігається безпосередньо після базової програми, існує велика ймовірність того, що на диску збережено щось інше. Тому оновлення для програми потрібно буде розмістити в іншому секторі, подалі від основних файлів програми. Через фізичний час, необхідний плечу читання/запису для переміщення, фрагментація може зрештою суттєво сповільнити вашу систему, оскільки плече потребуватиме посилань на все більше і більше окремих частин на вашому диску. Більшість операційних систем мають вбудовану програму, призначену для «дефрагментації» диска, яка просто перевпорядковує дані таким чином, щоб усі файли однієї програми були в одному місці. Процес триває довше залежно від того, наскільки фрагментованим став диск. Тепер ми можемо обговорити різні протоколи зберігання та те, як вони впливають на фрагментацію.

Windows використовує базову комп’ютерну мову під назвою MS-DOS (дискова операційна система Microsoft) і систему керування файлами під назвою NTFS або нову технологічну файлову систему, яка є стандартом для компанії з 1993 року. Коли надходить інструкція запису, файл NT система розмістить інформацію якомога ближче до початку диска/пластини. Хоча ця методологія є функціональною, вона залишає лише невелику буферну зону між різними файлами, що зрештою спричиняє фрагментацію. Через невеликий розмір цієї буферної зони Windows, як правило, найбільш сприйнятлива до фрагментації.

Linux це операційна система з відкритим кодом, що означає, що існує багато різних її версій, які називаються дистрибутивами, для різних програм. Найпоширеніші дистрибутиви, такі як Ubuntu, використовують файлову систему ext4. Linux має найкраще рішення щодо фрагментації, оскільки він розподіляє файли по всьому диску, даючи їм усім достатньо місця для збільшення розміру, не заважаючи один одному. Якщо файлу потрібно більше місця, операційна

система автоматично намагатиметься перемістити файли навколо нього, надавши йому більше місця. Особливо враховуючи ємність більшості сучасних жорстких дисків, ця методологія не є марнотратною та не призводить до фрагментації в Linux, доки ємність диска не перевищить приблизно 85%.

Всі вищеперераховані елементи впливають на робочі характеристики HDD, а саме:

- об’єм;
- форм-фактор;
- час доступу до файлів;
- швидкість запису даних;
- швидкість обертання шпинделя;
- інтерфейс підключення;
- об’єм кеш-пам’яті.

З всього переліку параметрів, ми будемо опиратися на основні, а саме: форм-фактор, об’єм та швидкість передачі даних. Для цього розглянемо табл. 5.4, в якій наведені популярні представники ринку.

Таблиця 5.4 – Вибір HDD

Назва	Seagate ST1000DM010	WD Blue WD10EZEX	Toshiba L200	WD Blue WD10SPCX
Об'єм, Гб	1000	1000	1000	1000
Форм-фактор	3,5	3,5	2,5	2,5
Швидкість передачі даних, Мб/с	156	150	140	135
Швидкість обертання шпинделя, об/хв	7200	7200	5400	5400
Кеш-пам’ять, Мб	64	64	128	16
Ціна, грн	1669	1557	1552	1799

Згідно табл. 5.4, нас задовольняє жорсткий диск фірми WD Blue, модель WD10EZEX. Він задовольняє нас за основними та вторинними параметрами, також безсумнівною перевагою є ціна продукції

6 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРИСТРОЮ ХМАРНОГО СХОВИЩА

6.1 Налаштування оболонки

NextCloudPi це операційна система для одноплатного комп'ютера Rasbperry Pi. Її особливістю є те, що вона призначена саме систем з NAS орієнтуванням та підходить до всієї лінійки комп'ютерів Raspberry Pi. NextCloudPi оболонка поставляється з власним інсталятором New Out Of Box Software (NOOBS) [26].

Для початку роботи та налаштування, нам необхідно отримати копію операційної системи. NOOBS поставляється в двох форматах:

- універсальний, офлайн інсталятор та онлайн інстялятор;
- легкий, тільки онлайн інсталятор.

Універсальний інсталятор вже має ОС Raspbian, тому її можна встановити з usb-флешки або SD-карти в автономному режимі. Для встановлення системи легким пакетом, система повинна мати доступ до мережі.

Починаємо працювати з NOOBS v1.3.10, за стандартними налаштуваннями в універсальному пакеті, в першу чергу встановлюється автономна версія Raspbian. Інші оболонки можуть бути встановлені лише за умови підключення комп'ютера до мережі інтернет, робоче вікно ОС Raspbian, наведене на рис. 6.1

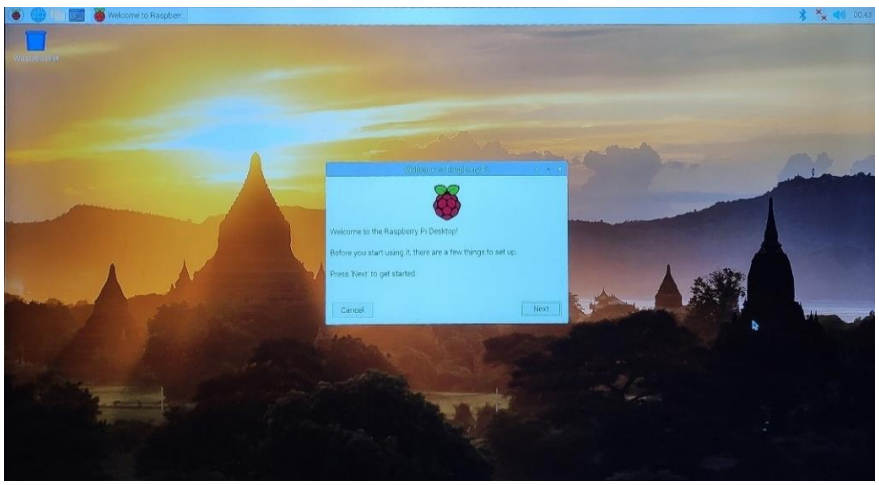


Рисунок 6.1 – Робоче вікно Raspbian

NextCloudPi – це безкоштовний проект, спільноти з відкритим вихідним кодом, оболонки Nextcloud, що вміщає свій власний інтерфейс керування з усіма необхідними інструментами, для розміщення персональних даних та зручного доступу до них.

Проект NextCloudPi вигідно відрізняється від конкурентів, бо користування ним не вимагає від користувача глибоких знань роботи з терміналом ядра Linux. Для взаємодії користувача з системою, був розроблений графічний інтерфейс, установка й налаштування якого займає на багато менше часу.

Підготовка до встановлення операційної системи. Для початку нам необхідно завантажити інсталятор оболонки NextCloudPi. Для цього перейдемо на офіційний сайт проекту ownyourbits.com [27], його головна сторінка зображена на рис. 6.2. У відповідному розділі виконуємо завантаження архіву формату «bz2» в якому запакований образ системи.

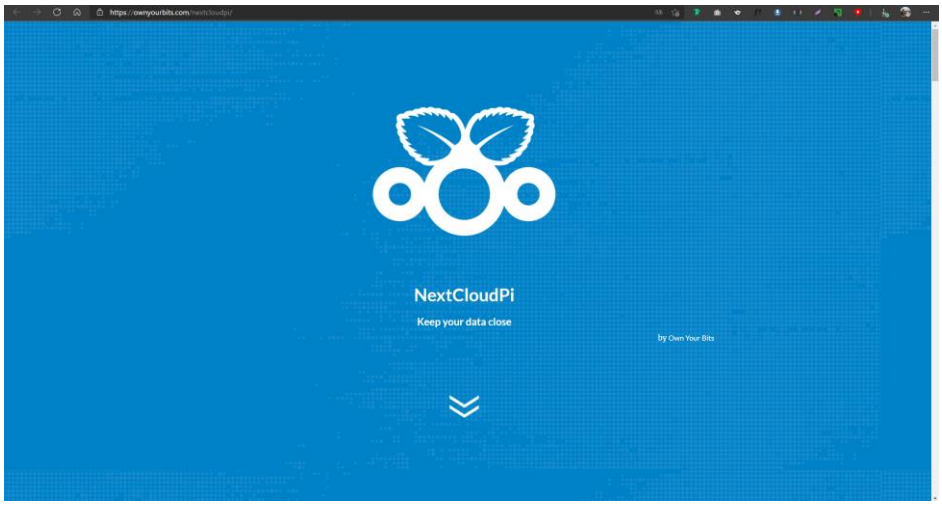


Рисунок 6.2 – Сайт NextCloudPi

Тепер, коли ми маємо образ системи, ми повинні підготувати сховище, на яке буде встановлене програмне забезпечення, в нашому випадку це SD-карта A-Data Premier. Для цього необхідно запустити спеціалізоване програмне забезпечення Rufus версії 3.14. Цей продукт створений для роботи з усіма типами flash-накопичувачі, дозволяє форматувати сховище в іншу файлову систему з користувацьким розміром кластеру та дозволяє обрати схему розділу [28]. За допомогою нього, завантажуємо образ NextCloudPi на flash-накопичувач, налаштування програми для цього зображені на рис. 6.3.

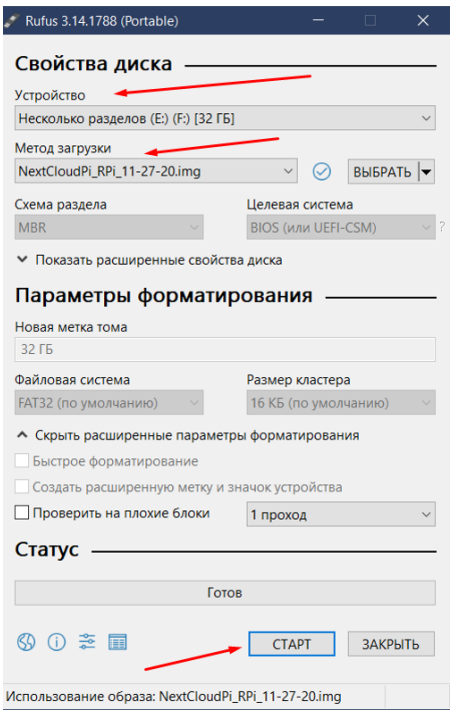


Рисунок 6.3 – Інтерфейс програми RUFUS

Після вдалого завантаження образу NextCloudPi ми можемо зібрати та протестувати систему. Зібраний пристрій зображений на рис. 6.4, також наведемо алгоритм збірки:

– під’єднуємо HDMI кабель для виводу зображення на дисплей;
– під’єднуємо клавіатуру через USB;
– під’єднуємо флешку для тестового збереження файлів.
– під’єднуємо кабель живлення до порту micro-usb .

Рисунок 6.4 – Збірка для тестування та налаштування

Так як це перше завантаження системи, то воно може зайняти до 10 хвилин, процес завантаження та налаштування зображений на рис. 6.5 [29].

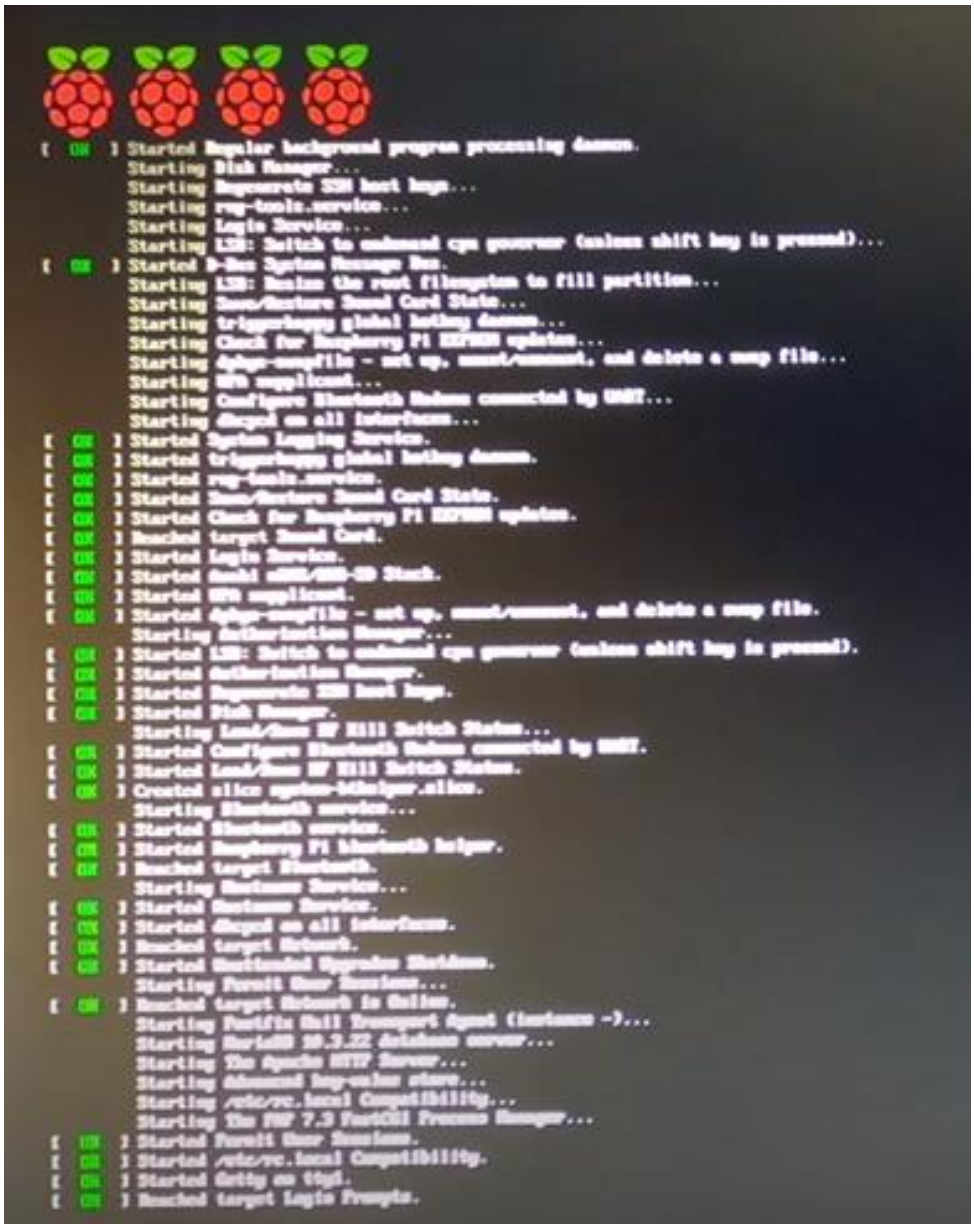


Рисунок 6.5 – Перше завантаження операційної системи

Під час налаштування, наладчику потрібно авторизуватися в системі через linux-термінал. Для цього, згідно технічної документації до NextCloudPi, потрібно використати стандартний логін «pi» та пароль «raspberry». Для забезпечення безпеки користувацьких файлів, ці дані необхідно змінити на унікальні. Тепер наладчик надає права доступу super_user для raspi-config, використовуючи термінальну команду **sudo raspi-config.** Приклад вдалого виконання команди зображений на рис. 6.6.

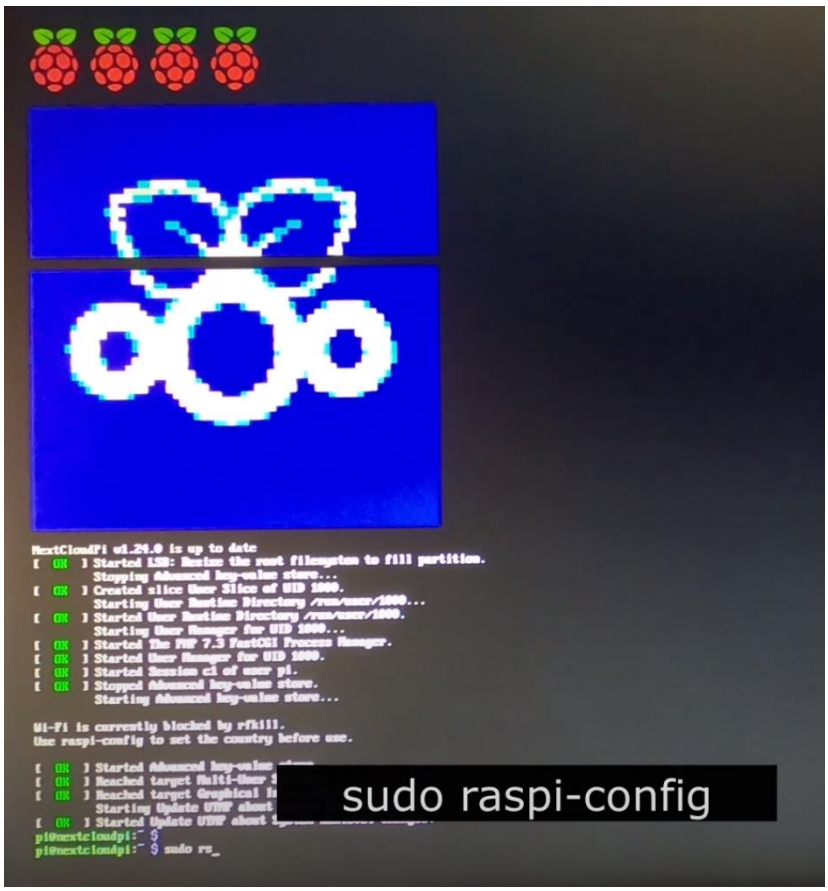


Рисунок 6.6 – Надання прав super_user

Це дозволяє продовжити первинне налаштування пристрою. Тепер нам доступний список конфігурацій, який зображені на рис. 6.7, які дозволено редагувати.

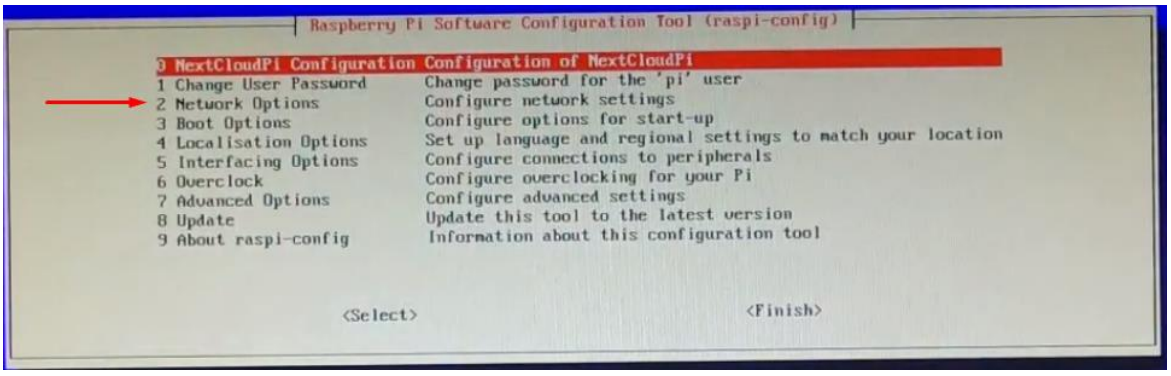


Рисунок 6.7 – Вікно raspi-config

Переходимо у пункт Network Options та обираємо підпункт Wi-Fi, як зображена на рис. 6.8. В цьому підменю, нам необхідно обрати країну, в якій буде використовуватись пристрій. Це необхідно для налаштування прийомо-

передавача на Rasbperry Pi, конкретніше – їх робочі частоти. Це необхідна процедура, бо в різних країнах існують різні обмеження в сфері частот передачі інформації.

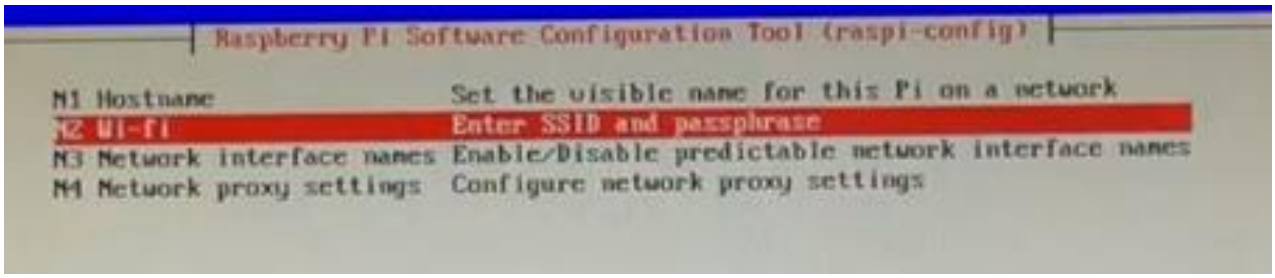


Рисунок 6.8 – Вибір налаштуваннь Wi-Fi

Коли з низинними налаштуваннями закінчено, потрібно ввести назву точки доступу та пароль від неї. Це потрібно для виключення з системи габаритного дисплею, так як він не входить в систему, а використовувався лише для її налаштування. Подальшу роботу та налаштування буде проводитись через технологію SSH.

SSH технологія. Secure Shell це мережевий протокол, який дозволяє налаштувати безпечний канал зв’язку через мережу інтернет, завдяки попередній спільній згоди про те, як буде проводитися обмін інформацією. SSH відноситься до прикладного рівня моделі OSI.

SSH в наших цілях є незмінним помічником, тому що це дає нам змогу безпечно підключатися до системи з будь-якої точки світу, де є доступ до інтернету, візуалізація цього зображена на рис. 6.9. Також, це зручно для налаштування, так як немає потреби знаходитись в безпосередній близкості до одноплатного комп’ютера [30].

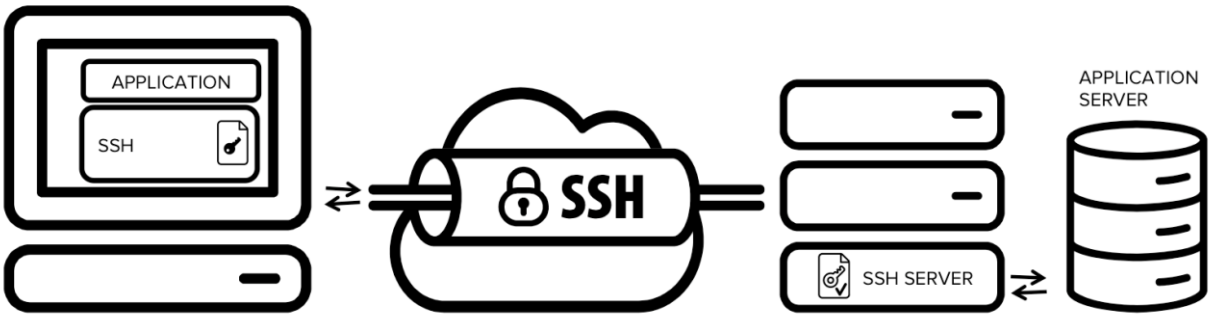


Рисунок 5.10 – SSH тунель

Безпека передачі даних забезпечується шифруванням, SSH був розроблений для безпечного способу передачі даних, як заміна застарілій технології TelNet. SSH зашифрує користувацькі дані та передає їх через «тунель», зловмисники при передачі, можуть лише побачити факт передачі та об’єм переданої інформації, але не можуть переглянути та розшифрувати інформацію.

В стандартному сценарії використання технології SSH, застосовується модель клієнт – сервер. Один пристрій в системі виступає в ролі сервера та приймає запити, а інший виступає в ролі клієнта SSH. Коли клієнт намагається отримати доступ до серверу SSH, він потрапляє в серверну оболонку. Вона може мати вигляд терміналу Linux або командного рядка Windows, в ній користувач ви взаємодії з сервером, до якого підключений.

Для безпосереднього налаштування NextCloudPi, потрібно під’єднатися до розгорнутого серверу на потужностях Raspberry Pi за протоколом SSH. Вводимо в адресний рядок браузеру IP-адресу серверу та отримуємо дані для входу, які зображені на рис. 6.9. Перший логін та пароль відповідають за обліковий запис адміністратора, там він зможе проводити унікальні налаштування, формувати групи доступу, додавати нових користувачів зі своїми правами доступу і так далі. Другий логін та пароль відповідають за стандартний користувацький доступ до сховища.

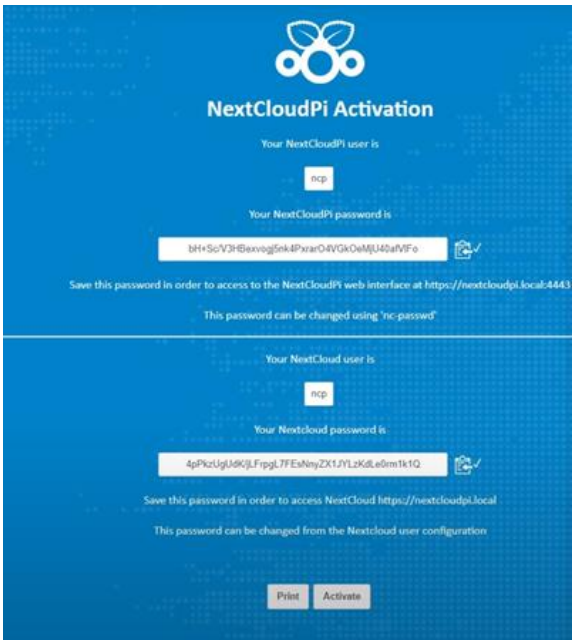


Рисунок 6.9 – Паролі для конфігурації та використання NextCloudPi

При першому запуску інтерфейсу NextCloudPi від адміністративного облікового запису, відкриється форма «швидкого налаштування». Ця форма зображена на рис. 6.10.

Після стандартного привітання, відкриється форма налаштування портів USB – USB configuration. Інтерактивне меню відобразить список підключених пристроїв для збереження інформації та попросить обрати, які використовувати для запису користувацьких файлів. Після вибору пристрою, користувач буде попереджений, про форматування диску та втрату всієї інформації в сховищі, це необхідно для встановлення файлової системи.

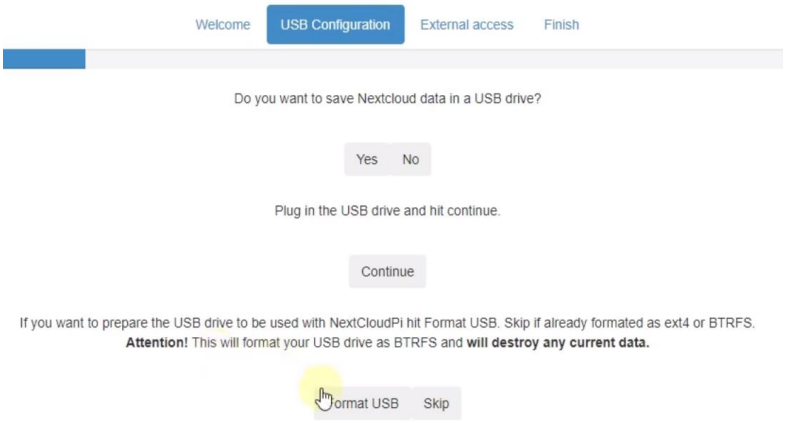


Рисунок 6.10 – Підготовка пристрою збереження даних

Другий пункт – External access, її форма зображена на рис. 6.11. Цей пункт налаштувань, відповідає за доступ до сховища за межами локальної мережі. Пункт налаштувань Port forwarding дає можливість налаштувати параметри захисту потоку інформації, який буде перенаправлятися на захищений порт по протоколу HTTP.

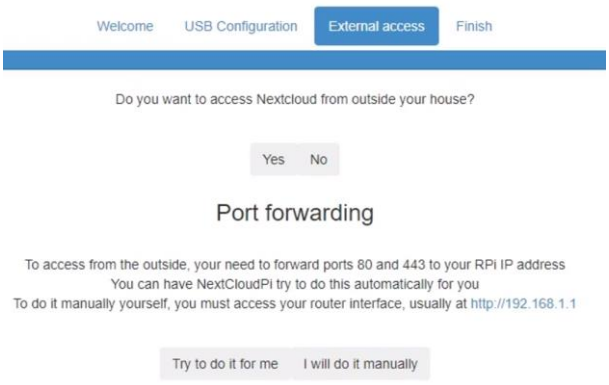


Рисунок 6.11 – Налаштування захисту даних

HTTP це набір правил для передачі файлів, таких як текст, зображення, звук, відео та інші мультимедійні файли через мережу інтернет. Щойно користувач відкриває свій веб-браузер на комп'ютері чи смартфоні, він опосередковано використовує HTTP [31]. HTTP це прикладний протокол, який працює поверх набору протоколів TCP/IP, який є основою інтернету, для більшої наочності, відобразимо це на рис. 6.12.

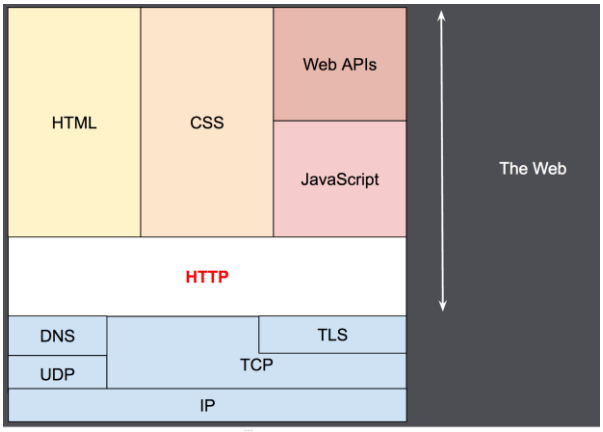


Рисунок 6.12 – Місце HTTP в інтернеті

За допомогою протоколу HTTP відбувається обмін ресурсами між клієнтськими пристроями та серверами через інтернет. Клієнтські пристрої надсилають серверам запити на ресурси, як показано на рис. 6.13, необхідні для завантаження веб-сторінки, а сервери в свою чергу надсилають відповідь клієнту для виконання запитів. Запити та відповіді спільно використовують під документи, такі як дані про зображення, текст, макети тексту тощо, які об'єднуються клієнтським веб-браузером для відображення повного файлу веб-сторінки.

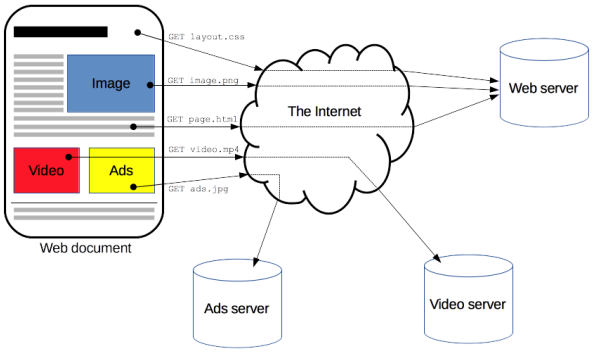


Рисунок 6.13 – Структурна схема доступу до матеріалів

На цьому пункті, базове налаштування NAS можна вважати завершеним. Інші налаштування доступу, повинен виконати саме користувач, для забезпечення максимальної захищеності даних.

На рис. 6.14, зображений інтерфейс налаштованого хмарного сховища, з прикладом розміщення файлів різного формату.

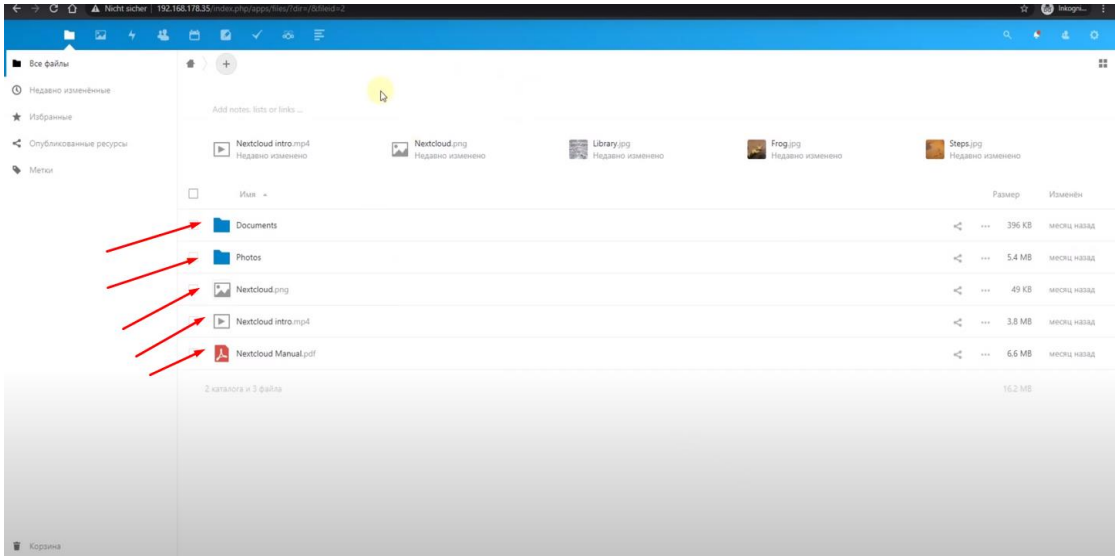


Рисунок 6.14 – Інтерфейс хмарного сховища

6.2 Розробка коду на Асемблері

Основна програма нашого проекту використовує мову програмування PHP на високому рівні. Ми також розглянемо приклад коду, написаного за допомогою найпростішої мови асемблера на нижчому рівні [32].

Код на рис. 6.15 відображає на екрані слово «Entrance».

```
start:
        mov r7, #4
        mov r0, #1
        ldr r1, =string
        mov r2, #stringlen
        swi 0

        mov r7, #1
        swi 0

        .data
string:
        .ascii "Entrance\n"
        stringlen = . - string
```

Рисунок 6.15 – Код програми для виведення тексту

Директиви асемблера та компонувальника є першими двома рядками коду, але вони не є інструкціями центрального процесора. Кожна програма повинна мати визначену «початкову» точку у своєму коді, для нас це була найперша лінія. За замовчуванням компонувальник починає виконувати код з першої зустрічної інструкції. Але нам потрібно наказати йому ігнорувати будь-які додаткові інструкції, щоб він почав із першої інструкції й не чекав.

Регістр R7 може вмістити до 16 номерів. Розробники можуть використовувати до 16 регістрів у мікросхемах архітектури ARM, ці регістри називаються R з нумерацією від 0 до 15. Номер 4 додається до реєстру R7, коли розробники дотримуються наведених нижче інструкцій.

Інструкція «mov» в цій ділянці коду, дуже схожа на однойменну інструкцію x86, звернемо увагу на символ «#» поруч із числом 4, цей символ вказує на ціле число, а не на адресу комірки пам’яті. У цьому випадку використовується системний виклик ядра Linux «write» для виведення нашого рядка. Для використання системних викликів регістри повинні бути заповнені необхідними значеннями до того, як ядро програми почне роботу. Номер системного виклику має бути розміщено в регістрі R7, число 4 є номером системного виклику «запису».

Використовуючи наступну інструкцію «mov», ми розміщуємо дескриптор файлу там, де має бути записаний рядок «Entrance». Тобто дескриптор стандартного вихідного потоку в регістрі R0. Оскільки в цьому випадку ми використовуємо стандартний вихідний потік, регістр містить стандартний дескриптор, який дорівнює 1. Далі нам потрібно ввести адресу рядка для виведення в регістр R1 за допомогою інструкції "ldr". Наприкінці коду, у розділі даних, ми оголошуємо цей рядок як послідовність символів ASCII. Для успішного використання системного виклику «write» нам також потрібно повідомити ядру операційної системи довжину вихідного рядка, тому ми вставляємо значення «stringlen» у регістр R2. Значення stringlen обчислюється шляхом віднімання адреси кінця рядка від адреси початку рядка.

На даний момент ми заповнили всі регістри необхідними даними і готові передати керування ядру Linux. Для цього ми використовуємо інструкцію "swi", назва якої розшифровується як "software interrupt", що перекладається на простір ядра операційної системи. Ядро операційної системи перевіряє вміст регістра R7, знаходить там ціле число 4 і робить висновок, що написаний код намагається

вивести рядок. Після цього він перевіряє вміст інших регістрів, виводить рядок і повертає керування нашій програмі.

В результаті виконання коду, користувач побачить на дисплеї слово «Entrance», після чого йому потрібно коректно завершити його виконання. Реалізація цього доволі проста, ми помістили в регістр R7 номер системного виклику «exit», після чого викликавши інструкцію програмного переривання під кодом 0. Ось і все, ядро операційної системи завершує виконання нашої програми і ми повертаємося до командної оболонки.

Тепер давайте розглянемо приклад більш складної програми, яка повинна відстежувати натискання клавіш користувача та відповідним чином реагувати. Приклад коду показано на рис. 6.16.

```
start:
        ldr r1, =string1
        mov r2, #string1len
        bl print_string
loop:
        mov r7, #3      @ read
        mov r0, #0      @ stdin
        ldr r1, =char
        mov r2, #2      @ два символи
        swi 0

        ldr r1, =char
        ldrb r2, [r1]
        cmp r2, #113    @ Код ASCII символа 'q'
        beq done

        ldr r1, =string2
        mov r2, #string2len
        bl print_string
        b loop

done:
        mov r7, #1
        swi 0

print_string:
        mov r7, #4
        mov r0, #1
        swi 0
        bx lr

        .data
string1:
        .ascii "Enter q to quit!\n"
        string1len = . - string1
string2:
        .ascii "Wrong button\n"
        string2len = . - string2
char:
        .word 0
```

Рисунок 6.16 – Код програми для виведення тексту

Наша програма спочатку переміщує внутрішній вказівник на початок рядка і вводить значення його довжини у відповідний регістр для подальшого виконання системного виклику «write», потім негайно переходить до підпрограми «print_string», яка наведена в коді нижче.

Для виконання переходу використовується інструкція «bl», це скорочено від «branch and link», яка сама зберігає поточну адресу в коді, що дозволяє нам

пізніше вдатися до інструкції «bx». Підпрограма «print_string» заповнює інші регістри, виконує системний виклик «write» так само, як перша ділянка коду, яку ми розглянули вище. Використання інструкцію «bx» дозволяє нам вийти з підпрограми та повернутися в головний цикл в один рядок.

Повернувшись до виконання коду, ми можемо знайти мітку під назвою «loop» – назва мітки вже означає, що ми повернемося до неї пізніше. Але спочатку ми відслідковуємо символи, які увів користувач клавіатурою, за допомогою системного виклику «read» (номер 3). Отже, ми вставляємо значення 3 у регістр R7 і значення 0 (дескриптор стандартного вхідного потоку) у регістр R0, оскільки ми читаємо користувачеві те, що ми вводимо, а не дані з файлу.

Далі ми розміщуємо адресу, де ми хочемо зберегти символ, прочитаний і поміщений ядром операційної системи в регістр R1 - у нашому випадку це область пам'яті «char», описана в кінці розділу даних.

Повертаючись до основного коду, бачимо, що значення 2 було вміщено в регістр R2, це відповідно двом символам, які ми хочемо зберегти. Потім відбувається перехід в простір ядра операційної системи для виконання операції читання. Користувач набирає текст та натискає Enter, після чого йде перевірка введених символів. Для цього поміщаємо адресу області пам'яті («char» у розділі даних) у регістр R1, потім використовується інструкцію «ldrb» для завантаження байта з області пам'яті, на яку вказує значення цього регістру.

Квадратні дужки в даному випадку вказують на те, що дані зберігаються в цікавій для нас області пам'яті, а не в самому регістрі. Таким чином, регістр R2 тепер містить єдиний символ з області пам'яті «char» з секції даних, причому це саме той символ, який ввів користувач. Наше наступне завдання буде полягати в порівнянні вмісту регістра R2 з символом «q», який є 113 символом таблиці ASCII. Тепер ми використовуємо інструкцію «cmp» для виконання операції порівняння, після чого використовуємо інструкцію «beq», ім'я якої розшифровується як "branch if equal", для переходу до мітки done в тому випадку, якщо значення з регістра R2 рівне 113. Якщо це не так, то ми виводимо наший другий рядок, після чого здійснюємо перехід до початку циклу за допомогою оператора b.

Нарешті, після мітки done ми повідомляємо ядру ОС про те, що ми хочемо завершити виконання програми, точно так як і в першій програмі.

7 ТЕХНІКО-ЕКОНОМІЧНА ЧАСТИНА

Повна собівартість системи у грошовій формі включає в себе поточні витрати за весь цикл виробництва та збут. Виробничу собівартість формують витрати, орієнтовані лише на виробництво, а повна собівартість включає в себе виробничу собівартість разом з податками, заробітною платою робітників та витратами на збут. Розрахунок собівартості системи за статтями витрат називається калькуляцією.

Витрати, пов'язані з виробництвом і збутом реалізацією системи групуються за такими статтями:

- матеріали та комплектуючі;
- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на соціальні заходи;
- витрати на утримання і експлуатацію устаткування;
- загальновиробничі витрати;
- адміністративні витрати;
- витрати на збут.

Витрати на матеріали та комплектуючі визначаються виходячи з ціни за одиницю матеріалу або комплектуючого та їх загальної кількості, наведемо в табл. 7.1 повний перелік комплектуючих, які необхідні для реалізації системи. Дані про ціни на матеріали та комплектуючі варто брати з відомостей виробників і постачальників матеріалів, сировини, комплектуючих, послуг в розрахунку на 1 одиницю випуску.

Таблиця 7.1 – Розрахунок витрат на комплектуючі

№ з/п	Найменування комплектуючих	Кількість, шт.	Ціна за од., грн.	Вартість, грн.
Модулі				
DD1	Raspberry Pi 3 B+	1	3500,00	3500,00
DD2, DD5	FT232BM	2	40,61	81,22
DD3	A-Data Premier	1	156,00	156,00

Продовження табл. 7.1.

DD4, DD6	93C46	2	4,73	9,46
Резистори				
R1, R2	С2-23-0,125-27Ом ±5%	2	2,50	5,00
R3, R9	С2-23-0,125-1,5кОм ±5%	2	0,50	1,00
R4, R10	С2-23-0,125-470Ом ±5%	2	0,60	1,20
R5, R11	С2-23-0,125-2,2кОм ±5%	2	0,50	1,00
R6, R12	С2-23-0,125-10кОм ±5%	2	0,50	1,00
R7, R8	С2-23-0,125-27Ом ±5%	2	2,50	5,00
Інше				
XP1	Ethernet-cable, 1м	1	26,34	26,34
XP2	Micro USB	1	40,67	40,67
XP3, X4	WD10EZEX	2	1557,00	3114,00
ZQ1, ZQ2	HC-49S	2	5,0	10,00
C1,C2	Y5V-100nf-25v ±20%	2	3,25	6,50
Сумарні витрати				6958,39

Крім модулів, резисторів, конденсаторів та інших радіотехнічних виробів, для побудови та збірки потрібно ще придбати основу, на якій все буде розміщено та з'єднувальні матеріали: припій, каніфоль, флюс та лак. Ціни на вищеперераховані матеріали, наведені в табл. 7.2.

Таблиця 7.2 – Приклад розрахунку витрат на сировину та матеріали

Матеріал, сировина	Одиниця виміру	Норма витрати	Ціна за одиницю, грн.	Вартість, грн.
Склотекстоліт	$м^2$	0,04	653	26,12
Каніфоль	кг	0,04	175	7,00
Флюс	кг	0,05	198	9,90
Припій	кг	0,08	312	24,96
Лак	кг	0,04	375	15,00
Сумарні витрати				82,98

З урахуванням транспортно-заготівельних витрат ($k_{т\text{-}з}$=5÷15%) вартість комплектуючих та матеріалів складе:

$$\text{КМ} = (6947{,}39 + 76{,}65) \cdot \frac{100 + 5}{100} = 7386{,}79 \text{ грн.}$$

Витрати на основну заробітну плату (Зо):

$$З_{\text{о}} = \sum_{i=1}^{n} Тг_i \cdot Нч_i \cdot n$$

Де:

- $\text{Тг}_\text{і}$ – годинна тарифна ставка окремого спеціаліста (інженера електронщика, лаборанта тощо), що задіяний у виробництві пристрою (установки), грн/год;
- $\text{Нч}_\text{і}$ – витрачений час робітником на виробництво і наладку пристрою (установки);
- n – кількість працівників, які задіяні у виробництві пристрою (установки).

Годинна тарифна ставка розраховується, виходячи з величини місячного окладу спеціаліста:

$$Тг_i = \frac{Тм_i}{Вф_i \cdot 8} = \frac{6700}{23 \cdot 8} = 36{,}41 \text{ грн.}$$

Де:

- $\text{Тм}_\text{і}$ – місячний оклад (ставка) спеціаліста, грн;
- $\text{Вф}_\text{і}$ – фактично відпрацьований час за розрахунковий період (місяць), днів.

Розрахуємо основну заробітну плату:

$$Зо = \sum_{i=1}^{n} Тг_i \cdot Нч_i \cdot n = 36{,}41 \cdot 1 \cdot 2 = 72{,}83 \text{ грн.}$$

Додаткова заробітна плата (10÷30% від $З_{\text{о}}$):

$$Зд = Зо \cdot Кд = 72{,}83 \cdot 25\% = 18{,}21 \text{ грн.}$$

Де:

– Кд – відсоток додаткової заробітної плати.

Відрахування на соціальні заходи містять відрахування від суми основної і додаткової заробітної платні за встановленими ставками:

– на державне страхування від нещасних випадків;

– на обов'язкове державне соціальне страхування на випадок безробіття;

– у зв'язку з тимчасовою втратою працездатності і витратами, зумовленими народженням дитини і похованням.

Загальний відсоток вищенаведених відрахувань дорівнює 36,3%. Розрахуємо суму:

$$Всз = (Зо + Зд) \cdot 36{,}3\% = (72{,}83 + 18{,}21) \cdot 36{,}3\% = 33{,}05 \text{ грн.}$$

Витрати на утримання та експлуатацію устаткування складають 120-150% від основної заробітної плати:

$$B_{\text{УЕУ}} = З_о \cdot 1{,}2 = 72{,}83 \cdot 120\% = 87{,}39 \text{ грн.}$$

Загально виробничі витрати визначаються із відомостей по аналізу повної собівартості виробу і в середньому можуть складати 130-250 % від основної заробітної плати.

$$B_{\text{ЗВ}} = З_о \cdot 130\% = 72{,}83 \cdot 130\% = 94{,}68 \text{ грн.}$$

Виробнича собівартість визначається як сума статей витрат:

$$C_{\text{В}} = КМ + З_{\text{о}} + З_{\text{д}} + B_{\text{СЗ}} + B_{\text{УЕУ}} + B_{\text{зв}} = 7692{,}94 \text{ грн.}$$

Адміністративні витрати визначаються із відомостей по аналізу повної собівартості виробу і в середньому можуть складати 140-200% від основної заробітної плати.

$$B_{\text{А}} = З_{\text{о}} \cdot 1{,}4 = 72{,}83 \cdot 1{,}4 = 101{,}96 \text{ грн.}$$

Зовнішні виробничі витрати, які мають зв'язок зі збутом виробів, складають 5-10% від виробничої собівартості:

$$B_{ЗВ} = C_{В} \cdot 0{,}05 = 7692{,}94 \cdot 0{,}05 = 384{,}65 \text{ грн.}$$

Повна собівартість:

$$ПС = C_{В} + B_{A} + B_{ЗВ} = 7692{,}94 + 101{,}96 + 384{,}65 = 8179{,}54 \text{ грн.}$$

Таблиця 7.3 – Калькуляція собівартості пристрою

Стаття калькуляції	Витрати, грн.
Матеріали та комплектуючі	7386,79
Витрати на основну заробітну плату	72,83
Додаткова заробітна плата	18,21
Відрахування на соціальні заходи	33,05
Витрати на утримання і експлуатацію устаткування	87,39
Загальновиробничі витрати	94,68
Виробнича собівартість	**7692,94**
Адміністративні витрати	101,96
Витрати на збут	384,65
Повна собівартість пристрою	**8179,54**

Прибуток визначається виходячи з нормативу (показника) рентабельності виробництва продукції, який встановлює підприємство

$$R = \frac{П}{C} \cdot 100\%$$

Відповідно оптова ціна пристрою визначається:

$$Ц_{опт} = C + \frac{R \cdot C}{100} = 8179{,}54 + \frac{8179{,}54 \cdot 30}{100} = 10633{,}4 \text{ грн.}$$

Визначення відпускної ціни пристрою. Відпускна ціна включає податок на додану вартість в розмірі 20%:

$$Ц_{розд.} = Ц_{опт.} + 20\% = 10633{,}4 + 20\% = 12760{,}08 \text{ грн.}$$

ВИСНОВКИ

У кваліфікаційній роботі магістра була розроблена електронна система мультисервісного доступу до корпоративного хмарного сховища даних на основі одноплатного комп'ютера Raspberry Pi 3 B+. Систем зберігає дані користувачів да забезпечує доступ до них через локальну або глобальну мережу на основі облікових записів. Основні функції, які забезпечує розроблена система:

- авторизація користувача за допомогою логіна та пароля;
- захищений доступ до даних;
- прийом користувацьких даних в поширених форматах;
- зберігання користувацьких даних на локальному сховищі;
- резервне копіювання за технологією Raid 1;
- редагування користувацьких даних на локальному сховищі;
- видалення користувацьких даних з локального сховища;
- контроль вільного простору в сховищі;
- контроль придатності сховища.

Пристрій відповідає технічному завданню.

У першому розділі проведений огляд можливих схем побудови хмарних сховищ, їх переваги та недоліки та проведений розгляд декількох конкурентних пристроїв. Також були розглянуті тарифи найпоширеніших постачальників хмарних сервісів.

В другому розділі міститься огляд на роботи в сфері захисту інформації, в області безпеки передачі даних через канал зв'язку SSH. Розглянутий принцип роботи SSH протоколу, на основі якого були виділені частини, які потенційно можуть бути вразливими. Проаналізовано методи протидії втрати конфіденційності.

Третій розділ містить розробку алгоритму роботи системи мультисервісного доступу до корпоративного хмарного сховища. На базі схеми алгоритму побудована структурна схема цієї ж системи. Четвертий розділ в свою чергу доповнює структурну схему, на базі чого проводиться розробка функціональної схеми системи.

У фінальних розділах роботи, наведений відбір елементної бази та розробка програмного забезпечення.

СПИСОК ЛІТЕРАТУРИ

1. Кузнєцов А. Тест: яке хмарне сховище найшвидше. https://www.iguides.ru/main/other/test_kakoe_oblachnoe_khranilishche_samoe_bystroe/.
2. https://www.microsoft.com/uk-ua/microsoft-365/onedrive/compare-onedrive-plans?activetab=tab:primaryr1
3. https://one.google.com/plans.
4. https://mega.io/pro
5. My Cloud™ Home Duo Personal Cloud Storage - Product Overview
6. Synology_DS220_Plus_Data_Sheet.
7. М. Райтман., 2016. Як знайти та завантажити в Інтернеті будь-які файли. https://computer.howstuffworks.com/router.htm
8. http://www.dom-spravka.info/_mobilla/ra_mod.html
9. Kaiser, M., 2020. SSH-MITM e Ssh Audits Made Simple.
10. Ylonen, T., Lonvick, C., 2006. The secure shell (SSH) transport layer protocol. RFC 4253. RFC Editor. URL: http://www.rfc-editor.org/rfc/rfc4253.txt http://www. rfc-editor.org/rfc/rfc4253.txt.
11. Provos, N., Friedl, M., Honeyman, P., 2003. Preventing privilege escalation. In: USENIX Security Symposium.
12. Pfoh, J., Schneider, C., Eckert, C., 2009. A formal model for virtual machine introspection. In: Proceedings of the 1st ACM Workshop on Virtual Machine Security. ACM, pp. 1e10.
13. Garfinkel, T., Rosenblum, M., 2003. A virtual machine introspection based architecture for intrusion detection. In: Network and Distributed Systems Security Symposium (NDSS), pp. 191e206.
14. Sentanoe, S., Taubmann, B., Reiser, H.P., 2018. Sarracenia: enhancing the performance and stealthiness of SSH honeypots using virtual machine introspection. In: Nordic Conference on Secure IT Systems. Springer, pp. 255e271.
15. Kittel, T., 2014. Libdwarfparser. https://github.com/kittel/libdwarfparser/ [accessed 2021-06-06].
16. Zhang, C., Yang, J., Yan, D., Yang, S., Chen, Y., 2013. Automated breakpoint generation for debugging. J. Softw. 8, 603e616.
17. Щепетов А.Г., Основи проектування приладів та систем; М.l Академія, 2016.

18. Погуляй О.Р. Пристрій реєстрації аналогових сигналів / Бережна О.В., Щокотова І.В., Романенко Є.С., Гагіна О.М., Погуляй О.Р. // Фізика, електроніка, електротехніка (ФЕЕ-2021). Матеріали та програма науково-технічної конференції. – Суми: СумДУ, 2021. - С.90.
19. https://elinux.org/RPi_SD_cards
20. Погуляй О.Р. Особливості раманівських спектрів полікристалічних плівок Cd1-xZnxTe, отриманих методом вакуумного термічного випаровування в квазізамкненому об'ємі. / Д'яченко О.В., Борисенко, Іващенко М.М., Опанасюк А.С. // Фізика, електроніка, електротехніка (ФЕЕ-2019). Матеріали та програма науково-технічної конференції. – Суми: СумДУ, 2019. - С.°90.
21. Hasso Plattner, Alexander. Zeier. In-Memory Data Management: Technology and Applications. — SpringerLink : Bücher. — Springer, 2012. — С. 45. — 267 с.
22. https://datasheets.raspberrypi.com/power-supply/usb-c-power-supply-product-brief.pdf
23. Погуляй О.Р. Оптимізація конструкції оптично-прозорих сонячних елементів на основі гетеропереходу n-Zn1-xMgxO / p-Cu2O. / Д'яченко О.В., Іващенко М.М., Опанасюк А.С. // Фізика, електроніка, електротехніка (ФЕЕ-2020). Матеріали та програма науково-технічної конференції. – Суми: СумДУ, 2020. - С.°90.
24. Погуляй О.Р. Коди Фібоначчі в системах обробки інформації. / Борисенко О.А., Бережна О.В., Романенко Є.С., Гагіна О.М., Погуляй О.Р. // Фізика, електроніка, електротехніка (ФЕЕ-2021). Матеріали та програма науково-технічної конференції. – Суми: СумДУ, 2021. - С.90.
25. Орлов П. А. Порівняльний аналіз ефективності використання сучасних хмарних сховищ // Молодий вчений. - 2017.
26. https://ownyourbits.com/nextcloudpi/s
27. https://rufus.ie/ru/
28. Глібін Є.С., Пряділов А.В., Програмування електронних пристроїв; Тольятті: Вид-во ТГУ, 2014.
29. https://levelup.gitconnected.com/what-is-ssh-103f89e3e4b8
30. https://seopressor.com/blog/http-vs-https/
31. http://rus-linux.net/MyLDP/algol/asm/asmschool_arm_assembly_language.html

ДОДАТОК 1

СЕКЦІЯ 5: Електронні системи, прилади і засоби кодування інформації ФЕЕ :: 2019

Оптимізація конструкції оптично-прозорих сонячних елементів на основі гетеропереходу *n*-$Zn_{1-x}Mg_xO$ / *p*-Cu_2O

Погуляй О.Р., *студент*; Д'яченко О.В., *науковий співробітник*; Іващенко М.М., *старший викладач*; Опанасюк А.С., *професор*
Сумський державний університет, м. Суми, Україна

Геліоенергетика є одним з найбільш ефективних та екологічно безпечних методів одержання електричної енергії. Як правило, для того щоб збирати та перетворювати сонячне випромінювання в електрику, потрібний значний простір (дахи будинків чи прибудинкова територія) для встановлення сонячних елементів (СЕ). Ці вимоги є суттєвою перешкодою для практичного використання таких приладів. Вказаний недолік привів дослідників до необхідності створення прозорих фотоелектричних перетворювачів, які можуть накладатися на віконне скло без відчутного зниження коефіцієнта пропускання світла, перетворюючи його в СЕ.

Перспективним поглинальним матеріалом для створення СЕ третього покоління є оксид міді Cu_2O, оскільки він є прозорим у видимій області, а використання плівок твердого розчину $Zn_{1-x}Mg_xO$ як віконного шару дає змогу оптимізувати електричні параметри цього приладу.

Для визначення конструктивних та технологічних особливостей СЕ на основі ГП *n*-$Zn_{1-x}Mg_xO$ / *p*-Cu_2O з використанням програми SCAPS, було проведено моделювання фізичних процесів у цих приладах.

В результаті моделювання визначені оптимальні товщини поглинального, струмознімального, віконного шарів та концентрація магнію у твердому розчині, які можуть забезпечити максимальну ефективність прозорих СЕ. Ці значення дорівнюють: $d_{Cu2O} = 5$ мкм, $d_{ZnMgO} = 25$ нм, $d_{ZnO:Al} = 25$ нм, $x(\mathrm{Mg}) = (0{,}25\text{-}0{,}30)$. Розраховано основні робочі характеристики приладів з конструкцією ZnO:Al / $Zn_{1-x}Mg_xO$ / Cu_2O, а саме: $U_{xx} = 1{,}48$ В, $J_{кз} = 16{,}11$ мА/см^2, $FF = 81{,}69$ %, $\eta = 9{,}63$ %.

Встановлено, що СЕ із поглинальним шаром Cu_2O, не зважаючи на те що мають відносно низкі значення ККД, через високу прозорість у видимій області спектру (87,38%) можуть бути використані як накладки на автомобільне, віконне скло, та екрани смартфонів, планшетів, ноутбуків чи інших гаджетів.

ДОДАТОК 2

ФЕЕ :: 2020

СЕКЦІЯ 5: Електронні системи, прилади і засоби кодування інформації

Особливості раманівських спектрів полікристалічних плівок $Cd_{1-x}Zn_xTe$, отриманих методом вакуумного термічного випаровування в квазізамкненому об'ємі

Знаменщиков Я.В., *старший викладач*;
Погуляй О.Р., *студент гр. ЕС-71*;
Колесник М.М., *доцент*; Опанасюк А.С., *професор*
Сумський державний університет, м. Суми, Україна

У роботі вивчені особливості раманівських спектрів товстих полікристалічних плівок твердого розчину $Cd_{1-x}Zn_xTe$ (CZT), отриманих шляхом випаровування суміші шихти CdTe та ZnTe у квазізамкненому об'ємі при зміні складу у широкому діапазоні значень від x = 0,06 до x = 0,68. Додатково з метою встановлення кристалічної якості зразків проводилось вивчення їх морфології поверхні та структурних характеристик.

Структурні дослідження показали, що плівки нанесені при T_e = 973 К, T_s = 673 К були однофазними та містили кубічну фазу CZT, а фазовий склад при зміні концентрації цинку не змінювався. З мікроструктурних досліджень було встановлено, що залежність кристалічної якості плівок CZT від концентрації цинку має параболічний характер.

Для досягнення резонансного підсилення інтенсивності мод фононних коливань вимірювання раманівських спектрів проводилося при резонансних умовах, коли енергія кванта збуджуючого лазерного випромінювання є дещо більшою, ніж ширина забороненої зони досліджуваного матеріалу. Раманівські спектри мали характерну для твердих розчинів CZT двомодову поведінку і включали моди поздовжніх (LO) та поперечних (TO) коливань CdTe та ZnTe.

За результатами раманівських досліджень для полікристалічних плівок CZT в широкому діапазоні зміни x було вивчено залежність положення мод фононних коливань від концентрації цинку. Отримана залежність в загальному випадку корелює з теорією оптичних фононних коливань для CZT, проте виявлені деякі відмінності в положенні мод в порівнянні з монокристалами, що може бути спричинено впливом мікронапружень кристалічної гратки в полікристалічних плівках CZT.

ДОДАТОК 3

Коди Фібоначчі в системах обробки інформації

Борисенко О.А., *професор*, Бережна О.В., *доцент*,
Романенко Є.С., *студентка*, Гагіна О.М., *студентка*,
Погуляй О.Р., *студент*
Сумський державний університет, м. Суми, Україна

Розвиток інформаційного суспільства залежить від ефективності використання «розумних» систем на базі реалізації розподілених обчислювань. Ефективне вирішення цієї актуальної задачі залежить від алгоритмів, які забезпечують високу надійність та швидкість при обчислюванні та передачі даних, захист обладнання від завад та збоїв.

Використання в якості завадостійких кодів, наприклад, потужних циклічних кодів забезпечує досить високу достовірність інформації в телекомунікаційних системах. Але ці роздільні коди забезпечують завадостійку передачу інформації, а не її обробку. По-друге, такі завадостійкі коди не виконують контроль правильності роботи джерела інформації, обчислювальної системи або цифрового автомата.

У роботі проведено обґрунтування використання завадостійких нероздільних кодів Фібоначчі для здійснення наскрізного контролю інформації в завадостійких системах обробки та передачі інформації. Важливою властивістю коду є те, що його кодові комбінації являють собою числа відповідної системи числення, над якими можна виконувати арифметико-логічні операції в обчислювальному пристрої.

Крім того, код Фібоначчі характеризується простотою алгоритмів виявлення та виправлення помилок. Ознакою помилки в кодової комбінації є наявність двох, трьох та більше одиниць поспіль. Поява трьох одиниць, що стоять поруч, дозволяє виправляти одиночну помилку шляхом інвертування одиниці, що розташована всередині. Ця особливість дозволяє здійснювати самоконтроль працездатності цифрового автомату Фібоначчі при обробці даних, усувати збої в його роботі без додаткового обладнання та часу, виправляти помилки без організації перезапитів під час інформаційного обміну даними.

Дослідження показали, що нероздільні коди є універсальними, які дозволяють одночасно контролювати збір, обробку, формування сигналів керування та передачу інформації. Код Фібоначчі є простим у технічній реалізації та може ефективно використовуватися для обробки даних «автоматом Фібоначчі», їх шифруванні та подальшої передачі.

ДОДАТОК 4

СЕКЦІЯ 5: Електронні системи, прилади і засоби кодування інформації ФЕЕ :: 2021

Пристрій реєстрації аналогових сигналів

Бережна О.В., *доцент*; Щокотова І.В., *зав. лабораторією*;
Васильєв В.Р., *студент*; Погуляй О.Р., *студент*
Сумський державний університет, м. Суми, Україна

Сучасна тенденція розвитку розподілених інтелектуальних мереж потребує впровадження різноманітних пристроїв вимірювання, контролю та керування, які використовують для інтеграції до мереж послідовні інформаційні канали зв'язку. Для впровадження та ефективної підтримки каналів необхідна наявність пристроїв моніторингу інформаційного обміну та працездатності мереж.

Одним із ефективних засобів моніторингу якості інформаційного обміну є комунікаційні аналізатори реального часу, які дозволяють досліджувати обмін по послідовним портам шляхом приєднання до фізичних послідовних портів серверів та робочих станцій з метою здійснення аналізу та документування обміну даними. Недоліком таких засобів є відсутність можливості аналізу рівнів аналогових сигналів у інформаційних шинах та пошук причин відмов при передачі даних.

Дослідження показали, що для вирішення такої задачі доцільно створення такого програмно-апаратного пристрою, який би поєднував в себе властивості цифрового комунікаційного аналізатору параметрів обміну даними в реальному часі та аналогового осцилографу. Підключення в розрив каналу такого «сніфера» дозволяє «прослуховувати» канали передачі даних, аналізувати рівні та форму модульованих сигналів, здійснювати їх демодуляцію, конвертувати в строки символів формата ASCII та аналіз на логічному рівні протоколів обміну між сервером та кінцевими пристроями.

Аналіз алгоритмів показав, що в якості апаратної платформи для побудови такого «сніфера» достатньо обрати модуль Arduino, який містить в собі мікроконтролер серії ATmega 328P, розробити програму, яка передбачає підключення бібліотек для використання інтерфейсу SPI та графічної бібліотеки екрану (U8glib.h), ініціалізацію змінних, констант та портів на вхід/вихід.

Перспективою розвитку функціональності запропонованого пристрою може бути контроль каналів передачі від несанкціонованого втручання, шифрування та дешифрування даних з метою зменшення витоків інформації.

ДОДАТОК 5

ФЕЕ :: 2022 *СЕКЦІЯ 5: Електронні системи, прилади і засоби кодування інформації*

Вплив складу твердого розчину $Cu_2ZnSn(S_xSe_{1-x})_4$ на оптичні втрати та ККД тонкоплівкових сонячних елементів

Кахерський С.І., *аспірант;* Погуляй О.Р., *студент;* Опанасюк А.С., *професор*

Сумський державний університет, м. Суми, Україна

Як поглинальні шари сонячних елементів (СЕ) третього покоління у наш час запропоновані такі матеріали як Cu_2ZnSnS_4, $Cu_2ZnSnSe_4$, $Cu_2ZnSn(S_xSe_{1-x})_4$. Великою перевагою цих сполук є відсутність у складі екологічно небезпечних і рідкісних компонентів, навпаки, елементи, що входять до їх складу, широко поширені у земній корі, а вартість їх видобутку відносно невисока. Окрім цього змінюючи концентрацію сірки та селену у п'ятикомпонентній сполуці $Cu_2ZnSn(S_xSe_{1-x})_4$ можна змінювати ширину забороненої зони E_g цього матеріалу точно підлаштовуючи його до максимуму ККД СЕ.

Метою даного дослідження було визначення, шляхом моделювання, оптичних втрат на границях розділу і в об'ємі допоміжних шарів СЕ в залежності від складу поглинального шару $Cu_2ZnSn(S_xSe_{1-x})_4$ та товщини струмознімального і віконного шарів у приладі зі структурою *n*-ITO(ZnO)/*n*-CdS/*p*-$Cu_2ZnSn(S_xSe_{1-x})_4$/тильний контакт та оптимізація його конструкції; визначення максимального ККД таких приладів з врахуванням оптичних втрат енергії.

Встановлено, що з врахуванням оптичних втрат енергії найоптимальнішу конструкцією з розглянутих має СЕ з конструкцією скло/ZnO/CdS/$Cu_2ZnSnSe_4$ при товщині струмознімального та віконного шарів d_{ZnO} = 100 нм; d_{CdS} = 25 нм. Врахування втрат світла на поглинання в допоміжних шарах приладів навіть при їх найменшій технологічній товщині, зменшує коефіцієнт проходження світла до шару $Cu_2ZnSnSe_4$ на (4,86–4,95)%, а до шару Cu_2ZnSnS_4 на (4,81–5,57) %. Проведені розрахунки свідчать, що шар ZnO у всіх випадках є дещо більш привабливим для створення фронтального контакту приладу ніж шар ITO. З врахуванням оптичних втрат та літературних даних визначено максимальний ККД СЕ в умовах освітлення AM1,5G. Показано, що в умовах освітлення AM1,5G максимальну ефективність мають СЕ на основі твердого розчину складу $Cu_2ZnSn(S_{0,67}Se_{0,33})_4$, як без врахування оптичних втрат (η = 33,83%) так і з їх врахуванням ($\eta_{пр}$ = (29,18-29,21)%).