

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

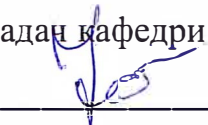
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект програмного рішення для емуляції ОС OpenWrt»

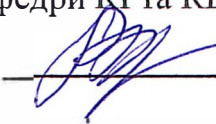
Виконав студент 4-го курсу групи КІ-19-1
спеціальності 123 «Комп'ютерна інженерія»

 Іваненко В.В

Керівник старший викладач кафедри КІ та КБ

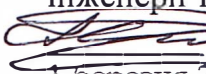
 Фальковський І. Г

Рецензент професор кафедри КІ та КБ, д.т.н., доцент

 Воротніков В. В.

Житомир – 2023

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
 Андрій Єфіменко
1 березня 2023 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Іваненко Володимир Вадимович

Тема роботи: «Проект програмного рішення для емуляції ОС OpenWrt»

затверджена Наказом університету від 1 березня 2023 р. № ____

Термін здачі студентом закінченої роботи 27 червня 2023 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

1. Система емуляції OpenWrt
2. Технологія Qemu

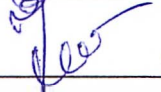
Обладнання та програмне забезпечення:

1. ПЗ Qemu
2. Операційна Debian Linux
3. Мова програмування Python
4. ПЗ Balena Etcher
5. Atheros AR9271 USB-пристрій

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

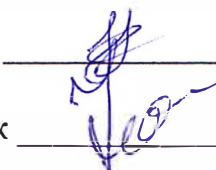
Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Фальковський І. Г	01.03.2023	01.03.2023
2	Фальковський І. Г	01.03.2023	01.03.2023
3	Фальковський І. Г	01.03.2023	01.03.2023

Студент  Іваненко В.В

Керівник  Фальковський І. Г

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	1.04.2023	Виконано
2	Пошук інформації стосовно емуляції та віртуалізації	2.04.2023 - 5.04.2023	Виконано
3	Аналіз інструментів для емуляції	6.04.2023	Виконано
4	Порівняння засобів емуляції та віртуалізації	7.04.2023 -13.04.2023	Виконано
5	Ознайомлення з особливостями архітектур вбудованих програм та операційних систем	7.04.2023 -13.04.2023	Виконано
6	Ознайомлення з технологією Qemu	14.04.2023-20.04.2023	Виконано
7	Вибір інструментів для реалізації програмного рішення	21.04.2023-26.04.2023	Виконано
8	Проектування програмного рішення	27.04.2023	Виконано
9	Вибір технологій для побудови мережі	28.04.2023 -1.05.2023	Виконано
10	Вибір технологій для побудови графічного інтерфейсу	2.05.2023	Виконано
11	Розробка програмного коду	3.05.2023 -16.06.2023	Виконано
12	Тестування та верифікація програмного коду	17.06.2023-18.06.2023	Виконано
13	Написання пояснювальної записки	18.06.2023-23.06.2023	Виконано
14	Отримання відгуку та рецензій	23.06.2023-26.06.2023	Виконано
15	Захист	27.06.2023	Виконано

Студент  Іваненко В.В

Керівник  Фальковський І. Г

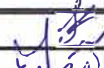
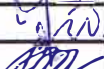
АНОТАЦІЯ

Кваліфікаційна робота складається з реалізації програмного рішення для емуляції ОС OpenWrt та пояснювальної записки до нього. Пояснювальна записка містить 73 сторінки, 67 ілюстрацій та 2 додатки.

Метою роботи є розробка програмного рішення для емуляції ОС OpenWrt і створення середовища для тестування мережевих пристроїв і налаштування мережевих сценаріїв

В роботі проведена емуляція ОС OpenWrt, застосування модифікованих образів на пристроях з різними архітектурами, створення та налаштування бездротової точки доступу з використанням USB Wi-Fi пристрою. Описані технології що використовуються для реалізації програмного рішення.

Ключові слова: ЕМУЛЯЦІЯ, ВІРТУАЛІЗАЦІЯ, ГПЕРВІЗОР, ВІРТУАЛЬНА МАШИНА, BRIDGE, QEMU, OPENWRT, ОБРАЗ, ЯДРО

					ККІК.КР.Б – 123 – 6 – ПЗ			
	Арк.	№ докум.	Підпис	Дата				
Зроб.	Іваненко В.В			20.06.23	Проект програмного рішення для емуляції ОС OpenWrt	Літ.	Арк.	Аркушів
Віник	Фальковський І. Г			20.06.23			4	73
Візн.	Воротніков В.В			20.06.23				
Вітр.								
Віверд.	Єфіменко А.А.			20.06.23	ФІКТ, Гр. КІ-19-1			

ANNOTATION

The qualification work consists of the implementation of a software solution for emulating the OpenWrt OS and an explanatory note to it. Explanatory note contains 73 pages, 67 illustrations and 2 appendices.

The purpose of the work is to develop a software solution for emulating the OpenWrt OS and create an environment for testing network devices and setting up network scenarios

The work includes emulation of the OpenWrt OS, application of modified images on devices with different architectures, creation and configuration of a wireless access point using a USB Wi-Fi device. The technologies used to implement the software solution are described.

Keywords: EMULATION, VIRTUALISATION, HYPERVISOR, VIRTUAL MACHINE, BRIDGE, QEMU, OPENWRT, IMAGE, KERNEL

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП	8
РОЗДІЛ 1 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ	9
1.1 Огляд операційної системи OpenWrt	9
1.2 Поняття і принципи емуляції та віртуалізації	10
1.3 Архітектурні аспекти, пов'язані з емуляцією та віртуалізацією	14
1.4 Аналіз наявних програмних рішень та визначення переваг і недоліків.....	17
Висновки до розділу 1	28
РОЗДІЛ 2: ВИБІР ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ	28
2.1 Визначення технічних вимог до програмного забезпечення.....	28
2.2 Вибір мови програмування	29
2.3 Вибір засобу для емуляції	30
2.4 Вибір засобу для побудови віртуальної мережі.....	30
Висновки до розділу 2	31
РОЗДІЛ 3: РОЗРОБКА СИСТЕМИ ЕМУЛЯЦІЇ	31
3.1 Проектування основних функцій	31
3.2 Реалізація функціональних модулів та їх взаємодія.....	31
3.3 Опис інтерфейсів користувача.....	33
Висновки до розділу 3	37
РОЗДІЛ 4: ТЕСТУВАННЯ ТА ВЕРИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	37
4.1 Підготовка тестових сценаріїв та наборів даних	37
4.2 Проведення тестування	38
Висновки до розділу 4	60
ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	62
ДОДАТКИ.....	65
Додаток А.....	66
Додаток Б	73

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС - Операційна система

GPL - General Public License

ВМ - Віртуальна машина

RDP - Remote Desktop Protocol

USB - Universal Serial Bus

VPN - Virtual private network

NAT - Network Address Translation

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Проєкт програмного рішення для емуляції ОС OpenWrt є актуальною розробкою, оскільки OpenWrt це популярна операційна система для маршрутизаторів і мережевих пристроїв, яка забезпечує широкі можливості налаштування та розширення. Розробка програмного рішення, що емулює OpenWrt, дозволить тестувати, валідувати та впроваджувати налаштування мережевих пристроїв без фізичної наявності апаратного обладнання.

Метою кваліфікаційної роботи є розробка програмного рішення для емуляції операційної системи OpenWrt та створення середовища для тестування мережевих пристроїв та налаштування мережевих сценаріїв

Об'єкт дослідження стосується операційної системи OpenWrt, а також можливостей, які вона пропонує для налаштування та розширення мережевих пристроїв.

Предметом дослідження є емуляції OpenWrt, створення середовища для емуляції та впровадження функціональних можливостей OpenWrt для тестування та налаштування мережевих пристроїв.

У ході кваліфікаційної роботи було розроблено проєкт програмного рішення, яке дозволяє тестувати та налаштувати мережеві пристрої у зручному середовищі та дозволяє емулювати операційну систему OpenWrt на віртуальних пристроях.

		Іваненко В.В		
		Фальковський І. Г		
Змн.	Арк.	№ докум.	Підпис	Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

8

РОЗДІЛ 1 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Огляд операційної системи OpenWrt

OpenWrt є відкритою операційною системою для маршрутизаторів, мережевих комутаторів та іншого мережевого обладнання. Базується на ядрі Linux і надає широкі можливості для управління мережевими пристроями та їх налаштування. Основні можливості OpenWrt включають функції маршрутизації, які дозволяють перенаправляти трафік мережі між різними мережами. Брандмауери які можна налаштувати, щоб захистити мережу від несанкціонованого доступу та атак. Підтримує бездротові стандарти, такі як Wi-Fi, і пропонує можливості конфігурації точок доступу, інтерфейси мережі та бездротову безпеку. Гнучке налаштування VPN і шифрування дозволяє налаштувати віртуальні приватні мережі для безпечного підключення до віддалених мереж і підтримує кілька протоколів шифрування. Програмне забезпечення можна додавати, видаляти та оновлювати безпосередньо на мережевому пристрої за допомогою вбудованої системи керування пакетами. OpenWrt дозволяє моніторити мережу, отримувати статистику трафіку, змінювати сервіси та керувати мережевими пристроями за допомогою вебінтерфейс або CLI.

Переваги використання полягають у широкому спектрі пакетів програмного забезпечення, доступних завдяки великій спільноті користувачів і розробників, що дозволяє розширювати функціональність мережевих пристроїв та відкритому використанні, оскільки OpenWrt доступна під ліцензією GPL.

До недоліків можна віднести складність використання, оскільки система призначена для технічно обізнаних користувачів. Для некваліфікованих користувачів або тих, хто не має досвіду, вона може бути дещо складним у використанні.

Іваненко В.В

Фальковський І. Г

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

9

1.2 Поняття і принципи емуляції та віртуалізації

Віртуалізація та емуляція є важливими технологіями для розгортання програмного забезпечення та ефективного використання ресурсів у сучасному світі. Віртуалізація створює окремі віртуальні середовища для роботи операційних систем і додатків, тоді як емуляція дозволяє імітувати поведінку та характеристики інших систем. Розробка, тестування, виконання та забезпечення безпеки — це лише деякі з можливих застосування цих технологій. У даному розділі будуть розглянуті та визначені основні терміни, пов'язані з емуляцією, віртуалізацією та будуть висвітлені архітектурні аспекти.

Розуміння цих термінів є ключовим для розробки програмного забезпечення

Перш за все розглянемо поняття емуляції:

Емуляція - це процес створення програмного середовища, що імітує характеристики та поведінку іншої системи або пристрою. Вона дозволяє виконувати програмне забезпечення, створене для однієї системи, на іншій системі, яка може мати іншу архітектуру або операційну систему.

Створюється "емульована система", яка може бути аналогічна до оригінальної системи або пристрою за своїми функціональними можливостями, взаємодією та поведінкою. При цьому програмне забезпечення, що здійснює емуляцію, імітує роботу апаратної частини, операційної системи та інших складових емульованої системи.

Один з ключових аспектів - це відтворення інструкційної архітектури оригінальної системи. Емулятор аналізує інструкції, що виконуються на оригінальній системі, та перетворює їх в еквівалентні інструкції для емульованої системи. Це дозволяє програмам, написаним для оригінальної системи, працювати на емульованій системі безпосередньо, незалежно від різниці в архітектурі.

Процес емуляції може включати такі етапи:

- 1) Розбір та інтерпретація інструкцій оригінальної системи. Емулятор аналізує інструкції програми, що виконується на оригінальній системі, та розуміє їх значення та функціонал.
- 2) Трансляція та перетворення оригінальної системи на відповідні інструкції для емульованої системи. Цей процес відбувається у реальному часі або перед запуском програми на емульованій системі.
- 3) Виконання трансльованих інструкцій на емульованій системі, імітуючи роботу оригінальної системи.
- 4) Обробка введення/виведення, взаємодія з периферійними пристроями та імітує їх роботу.

Емуляція використовується для різних цілей, таких як запуск програм на різних платформах, тестування та налагодження програмного забезпечення, збереження сумісності зі старими системами або пристроями, а також для ретро геймінгу, коли старі ігри відтворюються на сучасних комп'ютерах або консолях.

Наступним розглянемо поняття гіпервізор:

Гіпервізор, також відомий як віртуалізаційний монітор, є програмним забезпеченням або апаратним пристроєм, що дозволяє створювати та управляти віртуальними машинами. Його основна функція полягає в розділенні ресурсів фізичної машини між цими віртуальними машинами та контролі їх виконання.

Існують два типи гіпервізорів:

1. Тип 1 (native, bare-metal) гіпервізор: Цей тип гіпервізора встановлюється безпосередньо на фізичний сервер і працює без операційної системи хоста. Він має прямий доступ до ресурсів сервера та управляє віртуальними

Іваненко В.В

Фальковський І. Г

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

11

машинами безпосередньо. Приклади типу 1 гіпервізорів включають VMware ESXi, Citrix XenServer, Microsoft Hyper-V.

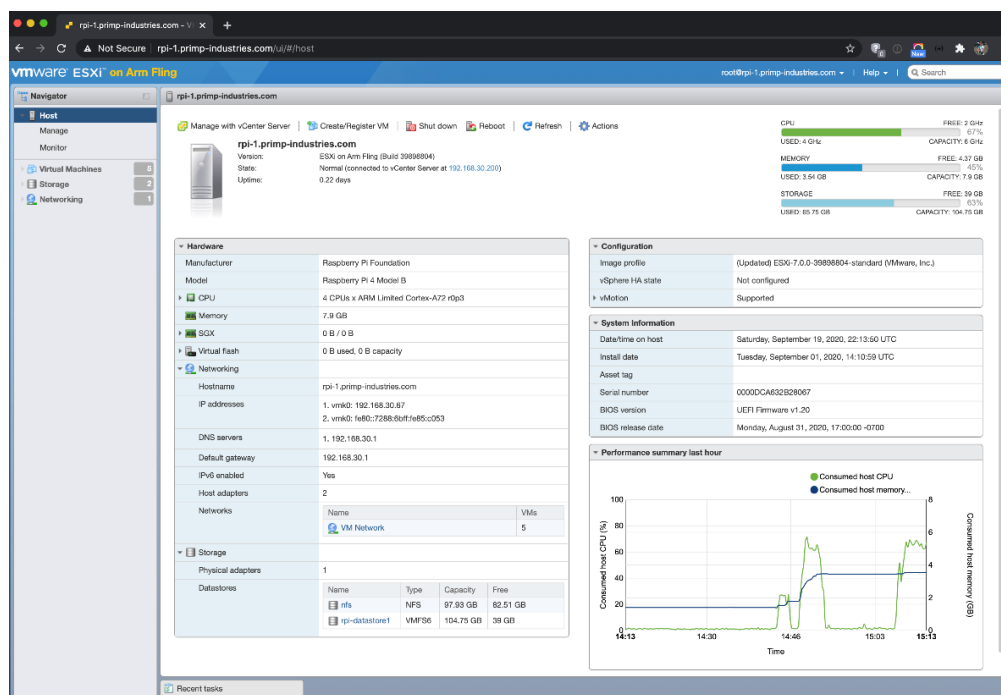


Рис 1.1 – Графічний інтерфейс VMware ESXi

2. Тип 2 (hosted) гіпервізор: Цей тип гіпервізора працює як програма в операційній системі хоста. Він встановлюється на загально використовувану операційну систему і створює віртуальні машини на її базі. Тип 2 гіпервізори залежать від операційної системи хоста для доступу до апаратного забезпечення сервера. Наприклад, Oracle VirtualBox, VMware Workstation, Parallels Desktop є прикладами типу 2 гіпервізорів.

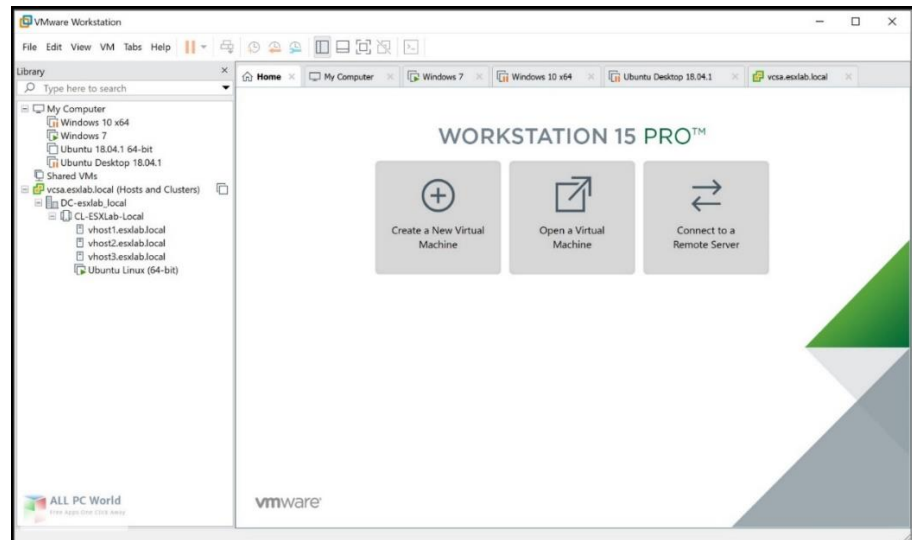


Рис 1.2 - Графічний інтерфейс VMware Workstation

Гіпервізори забезпечують віртуалізацію різних рівнів ресурсів, включаючи процесор, пам'ять, сховище та мережу. Вони розділяють фізичні ресурси між віртуальними машинами та контролюють доступ до них, щоб забезпечити ізоляцію та ефективне використання ресурсів.

Основні функції гіпервізора включають:

Створення та управління віртуальними машинами: дозволяє створювати нові віртуальні машини, налаштовувати їх параметри та управляти їх життєвим циклом, такими як запуск, зупинка та призупинення.

Розділення ресурсів: розподіляє фізичні ресурси, такі як процесорний час, оперативна пам'ять, дисковий простір та мережеві ресурси, між віртуальними машинами. Це дозволяє кожній віртуальній машині працювати в ізольованому середовищі з гарантованими ресурсами.

Моніторинг та керування: забезпечує моніторинг ресурсів та виконання віртуальних машин. Він може автоматично масштабувати віртуальні машини, розподіляти навантаження та забезпечувати високу доступність системи.

Міграція віртуальних машин: дозволяє переміщувати віртуальні машини з одного фізичного сервера на інший без перерви в роботі. Це називається живою міграцією і дозволяє забезпечити неперервну доступність та балансування навантаження.

Гіпервізори використовуються для багатьох цілей, таких як віртуалізація серверів, віртуалізація на робочому місці, тестування та розробка програмного забезпечення, віртуалізація на рівні операційної системи та багато іншого.

Віртуальна машина є програмним забезпеченням, що імітує фізичну машину, дозволяючи виконувати на ній операційну систему та додаткові програми. Створює ізольоване віртуальне середовище, яке забезпечує розділення ресурсів та виконання додатків у віртуальній системі. ВМ може мати встановлену гостьову операційну систему, яка працює від імені користувача або додатку. Гостьова операційна система вважається "віртуальною", оскільки вона працює у віртуальному середовищі. Гіпервізор надає віртуальній машині віртуальний процесор, який імітує роботу фізичного процесора. Це дозволяє гостьовій операційній системі та програмам працювати так, ніби вони виконуються на окремому фізичному сервері. Кожна віртуальна машина має свою віртуальну пам'ять, яка використовується гостьовою операційною системою та програмами для збереження даних та виконання операцій. Гіпервізор керує доступом до фізичної пам'яті та забезпечує ізоляцію між різними віртуальними машинами. ВМ може мати віртуальні пристрої, такі як віртуальні диски, мережеві інтерфейси, звукові пристрої і т. д. Ці віртуальні пристрої імітують фізичні пристрої і дозволяють гостьовій операційній системі взаємодіяти з реальними ресурсами фізичного сервера. Гіпервізор розподіляє ресурси фізичного сервера між різними віртуальними машинами, забезпечуючи гарантований доступ до процесорного часу, пам'яті, дискового простору та інших ресурсів.

1.3 Архітектурні аспекти, пов'язані з емуляцією та віртуалізацією

Розуміння архітектурних аспектів емуляції та віртуалізації, зокрема віртуальної пам'яті, віртуальних пристроїв та управління ресурсами, допомагає розробникам створювати ефективні та надійні рішення для емуляції та

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				14
Змн.	Арк.	№ докум.	Підпис	Дата		

забезпечувати належне управління ресурсами. Розглянемо кожен з цих аспектів детальніше:

Віртуальна пам'ять - це концепція, що використовується в операційних системах та системах віртуалізації для створення враження, ніби кожній програмі або процесу доступний окремий простір пам'яті, незалежний від фізичної пам'яті комп'ютера. Основна мета віртуальної пам'яті - це забезпечити можливість запуску більшого обсягу програм, ніж є доступний фізичної пам'яті. Замість того, щоб завжди тримати всю програму або процес у фізичній пам'яті, віртуальна пам'ять дозволяє завантажувати та вивантажувати частини програми або дані з диска за потребою. Кожна програма отримує віртуальний адресний простір, який складається з адрес, які використовуються самою програмою. Ці віртуальні адреси потім перетворюються в фізичні адреси за допомогою механізму перекладу адрес (так званий "переклад сторінок"), що забезпечує відповідність між віртуальними адресами та фізичними адресами в пам'яті. Коли програма спробує отримати доступ до віртуальної адреси, операційна система перевірить, чи відповідна сторінка з цією адресою присутня у фізичній пам'яті. Якщо сторінка вже знаходиться в пам'яті, то операційна система дозволяє програмі отримати доступ до фізичного місцезнаходження цієї сторінки. Якщо ж сторінка відсутня в пам'яті, відбувається процес вивантаження неактивних сторінок на диск і завантаження потрібних сторінок з диска до пам'яті. Віртуальна пам'ять дозволяє забезпечити ефективне використання обмежених ресурсів фізичної пам'яті, тому, що можна завантажувати лише необхідні сторінки програми або даних, а решту тримати на диску. Крім того, віртуальна пам'ять дозволяє розділяти пам'ять між різними процесами та забезпечує механізми захисту пам'яті, щоб запобігти несанкціонованому доступу до даних.

Віртуальні пристрої - це абстракція апаратного пристрою або пристроїв, яка створюється в рамках віртуалізаційних технологій. Вони дозволяють програмам та операційним системам взаємодіяти з апаратними пристроями, необхідними для їхньої роботи, відокремлюючи їх від конкретних фізичних ресурсів. Віртуальні

Іваненко В.В

Фальковський І. Г

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

15

пристрої створюються гіпервізором або віртуальною машиною, які надають інтерфейс для взаємодії з апаратними пристроями. Ці пристрої можуть бути повністю віртуалізованими, що означає, що вони повністю моделюють функціональність конкретного апаратного пристрою, або можуть бути частково віртуалізованими, де вони надають обмежений набір функцій або інтерфейсів. Одним із загальних прикладів віртуальних пристроїв є віртуальна мережева карта. Гіпервізор або віртуальна машина може створити віртуальну мережеву карту, яка імітує функціональність фізичної мережевої карти. Програми або операційні системи, що працюють на віртуальній машині, можуть використовувати цю віртуальну мережеву карту для здійснення мережевої взаємодії, незалежно від реального апарату. Віртуальні пристрої дозволяють розширювати функціональність, додавати нові пристрої або надавати інші можливості, незалежно від конкретного апаратного обладнання. Вони також сприяють забезпеченню ізоляції між різними віртуальними середовищами, дозволяючи їм використовувати власні віртуальні пристрої, незалежно від інших віртуальних машин або фізичних пристроїв.

Управління ресурсами в контексті емуляції та віртуалізації належить до процесу керування доступом та розподілу фізичних або віртуальних ресурсів між віртуальними машинами або контейнерами, що функціонують на одному фізичному хості. Це включає у себе керування процесорними потоками, пам'яттю, мережевими ресурсами, сховищем даних та іншими системними ресурсами.

Один з головних аспектів управління ресурсами - це розділення доступу до фізичних ресурсів між віртуальними машинами або контейнерами. Гіпервізор або віртуальна машина виступає як посередник і контролер, який надає віртуальним середовищам доступ до ресурсів згідно з обмеженнями та політиками, встановленими адміністратором системи.

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Процес управління ресурсами містить:

- 1) Планування процесорного часу, віртуальна машина або гіпервізор відповідає за розподіл процесорного часу між віртуальними машинами або контейнерами. Алгоритми планування використовуються для ефективного розподілу обчислювальних ресурсів і задоволення потреб кожної віртуальної машини чи контейнера.
- 2) Управління пам'яттю, доступ і розподіл фізичної пам'яті між віртуальними машинами або контейнерами контролюються гіпервізором або віртуальною машиною. Віртуальна пам'ять і розріджена алокація пам'яті — це методи, які допомагають ефективно використовувати обмежені ресурси пам'яті.
- 3) Керування мережевими ресурсами та розподіл пропускної здатності мережі між віртуальними машинами або контейнерами. Встановлення віртуальних мереж і віртуальних комутаторів, розробка правил мережевого фільтрування та управління мережевою пропускною здатністю можуть бути частиною цього процесу.
- 4) Управління сховищем даних, такого як фізичні або віртуальні диски, контролюється гіпервізором або віртуальною машиною. Створення віртуальних дисків, встановлення точки доступу до даних і забезпечення безпеки та ізоляції віртуальних машин або контейнерів — усе це відноситься до цього.

Управління ресурсами в емуляції та віртуалізації є важливим аспектом для забезпечення ефективного використання ресурсів та ізоляції середовищ. Вона дозволяє керувати та контролювати доступ до фізичних ресурсів, забезпечуючи кожному віртуальному середовищу відокремлену та безпечну роботу.

1.4 Аналіз наявних програмних рішень та визначення переваг і недоліків.

Наразі існує кілька рішень для емуляції мережевих пристроїв, які дозволяють тестувати, навчати та розробляти програмне забезпечення, що взаємодіє з

мережевими пристроями без потреби в фізичному обладнанні. Деякі з таких рішень включають наступне:

1) Емулятори мережевих пристроїв, що надаються виробниками:

Деякі виробники мережевих пристроїв надають власні емулятори, які дозволяють створювати віртуальні пристрої, що функціонують так само як фізичні. Наприклад, Cisco VIRL (Virtual Internet Routing Lab).

Cisco VIRL (Virtual Internet Routing Lab) - це платформа емуляції мережевих пристроїв, розроблена компанією Cisco. Вона дозволяє створювати віртуальні екземпляри мережевих пристроїв Cisco, таких як маршрутизатори, комутатори, фаєрволи, контролери WLAN та інші. Надає можливість створення складних мережевих топологій, моделювання різних конфігурацій пристроїв і взаємодії з ними. Він підтримує такі мережеві операційні системи, як Cisco IOS, IOS-XE, IOS-XR, NX-OS та ASA (Adaptive Security Appliance). Цей емулятор мережевих пристроїв використовується для різних цілей, таких як тестування конфігурацій, налагодження мережевих протоколів, навчання та підготовки кадрів. Він також може бути інтегрований з іншими інструментами та платформами для автоматизації тестування мереж. Cisco VIRL має власний графічний інтерфейс для налаштування та керування віртуальними пристроями, а також підтримує можливість використання API для автоматизації процесів. Це дозволяє інженерам швидко створювати складні тестові сценарії та проводити віртуальні експерименти з мережевими пристроями Cisco.

Іваненко В.В

Фальковський І. Г

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

18

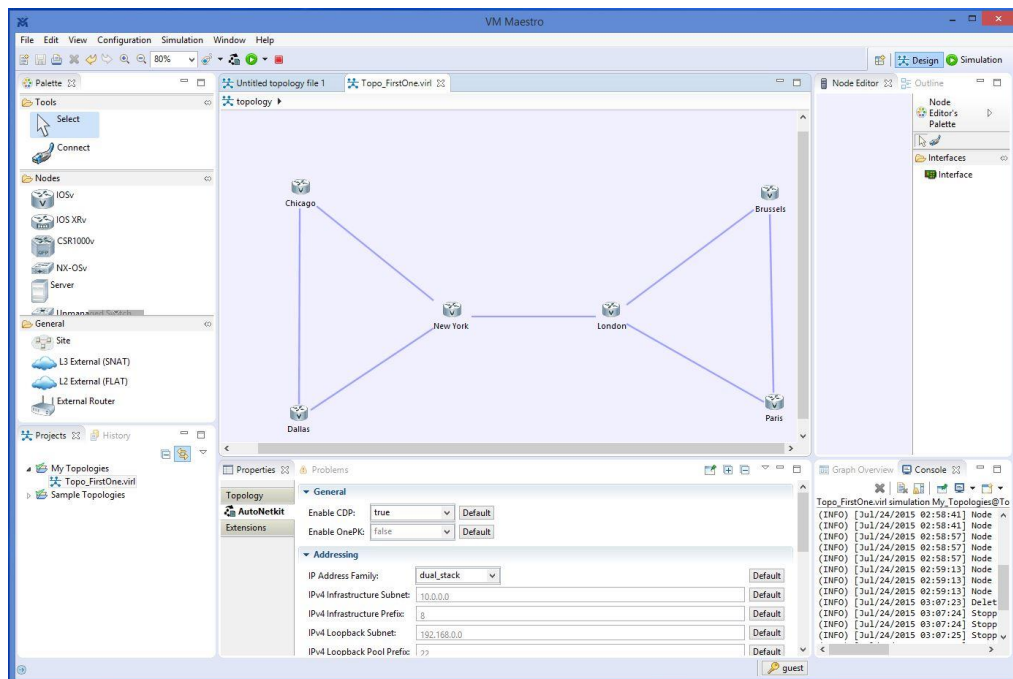


Рис 1.3 – Графічний інтерфейс Cisco VIRL

Переваги Cisco VIRL в тому що воно дозволяє створювати віртуальні екземпляри мережевих пристроїв Cisco, що функціонують так само як фізичні пристрої. Це забезпечує високий рівень реалістичності та точності емуляції. Підтримує різноманітні мережеві пристрої Cisco, включаючи маршрутизатори, комутатори, фаєрволи та контролери WLAN. Він також підтримує різні операційні системи, такі як Cisco IOS, IOS-XE, IOS-XR, NX-OS та ASA. Має графічний інтерфейс користувача, що спрощує створення, налаштування та керування віртуальними мережевими пристроями. Це робить платформу доступною для широкого кола користувачів, навіть без глибоких знань командного рядка. Підтримує використання API, що дозволяє автоматизувати процеси створення, конфігурації та керування віртуальними пристроями. Це дозволяє інтегрувати платформу з іншими інструментами та розробляти скрипти для автоматизації тестування та налагодження мереж.

Недоліком є ліцензія на використання, Cisco VIRL потребує відповідної вартості, яка може бути високою для деяких бюджетів. Емуляція мережевих пристроїв в

Cisco VIRL вимагає значних обчислювальних ресурсів і пам'яті. Для ефективної роботи рекомендується мати потужний комп'ютер або сервер з високими характеристиками. Деякі функції, доступні на фізичних пристроях, можуть бути обмежені або не підтримуватись в емульованому середовищі Cisco VIRL. Це може бути обмеженням для певних сценаріїв тестування або розробки.

2) Емулятори, що засновані на віртуалізації:

Віртуалізаційні платформи, такі як VMware або VirtualBox, можуть бути використані для створення віртуальних екземплярів мережевих пристроїв. Вони дозволяють запускати віртуальні машини з операційною системою мережевих пристроїв, таких як Cisco IOS або Juniper JunOS, і моделювати їх роботу.

Oracle VM VirtualBox є потужним і розширеним рішенням для емуляції мережевих пристроїв. Він використовується для різних цілей, включаючи розробку, тестування, відпрацювання мережних конфігурацій та навчання.

Особливості та можливості Oracle VM VirtualBox полягають у легкості створювати нові віртуальні машини та налаштовувати їх параметри, такі як розмір пам'яті, обсяг жорсткого диска, кількість процесорів тощо. VirtualBox підтримує різні операційні системи, включаючи Windows, Linux, macOS, Solaris та інші. VirtualBox має інтуїтивно зрозумілий інтерфейс, що дозволяє легко керувати віртуальними машинами. Ви можете запускати, зупиняти, призупиняти, відновлювати та переміщати віртуальні машини зручним способом. Додатково надає можливість встановлювати спеціальні гостьові додатки у віртуальні машини, що поліпшують їх функціональність та інтеграцію з хост-системою. Крім того, ви можете встановлювати розширення VirtualBox, які додають додаткові функції, такі як підтримка USB 2.0 та 3.0, віртуалізовані графічні пристрої та мережеві пристрої. Ви можете вибрати різні режими мережі, такі як мостовий режим, NAT, внутрішня мережа або сполучення декількох машин через віртуальну приватну мережу. VirtualBox дозволяє налаштовувати рівень доступу до ресурсів, таких як процесор, пам'ять та мережеві адаптери, для віртуальних машин. Ви можете призначати

пріоритети та обмеження, щоб забезпечити оптимальне використання ресурсів хост-системи. Підтримує ряд додаткових функцій, таких як віртуальні пристрої USB, підтримка віртуальних дисків VMDK та VHD, віддалений доступ до віртуальних машин через RDP та інші.

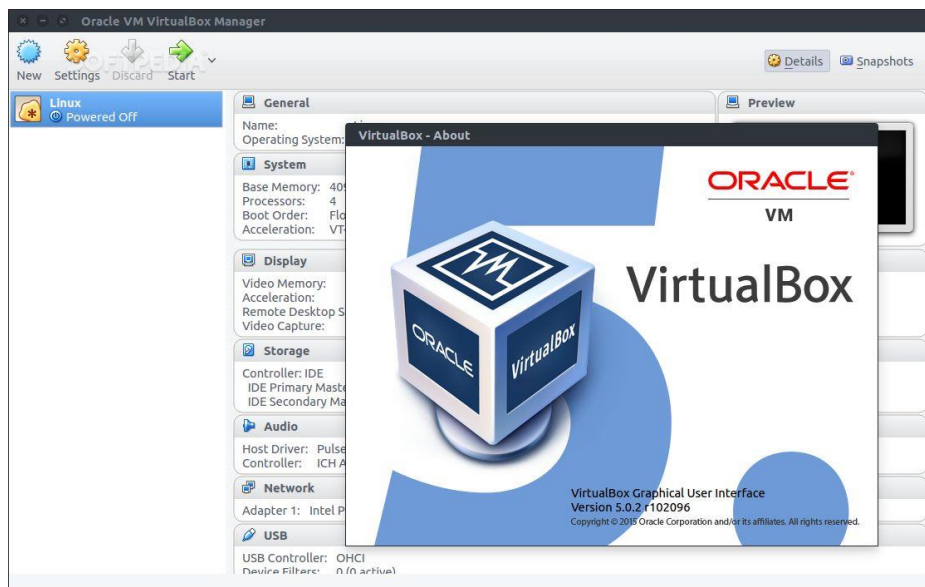


Рис 1.4 – Графічний інтерфейс VirtualBox

Недоліками VirtualBox може бути лише підтримка базових мережевих адаптерів, такі як NAT, міст або внутрішня мережа. Деякі специфічні мережеві функції або пристрої можуть бути обмежені в порівнянні з емуляторами, наданими виробниками. У порівнянні з деякими іншими віртуалізаційними платформами, VirtualBox може мати обмежену продуктивність під час виконання великої кількості віртуальних машин або ресурсомістких завдань.

VMware Workstation є одним з найпопулярніших і потужних віртуалізаційних рішень для створення та управління віртуальними машинами. Він надає багато функцій та можливостей для емуляції мережевих пристроїв.

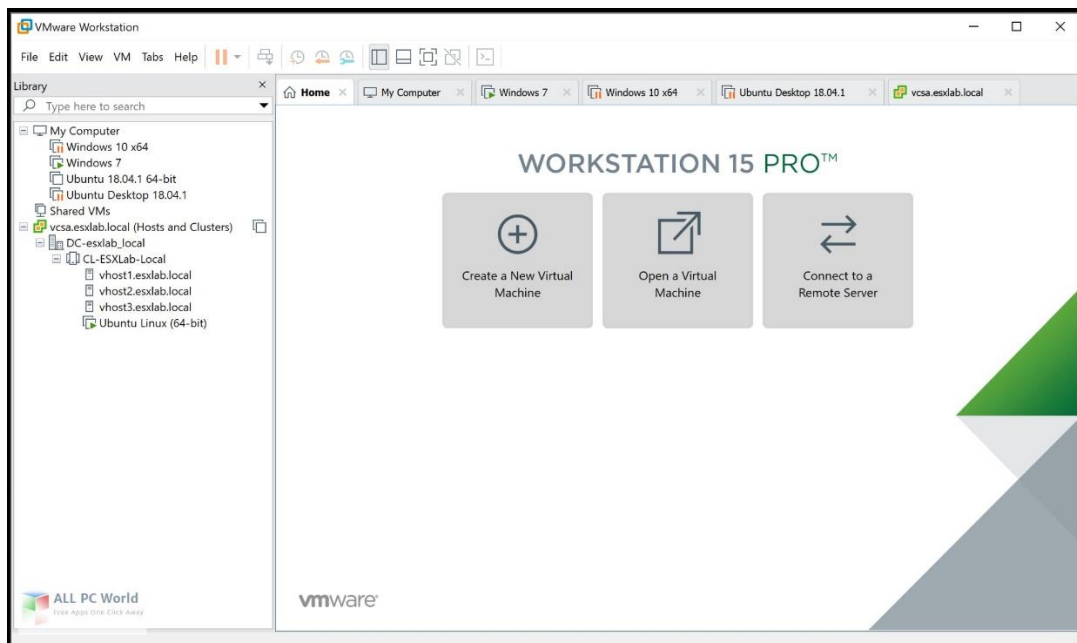


Рис 1.5 – Графічний інтерфейс VMware Workstation

Особливості і переваги VMware Workstation це гнучкість, широкі можливості для налаштування віртуальних машин та мережевих параметрів. Ви можете легко налаштовувати різні конфігурації, додавати апаратні пристрої, розгортати мережеві топології та налаштовувати мережеві з'єднання. VMware Workstation пропонує оптимізовану віртуалізацію, що дозволяє досягти високої продуктивності віртуальних машин. Він підтримує апаратне прискорення, оптимізоване використання ресурсів та ефективне керування пам'яттю. Дозволяє створювати снапшоти віртуальних машин, які зберігають стан машини на певний момент часу. Це дозволяє легко експериментувати з конфігурацією та відновлювати попередні стани в разі необхідності. Крім того, можна регулярно робити резервні копії віртуальних машин для захисту даних. Також підтримує інтеграцію з різними інструментами, такими як Docker, Vagrant, Git і багатьма іншими. Це дозволяє зручно працювати з різними технологіями та забезпечує більшу гнучкість у розробці та тестуванні.

Недоліки це вартість, бо VMware Workstation є комерційним продуктом і вимагає придбання ліцензії. Вартість може бути високою для окремих користувачів або невеликих підприємств. Робота з віртуальними машинами може вимагати значних

обчислювальних та пам'ятевих ресурсів. Для ефективної роботи з VMware Workstation рекомендується мати потужний комп'ютер з достатньою кількістю процесорів та оперативної пам'яті. Деякі складні конфігурації апаратного забезпечення можуть не підтримуватися повністю віртуалізацією, що може обмежити можливості роботи з деякими пристроями або

Hyper-V Manager є утилітою управління віртуалізаційною платформою Hyper-V, яка надає графічний інтерфейс для налаштування та керування віртуальними машинами та ресурсами Hyper-V.

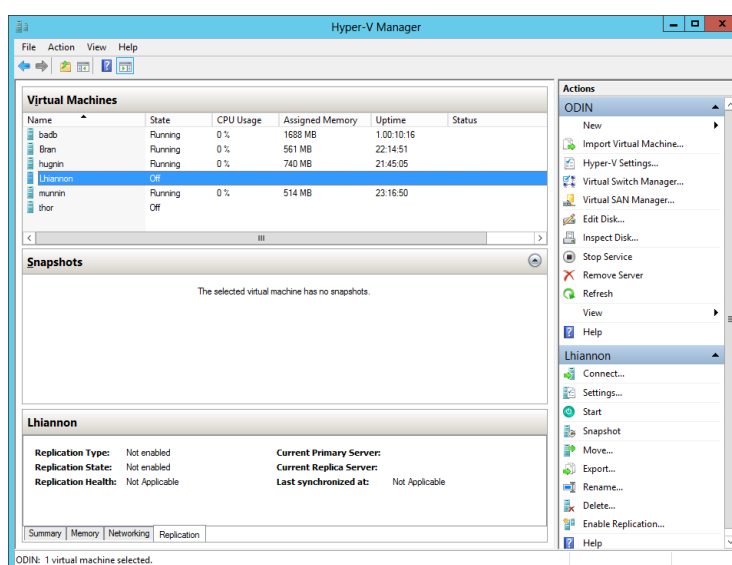


Рис 1.6 – Графічний інтерфейс Hyper-V Manager

Hyper-V Manager має глибоку інтеграцію з екосистемою Microsoft, що дозволяє зручно використовувати його в середовищах Windows Server та Windows 10. Ви можете легко інтегрувати його з іншими продуктами Microsoft, такими як Active Directory, System Center і PowerShell. Підтримує різні операційні системи, включаючи різні версії Windows та розповсюджені дистрибутиви Linux. Це дозволяє вам використовувати Hyper-V для віртуалізації різних робочих навколишніх середовищ. Hyper-V Manager дозволяє централізовано керувати різними серверами, на яких запуснені віртуальні машини Hyper-V. Ви можете здійснювати керування віртуальними машинами, змінювати їх конфігурацію, запускати та зупиняти їх, а також моніторити їх стан. Також надає інструменти для

моніторингу ресурсів та продуктивності віртуальних машин. Ви можете відстежувати використання процесора, оперативної пам'яті, дискового простору та інших ресурсів, що допомагає вам контролювати ефективність та завантаження віртуальних машин, і дозволяє налаштовувати мережеві з'єднання віртуальних машин, включаючи налаштування віртуальних комутаторів, віртуальних мереж та віртуальних адаптерів. Ви можете контролювати мережевий трафік та забезпечувати безпеку мережі віртуальних машин. Це все переваги і особливості, мінуси ж це обмежена підтримка операційних систем, підтримка віртуалізації тільки на операційних системах Windows. Це означає, що ви не зможете використовувати його для віртуалізації інших операційних систем, таких як Linux або macOS. У порівнянні з деякими іншими комерційними віртуалізаційними рішеннями, Hyper-V Manager може мати обмежені функції або меншу кількість розширених налаштувань. Деякі функції, які доступні у конкурентних продуктах, можуть бути відсутніми у Hyper-V Manager. Деякі користувачі можуть відчувати складність у налаштуванні мережевих з'єднань у Hyper-V Manager, особливо при використанні складних конфігурацій мережі, таких як віртуальні комутатори з VLAN або віртуальні прив'язки до фізичних мережевих адаптерів.

QEMU (Quick Emulator) є відкритим емулятором та віртуалізаційним інструментом, який дозволяє запускати гостьові операційні системи на хост-системі.

QEMU підтримує багато операційних систем та платформ, таких як Linux, Windows, macOS, FreeBSD та інші. Ви можете використовувати QEMU на різних системах для емуляції та віртуалізації різних гостьових операційних систем. Може емулювати різні апаратні пристрої, включаючи процесори, пам'ять, мережеві адаптери, звукові карти та інші пристрої. Це дозволяє використовувати QEMU для запуску гостьових операційних систем на хост-системі без фізичного апаратного забезпечення. QEMU підтримує різні формати образів віртуальних машин, такі як QEMU Copy-On-Write (qcow), RAW, Virtual Machine Disk (VMDK), ISO та інші. Це дозволяє зручно створювати, імпортувати та експортувати віртуальні машини в

Іваненко В.В

Фальковський І. Г

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

24

різних середовищах. Також, дозволяє налаштовувати ресурси віртуальних машин, включаючи кількість процесорних ядер, обсяг оперативної пам'яті, дисковий простір та інші параметри. Це дозволяє гнучко керувати розподілом ресурсів між різними віртуальними машинами.

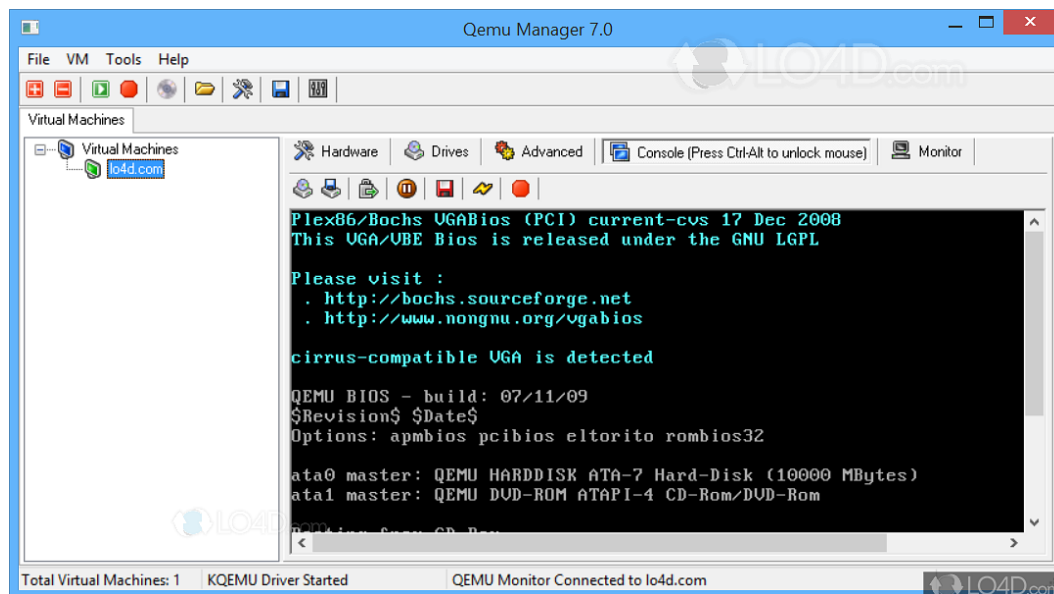


Рис 1.7 – Графічний інтерфейс Qemu Manager

Порівняно з деякими іншими віртуалізаційними рішеннями, такими як VMware або Hyper-V, QEMU може мати меншу швидкодію емуляції або віртуалізації. Це особливо помітно при запуску важких завдань або вимогливих за ресурсами гостьових операційних систем. QEMU може мати складний процес налаштування, особливо для новачків або користувачів без досвіду в емуляції або віртуалізації. Є багато параметрів та конфігураційних опцій, які можуть вимагати глибокого розуміння системи та апаратного забезпечення. Обмежена графічна підтримка може бути перешкодою, зазвичай працює у режимі командного рядка, а графічна підтримка може бути обмеженою. Порівняно з комерційними віртуалізаційними рішеннями, такими як VMware або Hyper-V, QEMU може мати меншу кількість готових шаблонів для швидкого створення віртуальних машин. Це означає, що ви, можливо, повинні налаштовувати більшість параметрів вручну.

3) Програмні емулятори мережевих пристроїв: Існують також програмні рішення, спеціалізовані на емуляції мережевих пристроїв. Один з найпопулярніших інструментів у цій сфері - GNS3 (Graphical Network Simulator), який дозволяє створювати складні мережеві топології, емулювати мережеві пристрої різних виробників та взаємодіяти з ними.

GNS3 (Graphical Network Simulator 3) є популярним емулятором мережевих пристроїв, який дозволяє створювати та моделювати віртуальні мережі. Основні переваги та можливості GNS3 включають у себе розширену підтримку мережевих пристроїв, включаючи маршрутизатори, комутатори, фаєрволи та інші пристрої від різних виробників, таких як Cisco, Juniper, MikroTik та інші. Це дозволяє створювати складні віртуальні мережі, що відтворюють реальні мережеві середовища. Можливість бути інтегрованим з фізичними мережевими пристроями, що дозволяє використовувати реальні мережеві пристрої разом з віртуальними для створення гібридних мережевих топологій. Це дає можливість тестувати та розробляти конфігурації на реальних пристроях без необхідності фізичного з'єднання з ними. GNS3 підтримує різні гостьові операційні системи, включаючи Cisco IOS, Juniper JunOS, Linux, Windows та інші. Це дозволяє запускати віртуальні машини з цими операційними системами та імітувати їх роботу в мережевій топології.

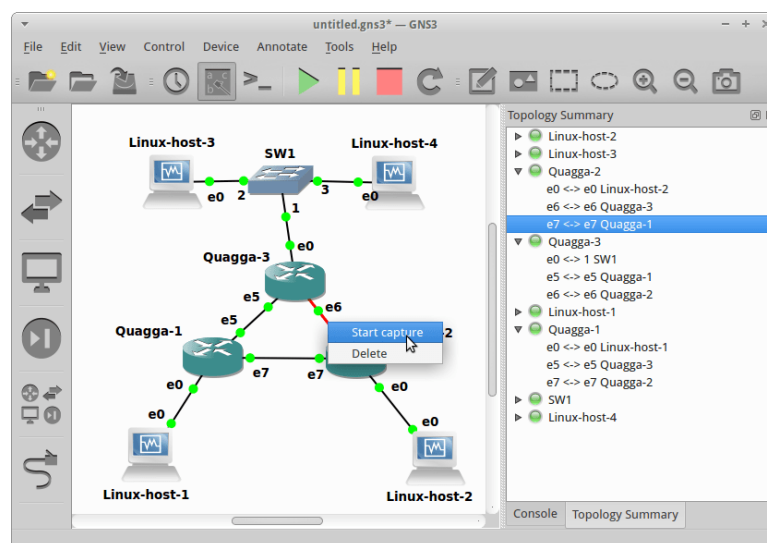


Рис 1.8 – Графічний інтерфейс GNS3

Деякі недоліки GNS3 містять в собі високі вимоги до обладнання, необхідність потужного комп'ютера з високою кількістю ресурсів, особливо при емуляції великих мереж. Великі топології можуть вимагати значних обсягів оперативної пам'яті та потужного процесора. GNS3 може мати певну складність налаштування, особливо для новачків. Налаштування взаємодії з мережевими пристроями, налаштування віртуальних машин та встановлення необхідного програмного забезпечення можуть вимагати деякого технічного розуміння. Залежність від образів ОС та ліцензій образів операційних систем, такі як Cisco IOS або Juniper JunOS, для запуску віртуальних машин. Однак, отримання цих образів може бути обмеженим через ліцензійні обмеження або доступність.

EVE-NG (Emulated Virtual Environment for Next Generation) є потужною платформою для емуляції мережевих пристроїв, яка дозволяє створювати та тестувати віртуальні мережі. Плюсами платформи є підтримка широкого спектра мережевих пристроїв, включаючи маршрутизатори, комутатори, фаєрволи та віртуальні машини, від різних виробників, таких як Cisco, Juniper, Fortinet, Palo Alto Networks та інші. Це дозволяє створювати складні мережеві топології, що відтворюють реальні середовища. Простий у використанні інтерфейс, що спрощує створення, конфігурування та керування віртуальними мережами. Ви можете легко створювати віртуальні машини, налаштовувати їх параметри та підключати до мережевих пристроїв за допомогою графічного інтерфейсу. Масштабованість та продуктивність, EVE-NG може бути розгорнутий на потужних серверах або хмарних платформах, що дозволяє створювати великі та складні мережі без втрати продуктивності. Він також підтримує використання розподілених архітектур, що дозволяє розподіляти навантаження між різними вузлами та оптимізувати продуктивність. Можливість бути інтегрований з реальними мережевими пристроями, дозволяючи підключати фізичні пристрої до віртуальних мереж та тестувати конфігурації в реальному часі. Це робить його корисним інструментом для валідації та тестування мережевих рішень перед їх реалізацією у виробничому середовищі.

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				27
Змн.	Арк.	№ докум.	Підпис	Дата		

До мінусів можна віднести вимоги до потужного сервера або використання хмарних платформ, що може бути витратним або обмеженим для окремих користувачів. Інсталяція та налаштування EVE-NG може бути складним для некваліфікованих користувачів, особливо при розгортанні на власних серверах або хмарних платформах. Деякі функції та можливості EVE-NG можуть бути обмеженими в безплатній версії, і для отримання повноцінного функціонала може бути необхідна комерційна ліцензія.

Ці рішення дозволяють інженерам мережі тестувати конфігурації, виконувати симуляції навантаження, навчати мережеві протоколи та розробляти програмне забезпечення, не залежачи від фізичного обладнання. Вибір певного рішення залежить від потреб, бюджету та вимог до функціональності емуляції.

Висновки до розділу 1

В першому розділі було наведено огляд операційної системи OpenWrt, поняття та принципи емуляції та віртуалізації, а також архітектурні аспекти, пов'язані з цими технологіями. Також проведено аналіз наявних програмних рішень для емуляції та визначення їх переваг і недоліків.

РОЗДІЛ 2: ВИБІР ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ

2.1 Визначення технічних вимог до програмного забезпечення

Програмне забезпечення повинно мати здатність емулювати операційну систему OpenWrt. Це включає виконання інструкцій та функцій, характерних для OpenWrt, а також підтримку необхідних бібліотек та середовища для правильного функціонування OpenWrt. Рішення повинно мати можливість створювати точку доступу з підтримкою USB адаптера. Це дозволить емулювати функціональність USB адаптера в середовищі OpenWrt, що дозволить підключати та використовувати USB пристрої в емульованій системі. Можливість налаштовувати віртуальну мережу для функціонування OpenWrt. Це включає

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				28
Змн.	Арк.	№ докум.	Підпис	Дата		

створення віртуальних мережесих інтерфейсів, налаштування IP адрес, маршрутизації та інших параметрів мережі для забезпечення правильної роботи OpenWrt в віртуальному середовищі.

2.2 Вибір мови програмування

Для реалізації проекту була обрана мова програмування Python, обґрунтовується це кількома факторами. Python є однією з найпопулярніших мов програмування в сучасному програмуванні. Вона має велику спільноту розробників, що сприяє доступності багатой екосистеми бібліотек і фреймворків. Використання Python дозволяє користуватись цими ресурсами та використовувати готові рішення. Має велику кількість бібліотек, які полегшують роботу з мережами, обробкою даних, конфігурацією та іншими завданнями. Використання цих готових рішень допомагає зменшити час розробки та підвищити якість програмного забезпечення. Python підтримується на різних операційних системах, що забезпечує платформову незалежність програмного забезпечення, також вбудована підтримка для взаємодії з іншими мовами програмування є величезною перевагою. Загалом, використання мови програмування Python для реалізації програмного забезпечення з використанням QEMU дозволяє отримати переваги простоти, ефективності розробки, наявності готових рішень та можливості інтеграції з наявними компонентами. Вибір бібліотеки для реалізації інтерфейсу користувача був на користь Tkinter, тому що вона вбудована у Python. Це означає, що її встановлення та використання є простими і зручними. Немає потреби встановлювати додаткові залежності або сторонні бібліотеки. Tkinter має простий та легкий у використанні API. Його інтерфейс стиль базується на графічній бібліотеці Tk, що робить його легким для вивчення та розуміння. Багато зручних функцій, таких як робота з кнопками, мітками, полями вводу, меню та іншими елементами, реалізуються за допомогою простих методів.

2.3 Вибір засобу для емуляції

Вибір засобу для емуляції в проєкті був на користь QEMU. Провівши аналіз переваг і недоліків аналогічних засобів емуляції та віртуалізації, я виділив декілька переваг. Насамперед QEMU підтримує широкий спектр архітектур та платформ, включаючи x86, ARM, PowerPC та MIPS. Це дозволяє емулювати різні типи пристроїв та операційних систем на різних апаратних платформах, розширені можливості емуляції, включаючи підтримку апаратного прискорення за допомогою KVM для платформ, що підтримують цю технологію. Дозволяє налаштувати різні параметри апаратного забезпечення, такі як кількість процесорів, обсяг пам'яті, мережеві налаштування тощо. Це дає більшу гнучкість та контроль над емульованими системами. Також надає кілька способів підключення USB пристроїв до віртуальної машини. Один з них - це підключення пристроїв в реальному часі (USB pass-through), що дозволяє взаємодіяти з пристроями безпосередньо з вашої гостьової ОС. Це особливо корисно, коли вам потрібно використовувати спеціальні USB-пристрої або взаємодіяти з зовнішніми пристроями. Крім того, QEMU також підтримує емуляцію віртуальних USB пристроїв, які можуть бути використані для симуляції підключених USB пристроїв до вашої віртуальної машини. Це дозволяє вам тестувати та використовувати віртуальні USB пристрої без реального обладнання.

Саме ці ключові переваги були вирішальними у виборі засобу для емуляції проєкту.

2.4 Вибір засобу для побудови віртуальної мережі

Для налаштування та керування віртуальною мережею було обрано Bridge-utils, бо це надійний та потужний пакет утиліт для керування мостами в Linux, що надає широкі можливості та забезпечує підтримку стандартів. Вибір обґрунтовується декількома факторами, а саме: функціональність, стандартизація та легкість використання. Bridge-utils надає потужний набір функцій для керування мостами в Linux. Він дозволяє налаштовувати різні параметри моста та здійснювати розширені налаштування, що можуть бути необхідними в складних

мережевих конфігураціях. Використовує відкритий протокол мосту Linux (Linux bridge protocol), що базується на стандарті IEEE 802.1D. Це забезпечує сумісність з іншими мережевими пристроями та стандартними рішеннями. Bridge-utils має простий та зрозумілий інтерфейс командного рядка, що дозволяє легко налаштовувати та управляти мостами. Його команди легко інтегруються в скрипти або автоматизовані процеси.

Висновки до розділу 2

В другому розділі представлено визначення технічних вимог до програмного забезпечення, виборі мови програмування, засобу для емуляції та засобу для побудови віртуальної мережі.

РОЗДІЛ 3: РОЗРОБКА СИСТЕМИ ЕМУЛЯЦІЇ

3.1 Проєктування основних функцій

На основі вивчених аналогічних програмних рішень, я виділив основні функції для реалізації. Ці функції включають у себе вибір архітектури, налаштування опцій мережі та USB пристроїв, вибір образу та ядра, запуск та зупинку емулятора, а також отримання інформації про підключені USB-пристрої та IP-адресу мережевого інтерфейсу. Реалізація цих функцій здійснюється за допомогою графічного інтерфейсу та використання відповідних бібліотек та функцій мови програмування.

3.2 Реалізація функціональних модулів та їх взаємодія

Модуль для налаштування опцій емуляції, таких як використання графічного режиму, використання власного ядра, мережеві налаштування та USB-параметри. Він включає прапорці та поля вводу на головному вікні програми, а також функції для обробки їх значень.

Запуск та зупинка емулятора, на основі обраних опцій, викликає процес QEMU з використанням цієї команди та зберігає посилання на цей процес для подальшого використання при зупинці.

Отримання інформації про USB-пристрої реалізована шляхом виконання команди "lsusb" для отримання списку підключених USB-пристроїв. Вона розбирає вихідні дані команди та повертає список ідентифікаторів та описів USB-пристроїв.

Для вибору образу та ядра створений модуль який містить функції для відкриття діалогового вікна вибору файлу, отримання шляху до обраного образу та ядра, а також відображення обраних файлів у відповідних полях вводу на головному вікні програми.

Також реалізована функція яка отримує IP-адресу мережевого інтерфейсу з використанням команди "ifconfig". Вона аналізує вихідні дані команди та повертає IP-адресу, пов'язану з вибраним мережевим інтерфейсом.

Для роботи віртуальної мережі, створений окремий файл під назвою network.py з набором функцій, які відповідають за керування віртуальними мостами та мережевими інтерфейсами. Він надає функції для створення та знищення віртуальних мостів, а також підключення до них віртуальних мережевих інтерфейсів. Використовує бібліотеки libvirt і subprocess для взаємодії з гіпервізором.

Взаємодія між модулями забезпечується за допомогою обміну даними через спільні змінні та виклик функцій з одного модуля до іншого. Наприклад, модуль для вибору образу та ядра може передати шлях до вибраного образу модуля для запуску та зупинки емулятора, а модуль для запуску та зупинки емулятора може отримати значення параметрів емуляції з модуля налаштування опцій емуляції. Ця взаємодія між модулями дозволяє забезпечити повну функціональність програми та забезпечує користувачеві зручний та інтуїтивно зрозумілий інтерфейс для налаштування.

Повний програмний код ПЗ з коментарями наведено в Додаток А.

Блок-схеми ПЗ наведені в Додаток Б.

3.3 Опис інтерфейсів користувача

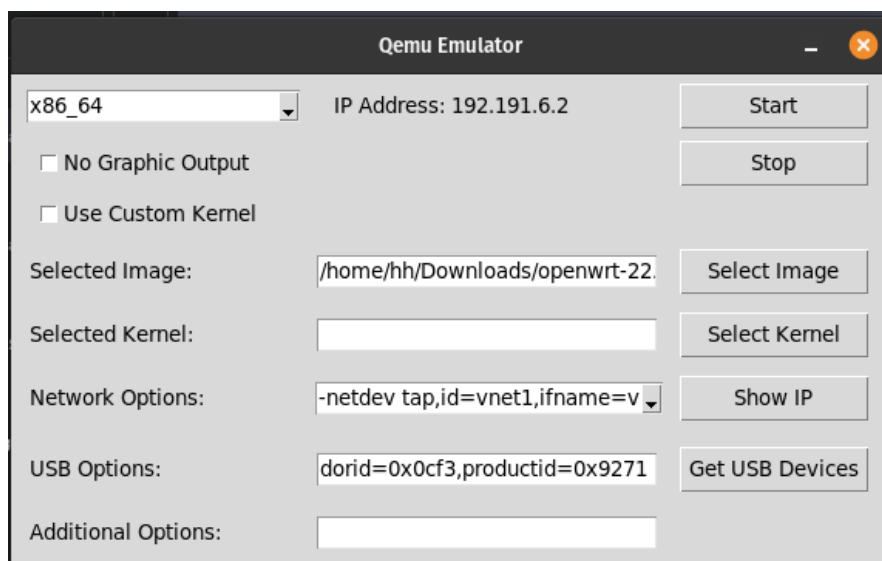


Рис 3.1 – Графічний інтерфейс основного модуля ПЗ

Користувач може обрати одну з доступних архітектур: "x86_64", "arm", "mipsel", "mipseb", або ввести власноруч. Обраний варіант зберігається у змінній `variable_arch`. Якщо прапорець No Graphic Output встановлений, графічний вивід QEMU буде вимкнений. Стан прапорця зберігається у змінній `variable_nographic`. Коли Use Custom Kernel прапорець встановлений, користувач може вибрати власне ядро. Стан прапорця зберігається у змінній `variable_use_kernel`. Створені поля для шляху до обраного образу, користувач вибирає або вводить шлях до файлу обраного образу за допомогою кнопки "Select Image". Шлях до обраного образу зберігається у змінній `variable_image_path`. Для гнучкого налаштування, користувач може ввести опції мережі в поле Network Options або вибрати з випадального списку наявних мереж, для налаштування USB пристроїв, передбачене поле USB Options та поле Additional options для додаткових опцій Qemu. Значення зберігаються у змінних `variable_network_options`, `variable_usb_options`, `variable_extra_options`. Створені кнопки управління станом, Start Emulator при натисканні на яку програма збирає всі введені налаштування та запускає емулятор QEMU з використанням цих налаштувань, та Stop Emulator, яка зупиняє роботу емулятора QEMU. Натискаючи кнопку Get USB Devices, створюється вікно модуля USB пристроїв, яке відображає інформацію про підключені USB-пристрої, коли емулятор QEMU працює. Інформація отримується

з використанням команди `lsusb` і оновлюється при кожному запиті. Щоб відображати IP адресу обраної мережі, використовується функція `show_interface_ip`, яка виконує команду `ifconfig`, та відображає результат у `ip_label`.

У підсумку, графічний інтерфейс надає змогу користувачу вибрати необхідні налаштування, запустити емулятор QEMU та отримати інформацію про підключені USB-пристрої та IP-адресу мережевого інтерфейсу.

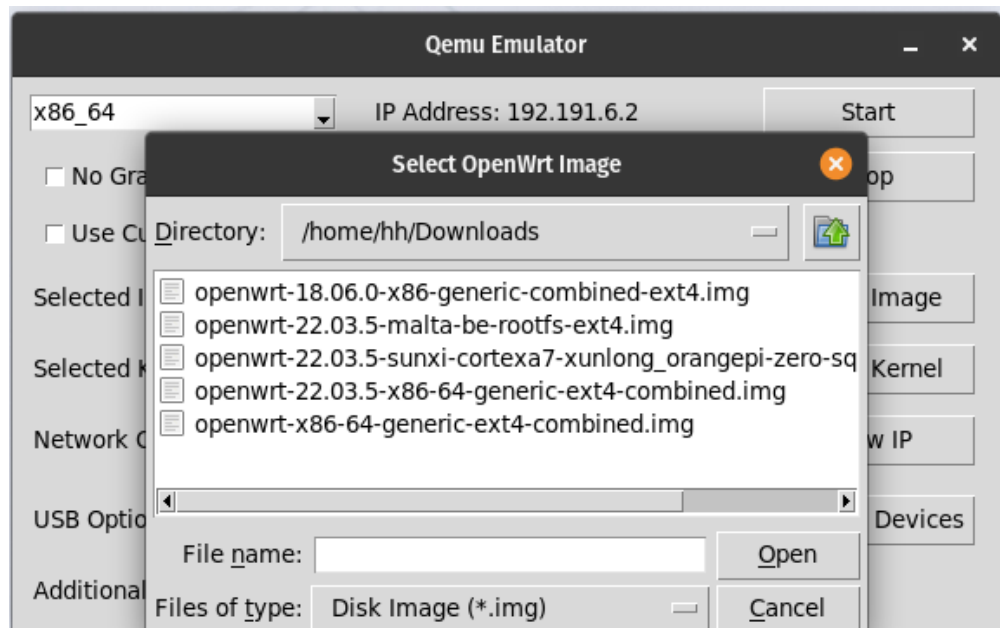


Рис 3.2 – Вікно вибору образу ОС

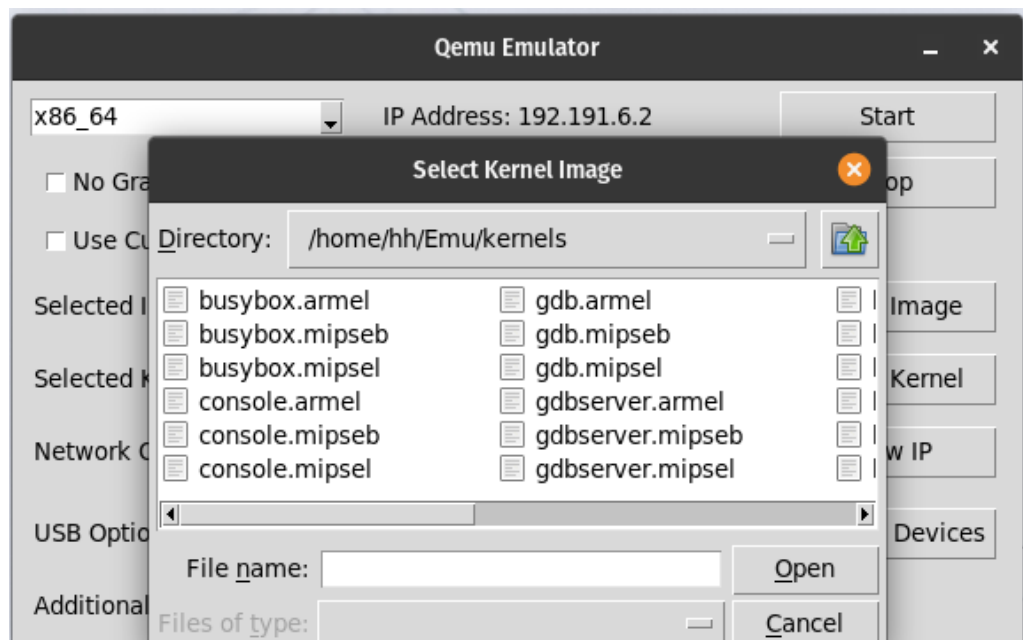


Рис 3.3 – Вікно вибору ядра

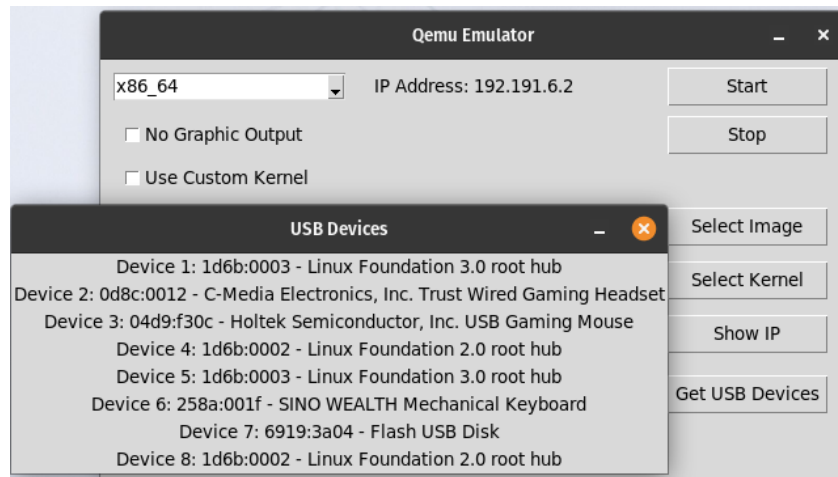


Рис 3.4 – Вікно доступних USB пристроїв

Графічний інтерфейс модуля Virtual Bridge Management забезпечує зручний спосіб створення та знищення віртуальних мостів і мережевих інтерфейсів. Після запуску програми відкривається вікно «Create Virtual Bridge». Користувачеві пропонується декілька полів введення та міток для введення необхідної інформації для створення віртуального мосту та мережі.

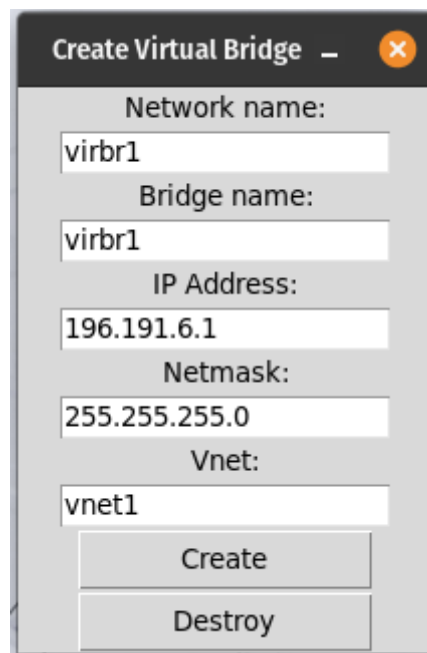


Рис 3.5 – Графічний інтерфейс network.py

Поля введення містять:

- Network name: бажана назва для віртуальної мережі.
- Bridge name: назва віртуального мосту, який буде створено.

- IP Address: IP-адреса для призначення віртуальній мережі.
- Netmask: маска мережі для налаштування мережі.
- Vnet: назва інтерфейсу віртуальної мережі.

Користувач заповнює необхідну інформацію у відповідних полях введення та натискає дві кнопки, перша це Create, натискання цієї кнопки ініціює створення віртуального мосту та мережевого інтерфейсу. Викликається функція submit, яка отримує значення, введені користувачем, і передає їх у функцію create_virbr для подальшої обробки. Якщо створення пройшло успішно, вікно повідомлення відображає повідомлення про успіх. Натиск кнопки Destroy запускає знищення віртуального мосту та мережевого інтерфейсу. Викликається функція знищення, яка отримує ім'я мосту та ім'я інтерфейсу віртуальної мережі, введене користувачем. За допомогою команд підпроцесу віртуальний міст руйнується за допомогою команди virsh net-destroy, а інтерфейс віртуальної мережі видаляється за допомогою команди ip link delete. Після натискання користувачем кнопки «Create» або «Destroy» виконується відповідна дія, і вікно закривається. Завдяки інтуїтивно зрозумілому графічному інтерфейсу користувачі можуть легко взаємодіяти з модулем і створювати або руйнувати віртуальні мости та мережеві інтерфейси відповідно до своїх вимог.

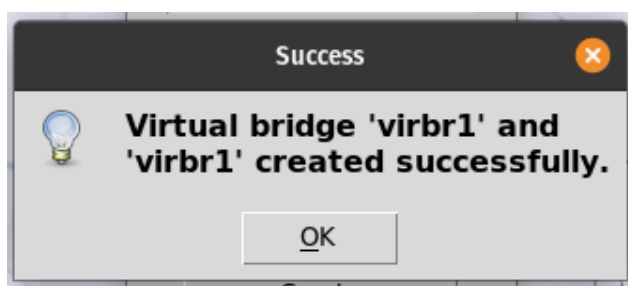


Рис 3.6 – Повідомлення про успішне створення мережі



Рис 3.7 – Повідомлення про успішне видалення мережі

		Іваненко В.В		
		Фальковський І. Г		
Змн.	Арк.	№ докум.	Підпис	Дата

Висновки до розділу 3

Розділ три присвячений проектуванню основних функцій, реалізації функціональних модулів та їх взаємодії, а також опису інтерфейсів користувача.

РОЗДІЛ 4: ТЕСТУВАННЯ ТА ВЕРИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Підготовка тестових сценаріїв та наборів даних

Підготовка тестових сценаріїв та наборів даних є важливою частиною процесу тестування, оскільки допомагає перевірити різні аспекти системи емуляції та виявити можливі проблеми або помилки. Цей етап вимагає ретельного аналізу вимог до системи та розуміння очікуваної поведінки системи в різних ситуаціях. Підготовка реалістичних тестових сценаріїв та використання репрезентативних наборів даних допоможе забезпечити надійне тестування та виявлення потенційних проблем. На основі доступних мені ресурсів, я склав такі сценарії роботи:

1) Емульована система на основі OpenWrt з архітектурою x86_64 з налаштованою точкою доступу використовуючи USB Wi-Fi адаптер Atheros AR9271.

Робота поділяється на такі кроки:

Підготовка системи:

- Встановлення віртуальної OpenWrt з архітектурою x86_64.
- Підключення USB Wi-Fi адаптера Atheros AR9271 до комп'ютера та віртуальної машини.

Конфігурація точки доступу:

- Запуск командного рядка OpenWrt.
- Встановлення необхідних пакетів та залежностей для роботи з Wi-Fi адаптером Atheros AR9271.
- Налаштування мережевих інтерфейсів

Іваненко В.В

Фальковський І. Г

Змн.

Арк.

№ докум.

Підпис

Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

37

- Конфігурація безпеки мережі, встановлення пароля та інших параметрів безпеки.

Налаштування точки доступу:

- Використання команд або налаштування файлів конфігурації для встановлення параметрів точки доступу, таких як ідентифікатор (SSID), канал, режим роботи, шифрування та інші налаштування.

Запуск точки доступу:

- Запуск Wi-Fi адаптера та активація точки доступу.
- Перевірка статусу точки доступу та підключення до неї.

Тестування та налагодження:

- Перевірка працездатності точки доступу шляхом спроби підключення та передачі даних через Wi-Fi.
- Розв'язання можливих проблем, таких як незадовільна якість зв'язку або неправильна конфігурація безпеки.

Управління та підтримка:

- Забезпечення доступу до налаштувань точки доступу через вебінтерфейс OpenWrt для подальшої зміни конфігурації або відладки.

4.2 Проведення тестування

Почнемо з вибору та завантаження образу OpenWRT. Я обрав універсальний образ останньої версії OpenWRT 22.03.5, на архітектурі x86_64.

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				38
Змн.	Арк.	№ докум.	Підпис	Дата		

Index of (root) / releases / 22.03.5 / targets / x86 / 64 /	
Image Files	
These are the image files for the x86/64 target. Check that the sha256sum of the file you download matches the one listed below. Shortened image file names below have the same prefix: <code>openwrt-22.03.5-x86-64-...</code>	
Image for your Device	sha256sum
generic-ext4-combined-efi.img.gz	412dc974d68e0d73fed9cfb6204ca9bdce2
generic-ext4-combined.img.gz	3bdb0d03b038c7cc7ccb21fa8eec0789816

Рис. 4.1 – Вибір та завантаження образу

Наступний крок це підключення Wi-Fi адаптер до хост системи, для роботи бездротової мережі. Обрав я Atheros AR9271, тому що він має широку сумісність, надійність і якість зв'язку.

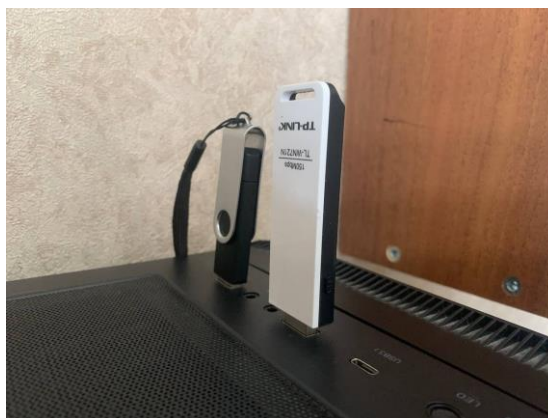


Рис. 4.2 – Підключений Atheros AR9271

Адаптер автоматично відобразився на хост системі

```
hh@pop-os:~$ lsusb | grep 9271
Bus 002 Device 004: ID 0cf3:9271 Qualcomm Atheros Communications AR9271 802.11n
hh@pop-os:~$
```

Рис. 4.3- USB Wi-Fi адаптер Atheros AR9271 у хост системі

Для підключення до емульованої ОС, у розробці реалізований наступний елемент.



Рис. 4.4 – Вікно відображення доступних USB пристроїв

У вікні відображено всі підключені USB пристрої з відомостями vendorid та productid. Саме вони потрібні нам для підключення пристрою до Qemu. Кінцева команда цього елементу виглядає так: «-device usb-ehci, id=ehci -device usb-host, bus=ehci.0,vendorid=0x0cf3,productid=0x9271».

Тепер я створюю віртуальний міст, та віртуальну мережу, використаю для цього інструмент network.py.

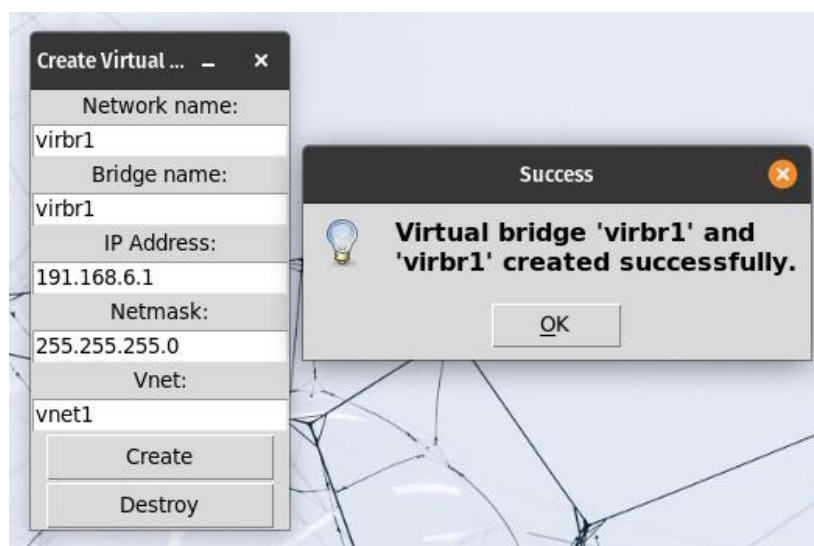


Рис. 4.5- Успішне налаштування мережі

```
virbr1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
inet 191.168.6.1 netmask 255.255.255.0 broadcast 191.168.6.255
ether 52:54:00:a9:e1:b4 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vnet1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
inet 191.168.6.2 netmask 255.255.255.0 broadcast 191.168.6.255
ether 1a:20:a1:5d:16:20 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hh@pop-os: ~/Emu$
```

Рис. 4.6 - Вміст команди ifconfig

```
hh@pop-os: ~/Emu$ brctl show
bridge name      bridge id                STP enabled  interfaces
virbr1           8000.525400a9e1b4        yes          vnet1
hh@pop-os: ~/Emu$
```

Рис. 4.7 - Вміст команди brctl show

Після успішного створення мережі, перейдемо до запуску віртуальної системи.

Виберемо образ.

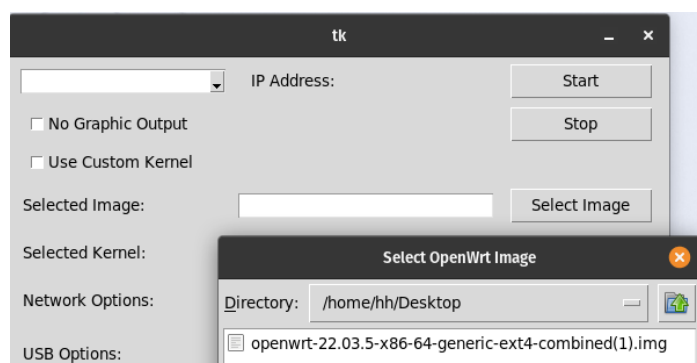


Рис. 4.8 - Вибір образу

Після образу, вкажемо потрібну архітектуру, та оберемо мережу.

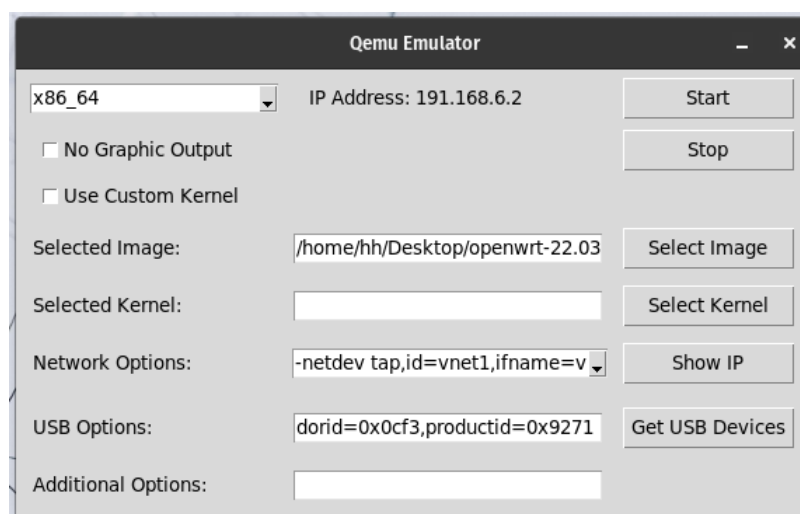


Рис. 4.9 - Налаштування перед початком роботи

Натискаємо кнопку Start, і починаємо працювати в командному рядку емульованої ОС.

		Іваненко В.В		
		Фальковський І. Г		
Змн.	Арк.	№ докум.	Підпис	Дата

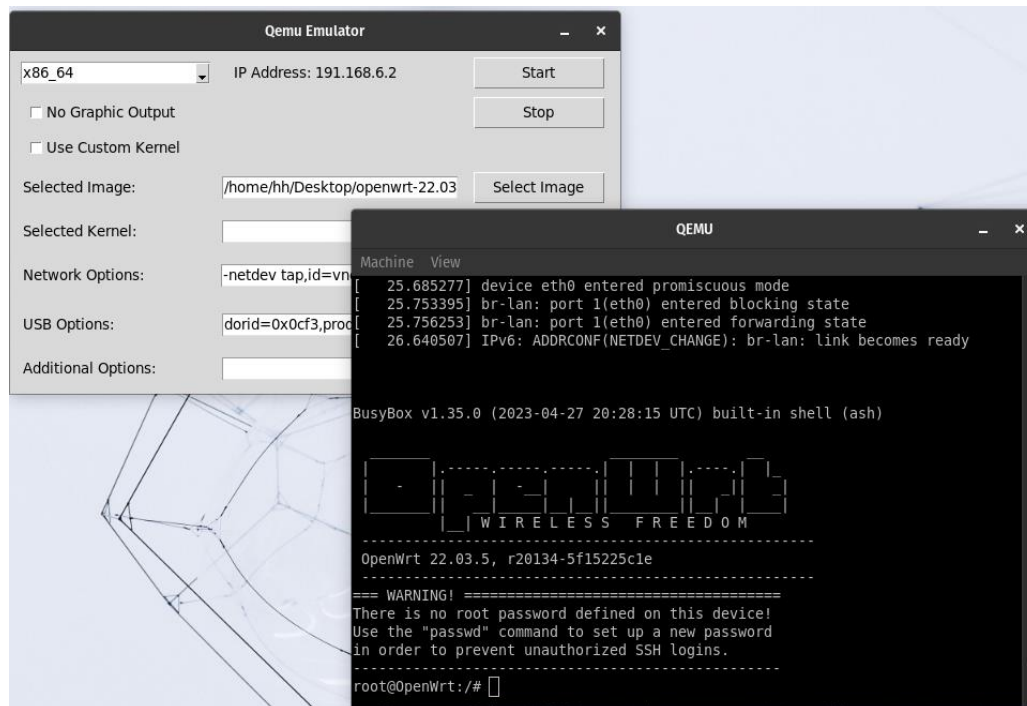


Рис. 4.10 - Запуск ОС

Першим етапом роботи в ОС, буде налаштування мережі.

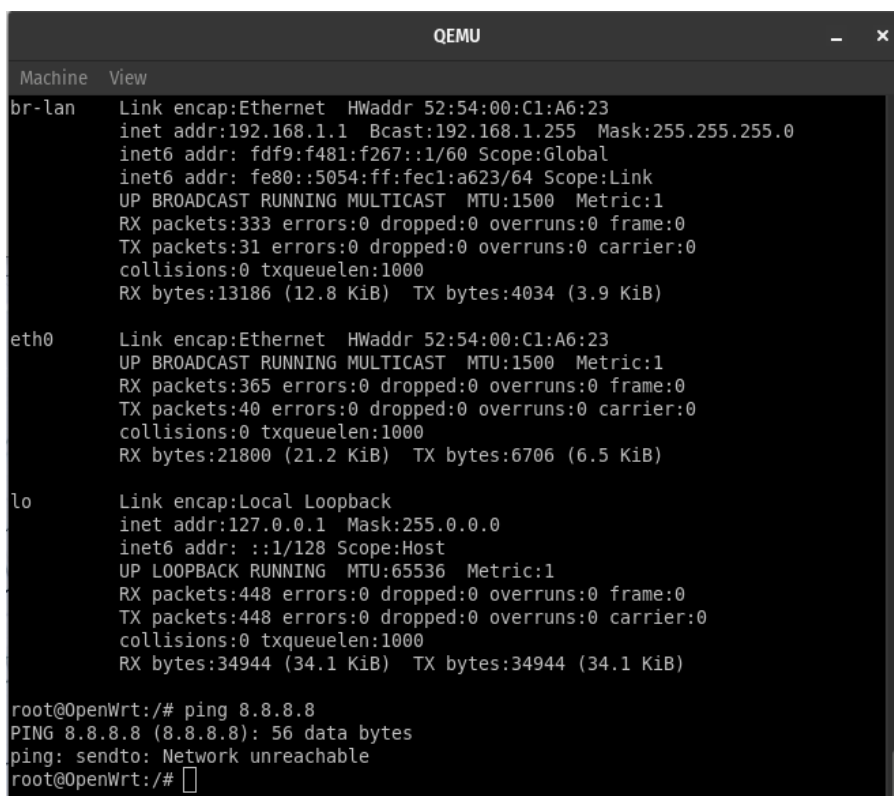


Рис. 4.11 – Робота мережі до редагування

На стандартних налаштуваннях доступу до інтернету ми не маємо, тому міняємо статичну адресу на динамічну за протоколом DHCP, використовуючи команду vi для редагування файлу конфігурації за шляхом /etc/config/network


```

root@OpenWrt:~# ifconfig
br-lan    Link encap:Ethernet  HWaddr 52:54:00:C1:A6:23
          inet addr:191.168.6.72  Bcast:191.168.6.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe01:a623/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4958 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4912 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:413757 (404.0 KiB)  TX bytes:481692 (470.4 KiB)

eth0      Link encap:Ethernet  HWaddr 52:54:00:C1:A6:23
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5430 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4954 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:510533 (498.5 KiB)  TX bytes:488762 (477.3 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:504 errors:0 dropped:0 overruns:0 frame:0
          TX packets:504 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41194 (40.2 KiB)  TX bytes:41194 (40.2 KiB)

root@OpenWrt:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=116 time=17.940 ms
64 bytes from 8.8.8.8: seq=1 ttl=116 time=17.654 ms
^Z[9]+ Stopped                  ping 8.8.8.8
root@OpenWrt:~# ping 191.168.6.1
PING 191.168.6.1 (191.168.6.1): 56 data bytes
64 bytes from 191.168.6.1: seq=0 ttl=64 time=0.610 ms
64 bytes from 191.168.6.1: seq=1 ttl=64 time=0.525 ms
64 bytes from 191.168.6.1: seq=2 ttl=64 time=0.618 ms
^Z[10]+ Stopped                 ping 191.168.6.1
root@OpenWrt:~#

```

Рис. 4.14 – Робота мережі після редагування

Тепер, з доступом до мережі, оновлюємо та встановлюємо потрібні пакети та залежності.

```

root@OpenWrt:~# opkg update
Downloading https://downloads.openwrt.org/releases/22.03.5/targets/x86_64/packages
/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_core
Downloading https://downloads.openwrt.org/releases/22.03.5/targets/x86_64/packages
/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/base/Pa
ckages.gz
Updated list of available packages in /var/opkg-lists/openwrt_base
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/base/Pa
ckages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/luci/Pa
ckages.gz
Updated list of available packages in /var/opkg-lists/openwrt_luci
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/luci/Pa
ckages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/package
s/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_packages
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/package
s/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/routing
/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_routing
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/routing
/Packages.sig
Signature check passed.
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/telepho
ny/Packages.gz
Updated list of available packages in /var/opkg-lists/openwrt_telephony
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/x86_64/telepho
ny/Packages.sig
Signature check passed.
root@OpenWrt:~#

```

Рис. 4.15 – Виконання команди opkg update

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				44
Змн.	Арк.	№ докум.	Підпис	Дата		

Виконаємо команду `opkg list | grep 9271`, що б дізнатися про наявні пакети для нашого USB пристрою.

```
root@OpenWrt:~# opkg list | grep 9271
ath9k-htc-firmware - 20230515-1 - AR9271/AR7010 firmware
kmod-ath9k-htc - 5.10.176+5.15.92-1-1 - This module adds support for wireless adapters based on Atheros USB AR9271 and AR7010 family of chipsets.
root@OpenWrt:~#
```

Рис. 4.16 – Список доступних пакетів

Додатково встановимо сервіс `hostapd`, для функціонування точки доступу, `kmod-usb2` `kmod-usb3` для роботи USB

```
[ 1450.891176] kmodloader: loading kernel modules from /etc/modules.d/*
[ 1450.904433] kmodloader: done loading kernel modules from /etc/modules.d/*
Configuring kmod-crypto-gcm.
[ 1451.074173] kmodloader: loading kernel modules from /etc/modules.d/*
[ 1451.087204] kmodloader: done loading kernel modules from /etc/modules.d/*
Configuring hostapd-common.
Configuring kmod-mac80211.
Configuring ath9k-htc-firmware.
Configuring hostapd.
Configuring kmod-ath.
Configuring kmod-ath9k-common.
Configuring kmod-ath9k-htc.
[ 1452.846934] kmodloader: loading kernel modules from /etc/modules.d/*
[ 1452.859856] kmodloader: done loading kernel modules from /etc/modules.d/*
root@OpenWrt:~#
```

Рис. 4.17 – Успішне встановлення потрібних пакетів

Перезавантажуємо систему, і виконуємо команду `iw list`.

```
[ 60.727978] ieee80211 phy0: Atheros AR9271 Rev:1
root@OpenWrt:~# iw list
Wiphy phy0
  wiphy index: 0
  max # scan SSIDs: 4
  max scan IEs length: 2257 bytes
  max # sched scan SSIDs: 0
  max # match sets: 0
  Retry short limit: 7
  Retry long limit: 4
  Coverage class: 0 (up to 0m)
  Device supports T-DLS.
  Available Antennas: TX 0x1 RX 0x1
  Configured Antennas: TX 0x1 RX 0x1
  Supported interface modes:
    * IBSS
    * managed
    * AP
    * AP/VLAN
    * monitor
    * mesh point
    * P2P-client
    * P2P-GO
    * outside context of a BSS
```

Рис. 4.18 – Виконання команди `iw list`

Бачимо що USB пристрій успішно під'єднано і він підтримує режим точки доступу.

```
root@OpenWrt:/# wifi up
'radio0' is disabled
'radio0' is disabled
root@OpenWrt:/#
```

Рис. 4.19 – Виконання команди wifi up

При виконанні команди wifi up, бачимо що пристрій вимкнений, тому відредагуємо конфігураційний файл за шляхом /etc/config/wireless.

```
config wifi-device 'radio0'
    option type 'mac80211'
    option path 'pci0000:00/0000:00:04.0/usb1/1-1/1-1:1.0'
    option channel '1'
    option band '2g'
    option htmode 'HT20'
    option disabled '0'

config wifi-iface 'default_radio0'
    option device 'radio0'
    option network 'lan'
    option mode 'ap'
    option ssid 'OpenWrt'
    option encryption 'none'
```

Рис. 4.20 – Вміст конфігураційного файлу після редагування

Змінили опцію disabled з 1 на 0, і перезавантажуємо службу network.

```
root@OpenWrt:/# /etc/init.d/network restart
364.970836] br-lan: port 1(eth0) entered disabled state
364.989315] device eth0 left promiscuous mode
364.992353] br-lan: port 1(eth0) entered disabled state
root@OpenWrt:/# [ 368.434859] 8021q: adding VLAN 0 to HW filter on device eth0
368.448846] br-lan: port 1(eth0) entered blocking state
368.451632] br-lan: port 1(eth0) entered disabled state
368.454454] device eth0 entered promiscuous mode
368.472189] br-lan: port 1(eth0) entered blocking state
368.474883] br-lan: port 1(eth0) entered forwarding state
369.508195] IPv6: ADDRCONF(NETDEV_CHANGE): br-lan: link becomes ready
373.717550] br-lan: port 2(wlan0) entered blocking state
373.724035] br-lan: port 2(wlan0) entered disabled state
373.730679] device wlan0 entered promiscuous mode
374.746139] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
374.751495] br-lan: port 2(wlan0) entered blocking state
374.754124] br-lan: port 2(wlan0) entered forwarding state

root@OpenWrt:/# wifi up
/bin/ash: @wifi: not found
root@OpenWrt:/# wifi up
root@OpenWrt:/# [ 427.112435] device wlan0 left promiscuous mode
427.117158] br-lan: port 2(wlan0) entered disabled state
430.403165] br-lan: port 2(wlan0) entered blocking state
430.406372] br-lan: port 2(wlan0) entered disabled state
430.409226] device wlan0 entered promiscuous mode
431.393485] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
431.396806] br-lan: port 2(wlan0) entered blocking state
431.399288] br-lan: port 2(wlan0) entered forwarding state

root@OpenWrt:/#
```

Рис. 4.21 – Виконання команди wifi up після редагування

Тепер все працює коректно. Перейдемо у вебінтерфейс за адресою 191.168.6.72

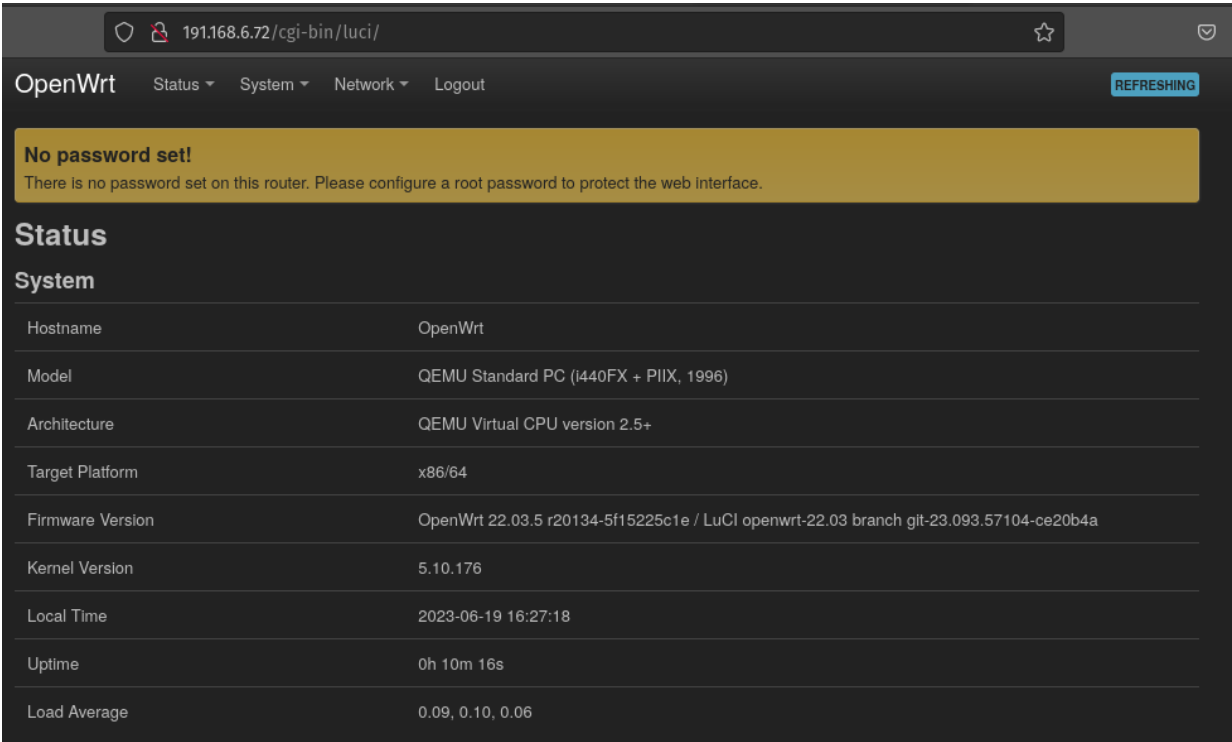


Рис. 4.22 – Вебінтерфейс

Тут ми бачимо відомості про систему і попередження про відсутність пароля, налаштуємо його.

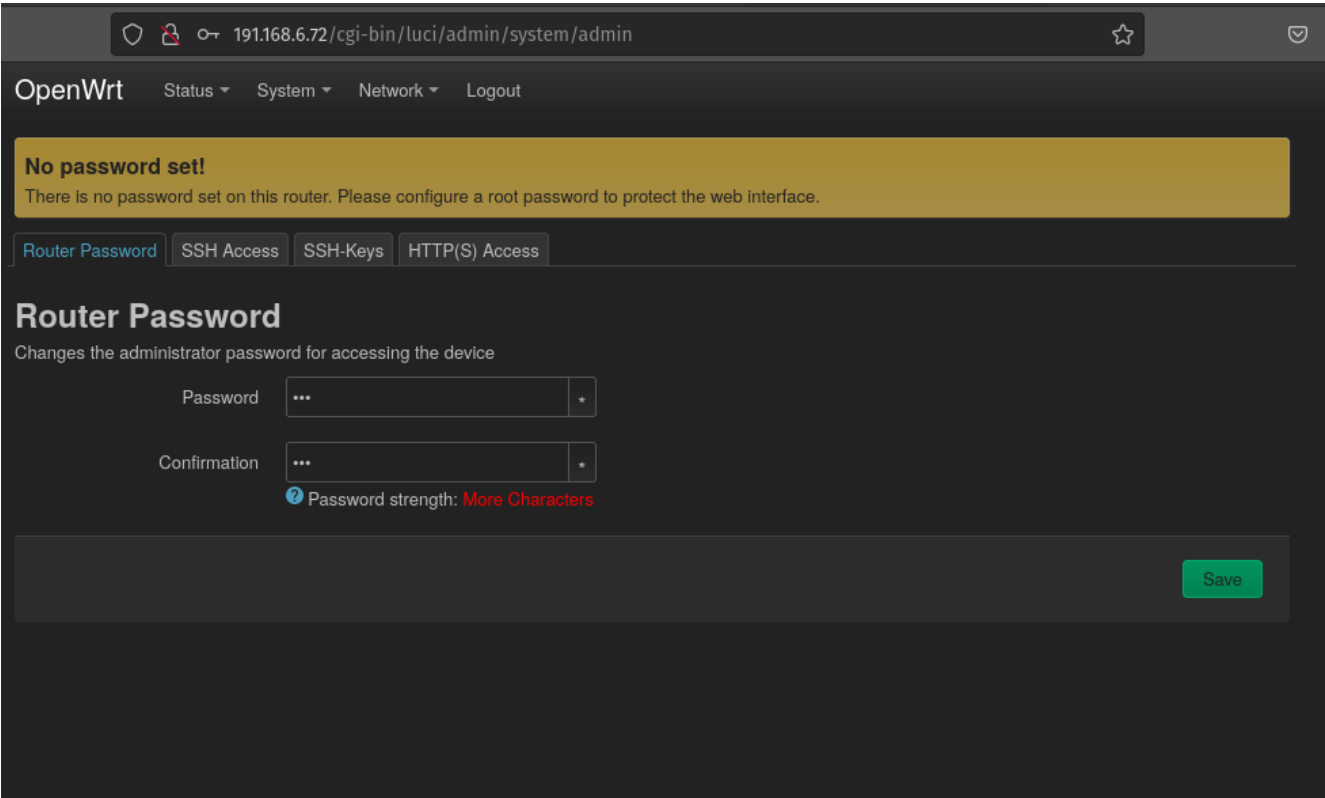


Рис. 4.23 – Налаштування пароля адміністратора у вебінтерфейсі

Налаштували пароль, тепер перейдемо до бездротових інтерфейсів.

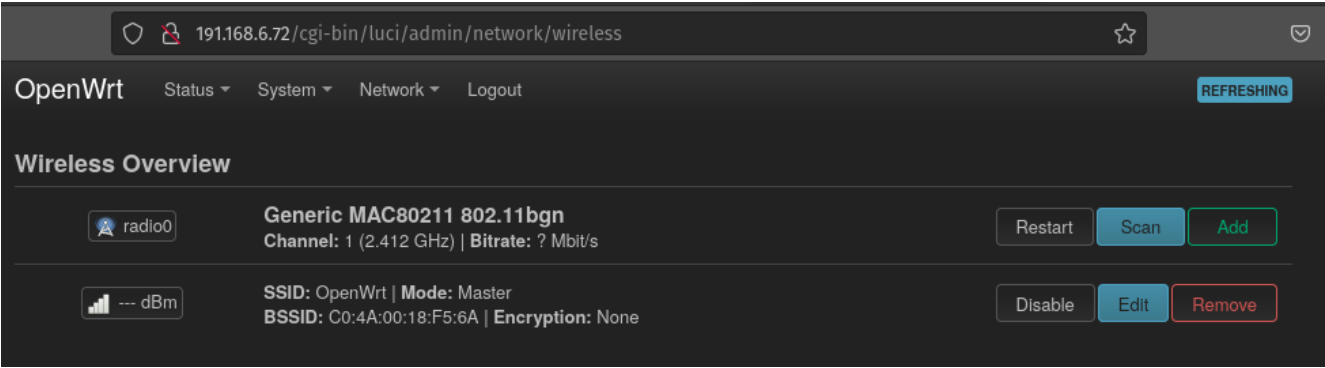


Рис. 4.24 – Доступні бездротові пристрої та мережі у вебінтерфейсі

Налаштуємо Wi-Fi точку доступу, задаємо назву та додаємо конфігурацію безпеки.

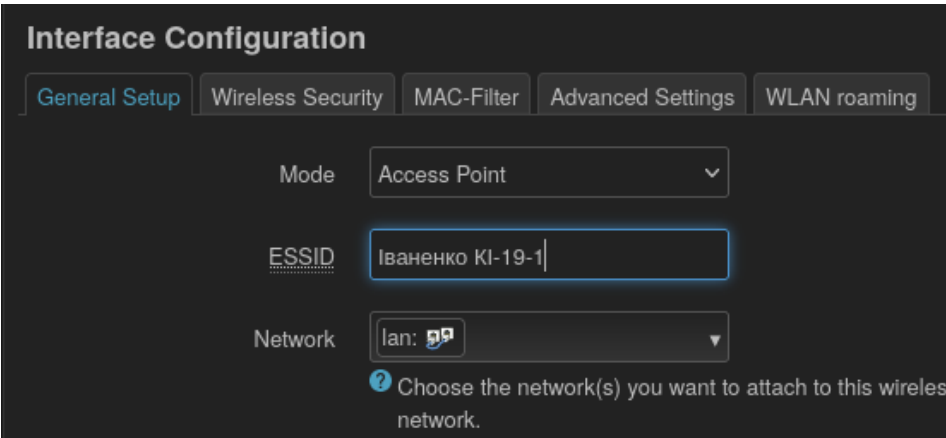


Рис. 4.25 – Налаштування назви

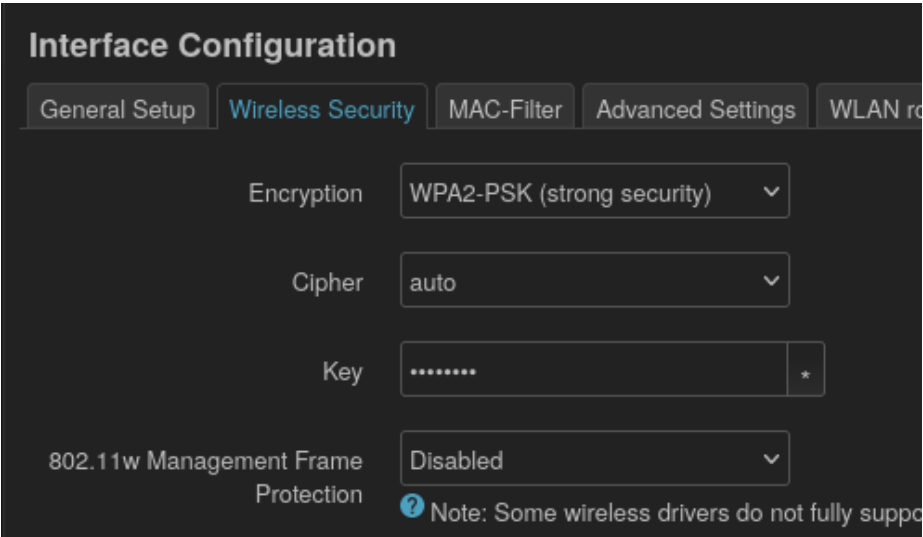


Рис. 4.26 – Налаштування пароля

Зберігаємо налаштування

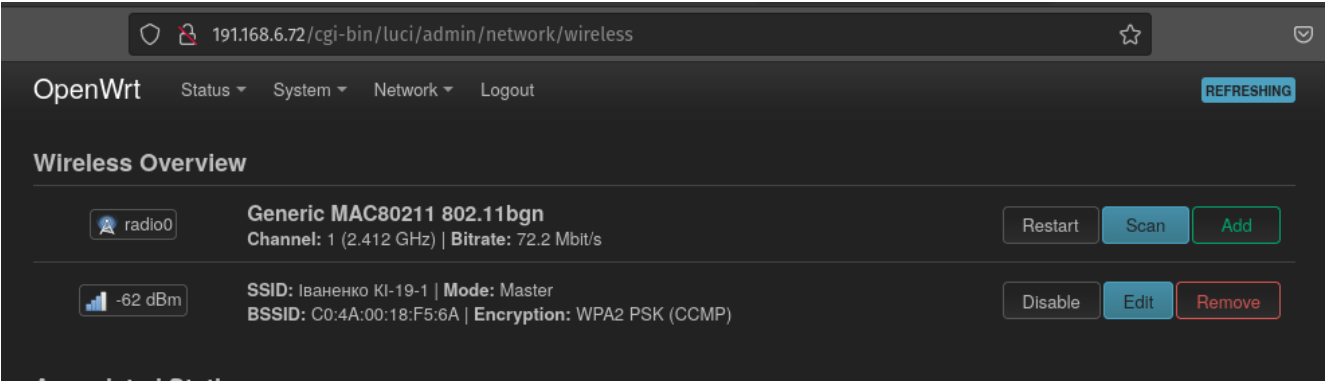


Рис. 4.27 – Доступні налаштовані бездротові пристрої та мережі у вебінтерфейсі

Все працює правильно, тепер спробуємо під'єднатися до мережі використовуючи смартфон

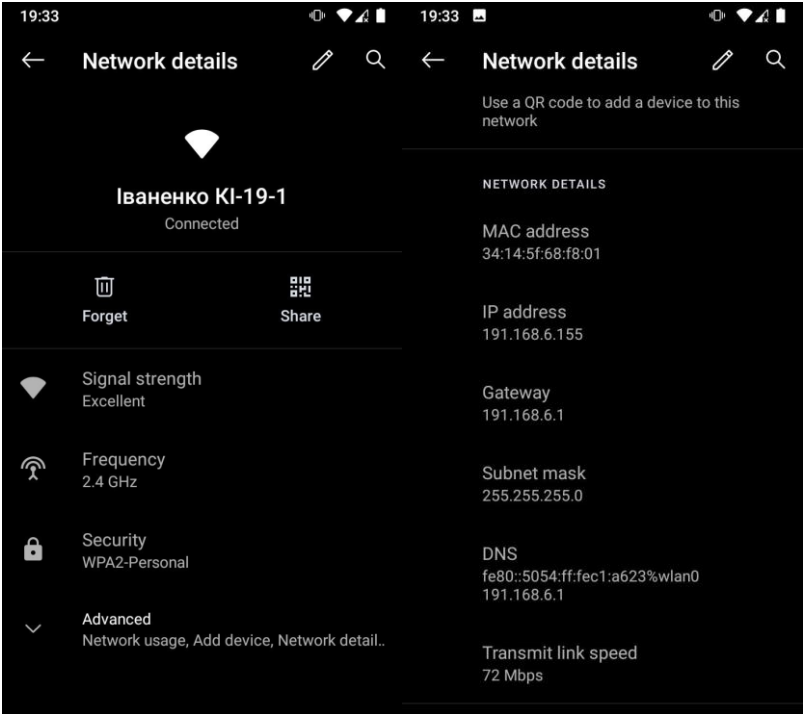


Рис. 4.28 – Відображення мережі на смартфоні

З'єднання стабільне, отримані налаштування мережі правильні, спробуємо перевірити доступ до мереж, використаємо команду ping.

```

19:34 19:34 19:34
~ $ ping 191.168.6.1
PING 191.168.6.1 (191.168.6.1) 56(84) bytes of data:
64 bytes from 191.168.6.1: icmp_seq=1 ttl=64 time=161 ms
64 bytes from 191.168.6.1: icmp_seq=2 ttl=64 time=82.2 ms
64 bytes from 191.168.6.1: icmp_seq=3 ttl=64 time=22.5 ms
^Z
[1]+  Stopped                  ping 191.168.6.1
~ $ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=38.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=34.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=33.2 ms
^Z
[2]+  Stopped                  ping 8.8.8.8
~ $

```

Рис. 4.29 – Перевірка мережі на смартфоні

Доступ до мережі, перевірка працездатності точки доступу на емульованій ОС OpenWrt успішна.

2) Створення точки доступу використовуючи модифікований образ зі сценарію 1, на основі x-86 пристрою.

Машиною на якій буде працювати ОС OpenWrt, буде нетбук на основі процесора Intel Atom з архітектурою x-86. Запишемо модифікований образ на флеш накопичувач програмою Etcher.

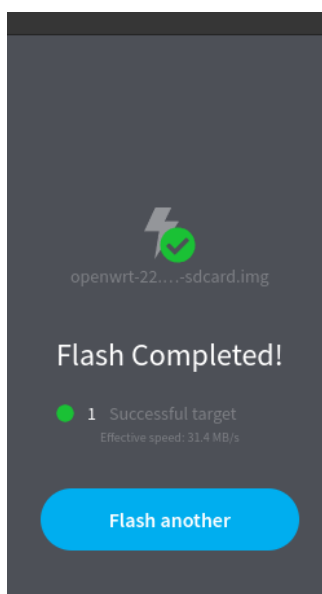


Рис. 4.30 – Запис образу у програмі Etcher

Під'єднаємо накопичувач, Wi-Fi USB адаптер та підключаємо мережу Ethernet.

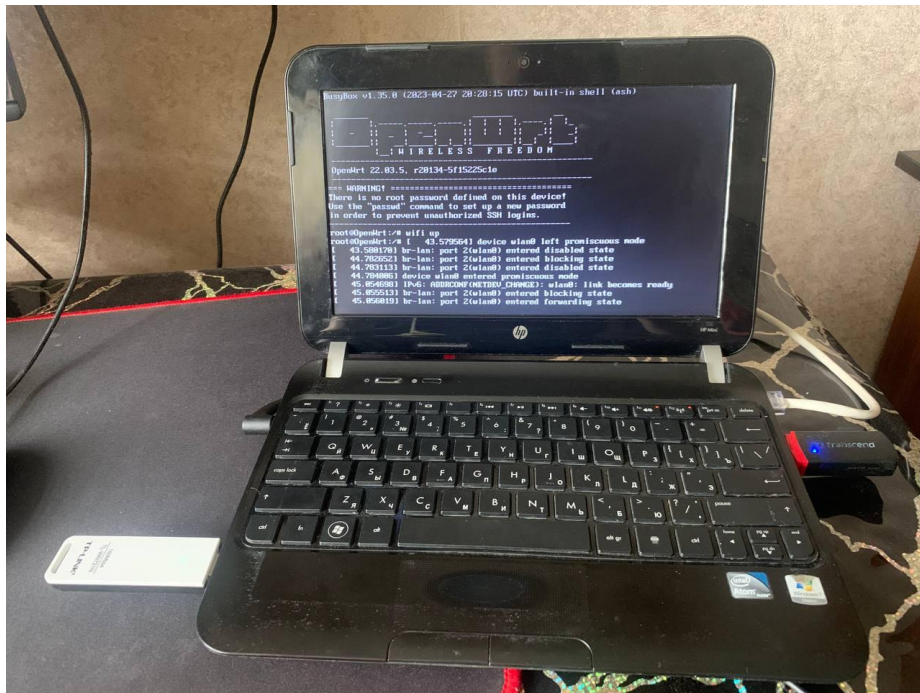


Рис. 4.31 – Робоча машина з архітектурою x-86

Запуск успішний, перевіримо роботу Wi-Fi адаптера та підключення до інтернету.

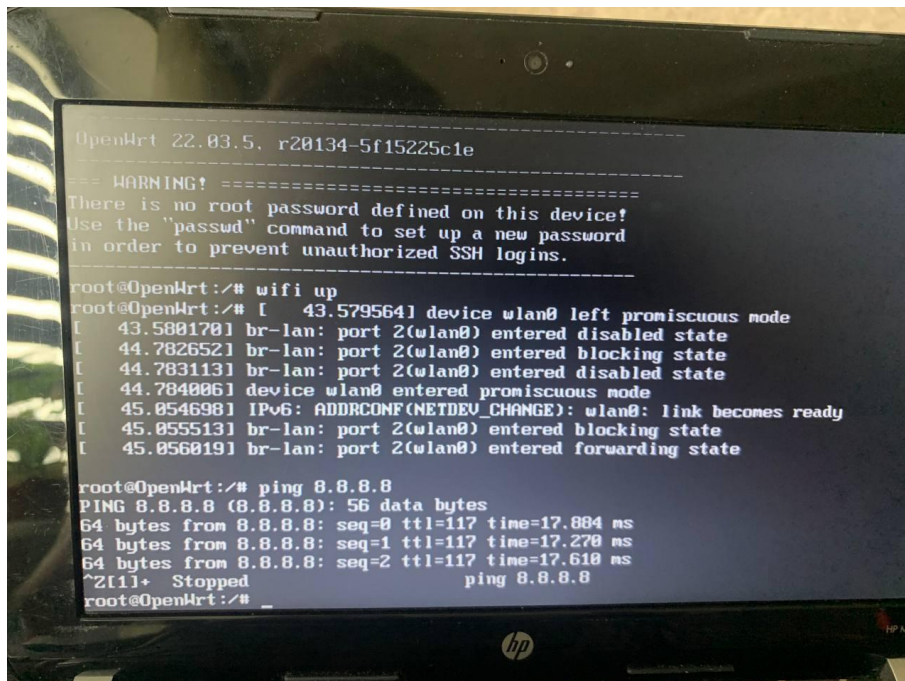


Рис. 4.32 – Перевірка мережі

Зв'язок з інтернетом присутній, під'єднаємось до мережі за допомогою смартфона.

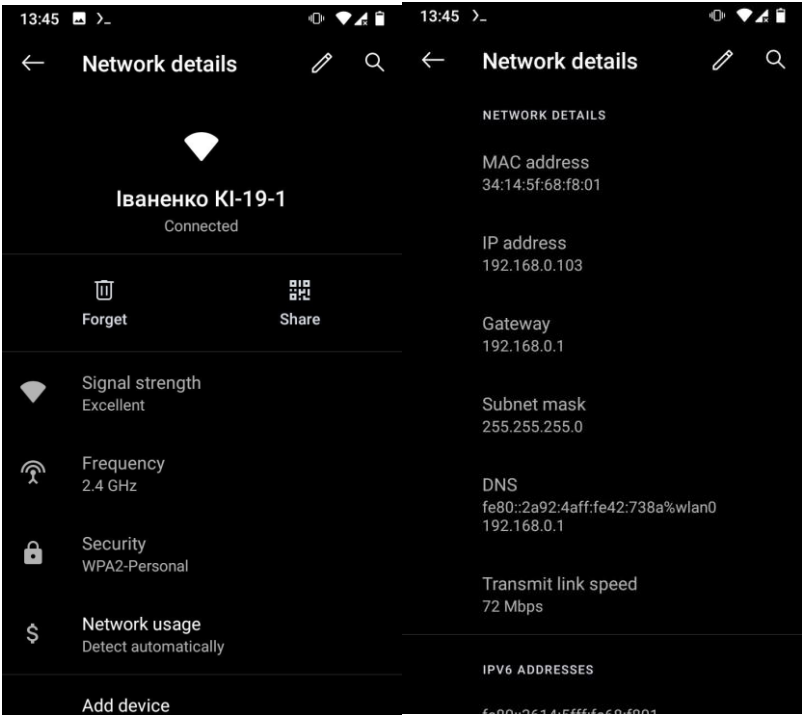


Рис. 4.33 – Відображення мережі на смартфоні

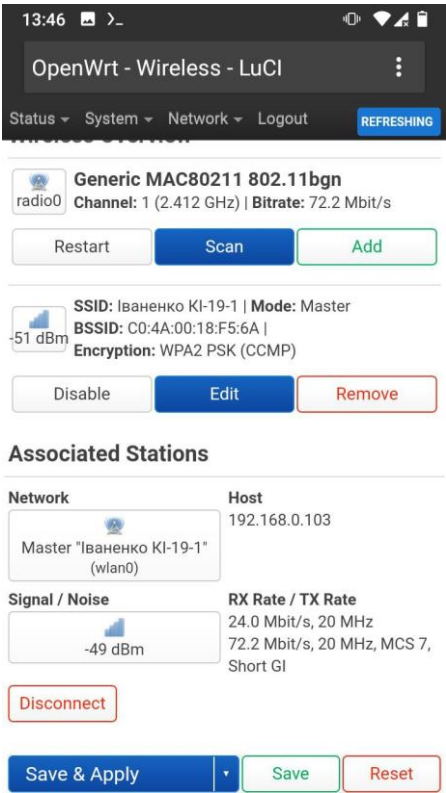


Рис. 4.34 – Відображення вебінтерфейсу мережі на смартфоні

Підключення успішне, смартфон отримав коректні налаштування мережі, перевіriamo з’єднання з мережею на смартфоні.

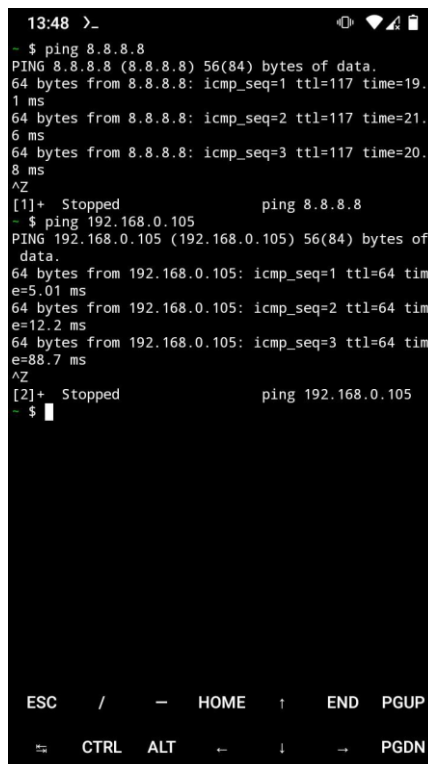


Рис. 4.35 – Перевірка мережі на смартфоні

Все працює правильно, сценарій успішний.

3) Система на основі OpenWrt з архітектурою ARM з налаштованою точкою доступу використовуючи USB Wi-Fi адаптер Atheros AR9271.

Робота буде проведена на основі кроків за сценарію 1:

Завантажимо потрібний образ з офіційного сайту

xunlong_orangepi-r1-squashts-sdcard.img.gz	4
xunlong_orangepi-zero-ext4-sdcard.img.gz	e

Рис. 4.36 – Вибір та завантаження системи

Завантажений образ обираємо в програмі, і налаштовує додаткові параметри. Змінено налаштування архітектури, мережі, доданий параметр –machine, для коректного запуску образу. На жаль, така конфігурація не підтримує підключення USB Wi-Fi пристроїв, тому ми не зможемо перевірити роботу точки доступу на віртуальній машині.

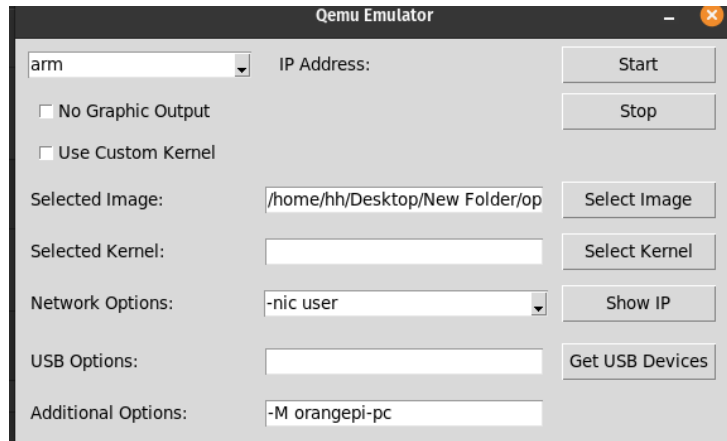


Рис. 4.37 – Налаштування перед запуском системи у програмі
Відредагуємо налаштування мережі, зі статичної адреси використаємо dhcp

```
config interface 'loopback'
    option device 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config globals 'globals'
    option ula_prefix 'fd52:07f4:9ab7::/48'

config device
    option name 'br-lan'
    option type 'bridge'
    list ports 'eth0'

config interface 'lan'
    option device 'br-lan'
    option proto 'dhcp'
```

Рис. 4.38 – Редагування конфігураційного файлу /etc/config/network
Відредагуємо налаштування мережі, зі статичної адреси використаємо dhcp
Перевіримо доступ до інтернету.

```

root@OpenWrt:/# ifconfig
br-lan    Link encap:Ethernet  HWaddr 02:81:4A:58:BD:80
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.0
          inet6 addr: fe80::81:4aff:fe58:bd80/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:130 errors:0 dropped:0 overruns:0 frame:0
          TX packets:153 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9873 (9.6 KiB)  TX bytes:12774 (12.4 KiB)

eth0      Link encap:Ethernet  HWaddr 02:81:4A:58:BD:80
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:131 errors:0 dropped:0 overruns:0 frame:0
          TX packets:162 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11803 (11.5 KiB)  TX bytes:15462 (15.0 KiB)
          Interrupt:43

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:71 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7789 (7.6 KiB)  TX bytes:7789 (7.6 KiB)

root@OpenWrt:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=255 time=57.941 ms
64 bytes from 8.8.8.8: seq=1 ttl=255 time=61.450 ms
64 bytes from 8.8.8.8: seq=2 ttl=255 time=48.165 ms
64 bytes from 8.8.8.8: seq=3 ttl=255 time=48.081 ms
^Z[2]+  Stopped                  ping 8.8.8.8
root@OpenWrt:/#

```

Рис. 4.39 – Перевірка доступу до мережі

З'єднання з інтернетом присутнє, встановимо пакети які були наведені в сценарії 1.

```

root@OpenWrt:/# opkg list | grep AR9271
ath9k-htc-firmware - 20230515-1 - AR9271/AR7010 firmware
kmod-ath9k-htc - 5.10.176+5.15.92-1-1 - This module adds support for wireless adapt
ers based on Atheros USB AR9271 and AR7010 family of chipsets.
root@OpenWrt:/# opkg install ath9k-htc-firmware kmod-ath9k-htc kmod-usb2 hostapd
Installing ath9k-htc-firmware (20230515-1) to root...
Downloading https://downloads.openwrt.org/releases/22.03.5/packages/arm_cortex-a7_n
eon-vfpv4/base/ath9k-htc-firmware 20230515-1_arm_cortex-a7_neon-vfpv4.ipk
Installing kmod-ath9k-htc (5.10.176+5.15.92-1-1) to root...
Downloading https://downloads.openwrt.org/releases/22.03.5/targets/sunxi/cortexa7/p
ackages/kmod-ath9k-htc 5.10.176%2b5.15.92-1-1_arm_cortex-a7_neon-vfpv4.ipk
Installing iw (5.16-1) to root...

```

Рис. 4.40 – Встановлення пакетів

Після встановлення пакетів запишемо модифікований образ на флеш накопичувач.

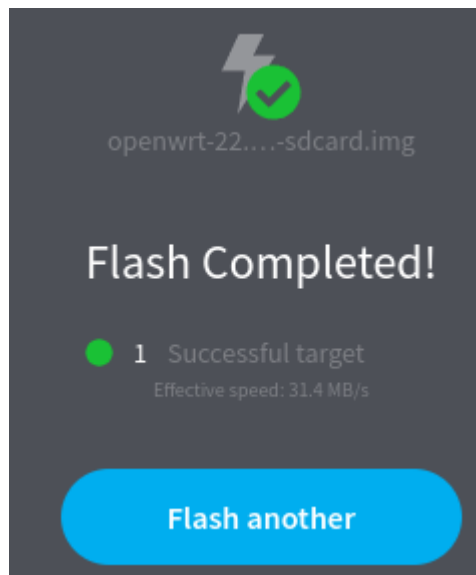


Рис. 4.41 – Запис образу на флеш накопичувач програмою Etcher
Під’єднуємо флеш накопичувач, USB Wi-Fi пристрій та Ethernet до пристрою OrangePI Zero LTS.



Рис. 4.42 – Готовий до роботи пристрій OrangePI Zero LTS.

		Іваненко В.В		
		Фальковський І. Г		
Змн.	Арк.	№ докум.	Підпис	Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

56

Запустимо пристрій та під'єднаємось за допомогою SSH.

```
root@OpenWrt: -
hh@pop-os:~$ ssh root@192.168.0.106

BusyBox v1.35.0 (2023-04-27 20:28:15 UTC) built-in shell (ash)

  _   _   _   _   _   _   _   _   _   _   _   _   _   _   _   _   _   _   _
 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 | |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
  |_|_| W I R E L E S S F R E E D O M

OpenWrt 22.03.5, r20134-5f15225c1e

==== WARNING! =====
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
=====

root@OpenWrt:~#
```

Рис. 4.43 – Підключення до пристрою використовуючи технологію SSH

Перевіримо налаштування мережі та доступ до інтернету.

```
root@OpenWrt:~# ifconfig
br-lan    Link encap:Ethernet  HWaddr 02:42:E8:1D:06:24
          inet addr:192.168.0.106  Bcast:192.168.0.255  Mask:255.255.255
          inet6 addr: fe80::42:e8ff:fe1d:624/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:345 errors:0 dropped:0 overruns:0 frame:0
          TX packets:311 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34486 (33.6 KiB)  TX bytes:30613 (29.8 KiB)

eth0      Link encap:Ethernet  HWaddr 02:42:E8:1D:06:24
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:351 errors:0 dropped:0 overruns:0 frame:0
          TX packets:315 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:39688 (38.7 KiB)  TX bytes:31129 (30.3 KiB)
          Interrupt:43

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:111 errors:0 dropped:0 overruns:0 frame:0
          TX packets:111 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9018 (8.8 KiB)  TX bytes:9018 (8.8 KiB)

root@OpenWrt:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=117 time=17.651 ms
64 bytes from 8.8.8.8: seq=1 ttl=117 time=17.247 ms
64 bytes from 8.8.8.8: seq=2 ttl=117 time=17.156 ms
^Z[1]+  Stopped                  ping 8.8.8.8
root@OpenWrt:~#
```

Рис. 4.44 – Перевірка мережі на пристрої

Інтернет працює, налаштуємо Wi-Fi пристрій.

```

config wifi-device 'radio0'
    option type 'mac80211'
    option path 'platform/soc/1c1b000.usb/usb3/3-1/3-1:1.0'
    option channel '1'
    option band '2g'
    option htmode 'HT20'
    option disabled '0'

config wifi-iface 'default_radio0'
    option device 'radio0'
    option network 'lan'
    option mode 'ap'
    option ssid 'OpenWrt'
    option encryption 'none'

~
:wq

```

Рис. 4.45 – Редагування конфігураційного файлу /etc/config/wireless

Перезавантажимо службу та запустимо роботу пристрою.

```

root@OpenWrt:~# /etc/init.d/network restart
root@OpenWrt:~# wifi up
root@OpenWrt:~#

```

Рис. 4.46 – Запуск Wi-Fi пристрою

Перейдемо до вебінтерфейсу та налаштуємо точку доступу

Рис. 4.47 – Налаштування пароля адміністратора у вебінтерфейсі

Рис. 4.48 – Налаштування назви точки доступу

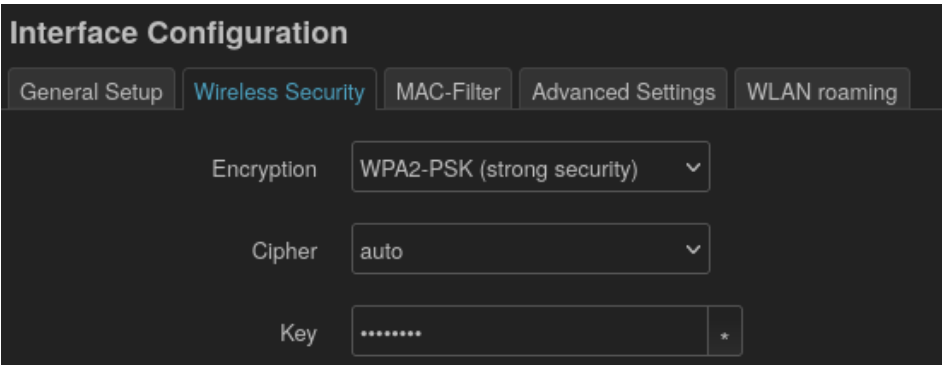


Рис. 4.49 – Налаштування безпеки точки доступу

Налаштування успішне, під’єднаємо смартфон.

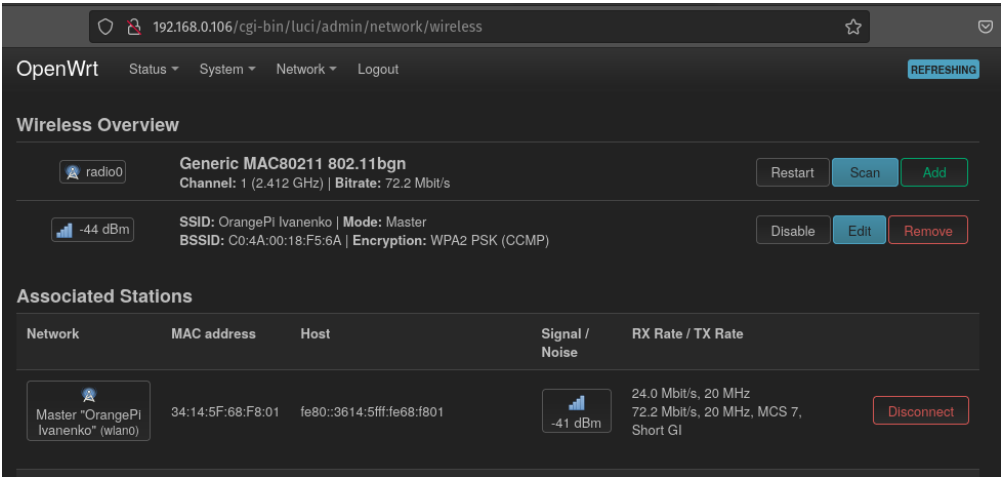


Рис. 4.50 – Відображення підключеного смартфона у вебінтерфейсі

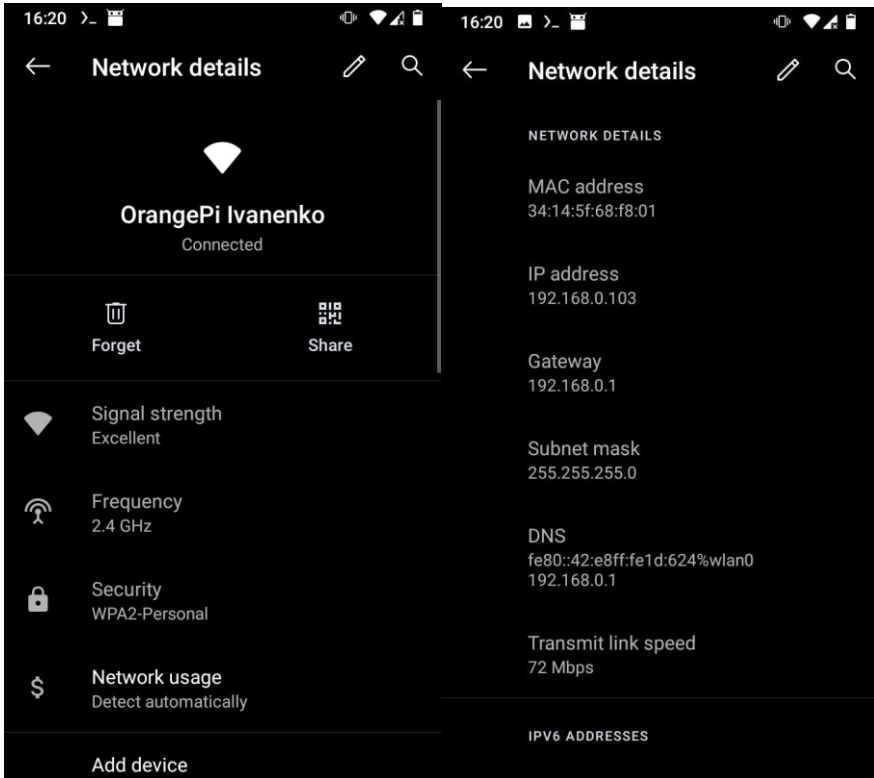


Рис. 4.51 – Відображення мережі у смартфоні

Налаштування вірні, перевіримо зв'язок з мережею

```
16:21 >_
~ $ ping 192.168.0.103
PING 192.168.0.103 (192.168.0.103) 56(84) bytes of data.
64 bytes from 192.168.0.103: icmp_seq=1 ttl=64 time=0.216 ms
64 bytes from 192.168.0.103: icmp_seq=2 ttl=64 time=0.389 ms
64 bytes from 192.168.0.103: icmp_seq=3 ttl=64 time=0.390 ms
^Z
[3]+  Stopped                  ping 192.168.0.103
~ $ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=36.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=36.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=34.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=30.4 ms
^Z
[4]+  Stopped                  ping 8.8.8.8
~ $ █
```

ESC / — HOME ↑ END PGUP
CTRL ALT ← ↓ → PGDN

Рис. 4.52 – Перевірка мережі

Доступ до мережі є, все працює коректно, сценарій роботи успішний.

6.3 Аналіз результатів експериментів

Висновки до розділу 4

В розділі чотири, було проведено підготовку тестових сценаріїв та наборів даних для перевірки функціональності та продуктивності програмного рішення. Це містило в собі розробку різноманітних сценаріїв, які покривали різні аспекти системи емуляції, а також створення відповідних наборів даних для тестування.

ВИСНОВКИ

У даній кваліфікаційній роботі було розглянуто методи та технології для емуляції та віртуалізації вбудованого програмного забезпечення на ОС, реалізований програмний код для емуляції ОС OpenWrt для подальшого налаштування сценаріїв використання.

В першому розділі було наведено огляд операційної системи OpenWrt, поняття та принципи емуляції та віртуалізації, а також архітектурні аспекти, пов'язані з цими технологіями. Також проведено аналіз наявних програмних рішень для емуляції та визначення їх переваг і недоліків.

В другому розділі представлено визначення технічних вимог до програмного забезпечення, виборі мови програмування, засобу для емуляції та засобу для побудови віртуальної мережі.

Розділ три присвячений проектуванню основних функцій, реалізації функціональних модулів та їх взаємодії, а також опису інтерфейсів користувача.

В розділі чотири, було проведено підготовку тестових сценаріїв та наборів даних для перевірки функціональності та продуктивності програмного рішення. Це містило в собі розробку різноманітних сценаріїв, які покривали різні аспекти системи емуляції, а також створення відповідних наборів даних для тестування.

		Іваненко В.В		
		Фальковський І. Г		
Змн.	Арк.	№ докум.	Підпис	Дата

ККІКБ.КР.Б – 123 – 6 – ПЗ

Арк.

61

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- 1) Xen and the art of virtualization, ACM SIGOPS Operating Systems Review, [Електронний ресурс] / P.Barham, B. Dragovic, K. Fraser, K. Harris., 2003. – Режим доступу до ресурсу: <https://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf>

- 2) Virtual machines. [Електронний ресурс] / Elsevier, 2005. – Режим доступу до ресурсу: https://booksite.elsevier.com/samplechapters/9780123858801/Chapter_3.pdf

- 3) Understanding Virtualization: From Desktop to the Enterprise. [Електронний ресурс] / Wiley. Smith, J. 2018– Режим доступу до ресурсу: <https://catalogimages.wiley.com/images/db/pdf/9781119267720.excerpt.pdf>

- 4) Blenk A. Survey on Network Virtualization Hypervisors for Software Defined Networking [Електронний ресурс] / A. Blenk, B. Arsany, R. Martin. – 2015. – Режим доступу до ресурсу: <https://arxiv.org/pdf/1506.07275.pdf>.

- 5) Garfinkel T. When Virtual is Harder than Real: Security Challenges in Virtual Machine Bases computing Environments [Електронний ресурс] / T. Garfinkel, M. Rosenblum // Stanford University Department of Computer Science – Режим доступу до ресурсу: <http://www.stanford.edu/~talg/papers/HOTOS05/virtual-harder-hotos05.pdf>

- 6) Andreas R. Andreas Rivera [Електронний ресурс] / Rivera Andreas. – 2023. – Режим доступу до ресурсу: <https://www.businessnewsdaily.com/6014-pros-cons-virtualization.html>.

- 7) Bellard F. QEMU, a fast and portable dynamic translator [Електронний ресурс] / Bellard // USENIX. – 2005. – Режим доступу до ресурсу: <https://www.usenix.org/legacy/publications/library/proceedings/usenix05/tech/freenix/bellard.html>.

- 8) The QEMU Project. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.qemu.org/>

9) The VirtualBox Documentation. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.virtualbox.org/wiki/Documentation>

10) The VMware Documentation. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.vmware.com/support/pubs/>

11) Linux bridge networking [Електронний ресурс] – Режим доступу до ресурсу: <https://wiki.linuxfoundation.org/networking/bridge>.

12) An introduction to Linux bridging commands and features [Електронний ресурс] – Режим доступу до ресурсу: <https://developers.redhat.com/articles/2022/04/06/introduction-linux-bridging-commands-and-features>.

13) How to Use lsusb in Linux [Електронний ресурс] – Режим доступу до ресурсу: <https://www.howtogeek.com/devops/how-to-use-lsusb-in-linux-with-a-practical-example/>.

14) Qemu Networking [Електронний ресурс] – Режим доступу до ресурсу: <https://wiki.qemu.org/Documentation/Networking>.

15) Network basics /etc/config/network [Електронний ресурс] – Режим доступу до ресурсу: <https://openwrt.org/docs/guide-user/base-system/basic-networking>.

16) USB WiFi adapters that are supported with Linux in-kernel drivers [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/morrownr/USB_WiFi/blob/main/home/USB_WiFi_Adapters_that_are_supported_with_Linux_in-kernel_drivers.md.

17) Enabling a Wi-Fi access point on OpenWrt [Електронний ресурс] – Режим доступу до ресурсу: https://openwrt.org/docs/guide-quick-start/basic_wifi.

18) OpenWrt on x86 hardware [Електронний ресурс] – Режим доступу до ресурсу: https://openwrt.org/docs/guide-user/installation/openwrt_x86.

- 19) Daming D. Towards Automated Dynamic Analysis for Linux-based Embedded Firmware [Електронний ресурс] / D. Daming, M. Egele, Chen – Режим доступу до ресурсу: https://www.dcddcc.com/docs/2016_paper_firmadyne.pdf.
- 20) Kim M. FirmAE: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis [Електронний ресурс] / M. Kim, D. Kim, E. Kim – Режим доступу до ресурсу: https://syssec.kaist.ac.kr/pub/2020/kim_acsac2020.pdf.
- 21) OpenWrt on VirtualBox HowTo [Електронний ресурс] // OpenWrt Wiki – Режим доступу до ресурсу: <https://openwrt.org/docs/guide-user/virtualization/virtualbox-vm>.
- 22) Jeong Y. ARP Spoofing Detector Using OpenWrt // Communication Security in Socialnet-Oriented Cyber Spaces [Електронний ресурс] / Y. Jeong, H. Kim, H. Jo // DOI. – 2022. – Режим доступу до ресурсу: <https://doi.org/10.1155/2022/2196998>.
- 23) Malacea C. Mobility in IP networks using LISP and OpenWRT [Електронний ресурс] / C. Malacea, D. Robu, M. Alexandru. – 2018. – Режим доступу до ресурсу: https://www.afahc.ro/ro/revista/2018_3/RevNo3-2018.pdf#page=85.

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				
Змн.	Арк.	№ докум.	Підпис	Дата		64

ДОДАТКИ

		Іваненко В.В			ККІКБ.КР.Б – 123 – 6 – ПЗ	Арк.
		Фальковський І. Г				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Run.py

```

import subprocess
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
import shutil
import re
root = tk.Tk()
root.title("Qemu Emulator")
root.geometry("570x320")
root.resizable(0, 0)
# Змінні
variable_arch = tk.StringVar()
variable_nographic = tk.BooleanVar()
variable_use_kernel = tk.BooleanVar()
variable_image_path = tk.StringVar()
variable_kernel_path = tk.StringVar()
variable_network_options = tk.StringVar()
variable_usb_options = tk.StringVar(value="-device usb-ehci,id=ehci -device usb-
host,bus=ehci.0,vendorid=0x0cf3,productid=0x9271")
variable_additional_options = tk.StringVar()
variable_image_path = tk.StringVar()
variable_kernel_path = tk.StringVar()
qemu_process = None # Глобальна змінна процесу QEMU

# Створення випадального списку для вибору архітектури
combobox_arch = ttk.Combobox(root, textvariable=variable_arch)
combobox_arch.grid(row=0, column=0, padx=10, pady=10)
combobox_arch['values'] = ("x86_64", "arm", "mips")
# Функція для оновлення значення архітектури
def change_value(event):
    selected_item_arch = combobox_arch.get()
    variable_arch.set(selected_item_arch)
combobox_arch.bind("<<ComboboxSelected>>", change_value)
# Створення прапорців для вибору опцій мережі QEMU
checkbox_nographic = tk.Checkbutton(root, text="No Graphic Output",
variable=variable_nographic)
checkbox_nographic.grid(row=1, column=0, padx=10, pady=5, sticky="w")
checkbox_use_kernel = tk.Checkbutton(root, text="Use Custom Kernel",
variable=variable_use_kernel)
checkbox_use_kernel.grid(row=2, column=0, padx=10, pady=5, sticky="w")
# Створення полів вводу для вибору шляхів до обраного образу і ядра

```

```

image_name_label = tk.Label(root, text="Selected Image:")
image_name_label.grid(row=3, column=0, padx=10, pady=5, sticky="w")
image_textbox = tk.Entry(root, width=27, textvariable=variable_image_path)
image_textbox.grid(row=3, column=1, sticky="w")
image_textbox.xview_moveto(1)

kernel_name_label = tk.Label(root, text="Selected Kernel:")
kernel_name_label.grid(row=4, column=0, padx=10, pady=5, sticky="w")
kernel_textbox = tk.Entry(root, width=27, textvariable=variable_kernel_path)
kernel_textbox.grid(row=4, column=1, sticky="w")
kernel_textbox.xview_moveto(1)
# Створення лейблів і полів вводу для опцій мережі
network_options_label = tk.Label(root, text="Network Options:")
network_options_label.grid(row=5, column=0, padx=10, pady=5, sticky="w")

# Створення лейблів і полів вводу для USB опцій
usb_options_label = tk.Label(root, text="USB Options:")
usb_options_label.grid(row=6, column=0, padx=10, pady=5, sticky="w")
usb_options_textbox = tk.Entry(root, width=27, textvariable=variable_usb_options)
usb_options_textbox.grid(row=6, column=1, sticky="w")
usb_options_textbox.xview_moveto(1)

# Створення лейблів і полів вводу для додаткових опцій
additional_options_label = tk.Label(root, text="Additional Options:")
additional_options_label.grid(row=7, column=0, padx=10, pady=5, sticky="w")

additional_options_textbox = tk.Entry(root, width=27,
textvariable=variable_additional_options)
additional_options_textbox.grid(row=7, column=1, sticky="w")
additional_options_textbox.xview_moveto(1)

# Графічний інтерфейс (GUI) налаштування
def select_image():
    image_path = filedialog.askopenfilename(title="Select OpenWrt Image",
filetypes=[("Disk Image", "*.img")])
    variable_image_path.set(image_path)
def select_kernel():
    kernel_path = filedialog.askopenfilename(title="Select Kernel Image")
    variable_kernel_path.set(kernel_path)
def start_emulator():
    global qemu_process
    qemu_executable = shutil.which("qemu-system-" + variable_arch.get())
    if not qemu_executable:
        print("QEMU executable not found.")

```

```

return

qemu_command = [qemu_executable]
if variable_image_path.get():
    qemu_command.append("-hda")
    qemu_command.append(variable_image_path.get())
if variable_nographic.get():
    qemu_command.append("-nographic")
if variable_use_kernel.get() and variable_kernel_path.get():
    qemu_command.append("-kernel")
    qemu_command.append(variable_kernel_path.get())
network_options = variable_network_options.get()
if network_options:
    qemu_command.extend(network_options.split())
usb_options = variable_usb_options.get()
if usb_options:
    qemu_command.extend(usb_options.split())
additional_options = variable_additional_options.get()
if additional_options:
    qemu_command.extend(additional_options.split())

qemu_process = subprocess.Popen(qemu_command)

def get_usb_devices():
    # Виконання команди 'lsusb'
    output = subprocess.check_output(['lsusb']).decode('utf-8')
    lines = output.split('\n')
    # Вихідні дані 'lsusb' і отримання інформації про пристрої
    device_list = []
    for line in lines:
        if line:
            parts = line.split()
            device_id = parts[5]
            device_vendor = ''.join(parts[6:])
            device_list.append((device_id, device_vendor))

    return device_list

def show_devices():
    # Створення нового вікна
    window = tk.Toplevel(root)
    window.title("USB Devices")
    # Отримання списку USB-пристроїв
    devices = get_usb_devices()

```

```

# Створення лейблу для кожного пристрою
for i, device in enumerate(devices):
    device_label = tk.Label(window, text=f"Device {i+1}: {device[0]} - {device[1]}",
anchor='e', justify='right', wraplength=3000)
    device_label.pack()

def show_interface_ip():
    selected_interface = variable_network_options.get()
    interface_name = re.search(r'ifname=(\w+)', selected_interface).group(1)

    # Виконання команди 'ifconfig', щоб отримати IP-адресу вибраного інтерфейсу
    result = subprocess.run(['ifconfig', interface_name], capture_output=True, text=True)
    output = result.stdout
    ip_match = re.search(r'inet (\d+\.\d+\.\d+\.\d+)', output)

    if ip_match:
        ip_address = ip_match.group(1)
        ip_label.config(text=f"IP Address: {ip_address}")
    else:
        ip_label.config(text="IP Address: Not Available")

ip_label = tk.Label(root, text="IP Address: ")
ip_label.grid(row=0, column=1, padx=10, pady=5, sticky="w")

def select_network_interface():
    # Отримання доступних мережевих інтерфейсів за допомогою команди ifconfig
    result = subprocess.run(['ifconfig'], capture_output=True, text=True)
    output = result.stdout

    # Вихідні дані для отримання назви інтерфейсів
    interface_lines = output.split('\n')
    interfaces = []
    for line in interface_lines:
        if line.startswith('vnet'):
            interface_name = line.split(':')[0]
            interfaces.append(interface_name)

    if len(interfaces) == 0:
        tk.messagebox.showinfo("No Interfaces", "No network interfaces found.") # type:
        ignore
    else:
        # Очистити попередні опції мережі

```

```

variable_network_options.set("")
interface_menu = ttk.Combobox(root,width=26,
textvariable=variable_network_options, values=interfaces)
interface_menu.grid(row=5, column=1, sticky="w")
interface_menu.xview_moveto(1)
interface_menu.current(0)
def set_interface(event):
    selected_interface = interface_menu.get()
    variable_network_options.set(f"-netdev
tap,id={selected_interface},ifname={selected_interface},script=no,downscript=no -
device virtio-net-pci,netdev={selected_interface},mac=52:54:00:c1:a6:23")
    interface_menu.bind("<<ComboboxSelected>>", set_interface)
def stop_emulator():
    global qemu_process
    if qemu_process:
        qemu_process.terminate()
        qemu_process.wait()
        qemu_process = None
# Кнопки
select_image_button = tk.Button(root, text="Select Image", command=select_image)
select_image_button.grid(row=3, column=2, padx=10, pady=5, sticky="w")
select_kernel_button = tk.Button(root, text="Select Kernel", command=select_kernel)
select_kernel_button.grid(row=4, column=2, padx=10, pady=5, sticky="w")
start_button = tk.Button(root, text="Start", command=start_emulator)
start_button.grid(row=0, column=2, padx=10, pady=0, sticky="w")
stop_button = tk.Button(root, text="Stop", command=stop_emulator)
stop_button.grid(row=1, column=2, padx=10,sticky="w")
get_devices_button = tk.Button(root, text="Get USB Devices",
command=show_devices)
get_devices_button.grid(row=6, column=2, padx=10, pady=10, sticky="w")
show_ip_button = tk.Button(root, text="Show IP", command=show_interface_ip)
show_ip_button.grid(row=5, column=2, padx=10, pady=5, sticky="w")
button_list = [select_image_button, select_kernel_button, start_button, stop_button,
get_devices_button,show_ip_button]
# Налаштування розміру кнопок
for button in button_list:
    button.configure(width=12, height=1)
select_network_interface()
root.mainloop()

```

Network.py

```
import libvirt
import xml.etree.ElementTree as ET
import tkinter as tk
from tkinter import messagebox
import subprocess
import getpass

def create_virbr(network_name, bridge_name, ip_address, netmask, vnet_name):
# Створення XML-опису мережі
    xml_desc = f'''
    <network>
        <name>{network_name}</name>
        <bridge name='{bridge_name}' stp='on' delay='0'/>
        <forward mode='nat'/>
        <ip address='{ip_address}' netmask='{netmask}'>
            <dhcp>
                <range start='{ip_address[:-1]}2' end='{ip_address[:-1]}254'/>
            </dhcp>
        </ip>
    </network>
    '''

# Підключення до гіпервізора
    conn = libvirt.open()
    network = conn.networkCreateXML(xml_desc)
    messagebox.showinfo("Success", f"Virtual bridge '{bridge_name}' and '{network_name}' created successfully.")

# Створення та підключення віртуального мережевого інтерфейсу
    username = getpass.getuser()
    cmds = [
        f"sudo tunctl -t {vnet_name} -u {username}",
        f"sudo ip link set {vnet_name} up",
        f"sudo brctl addif {bridge_name} {vnet_name}",
        f"sudo ifconfig {vnet_name} {ip_address[:-1]}2 netmask {netmask}"
    ]
    for cmd in cmds:
        subprocess.run(cmd.split())

def create_bridge():
    def submit():
# Отримання значень з полів введення
        network_name = network_name_entry.get()
        bridge_name = bridge_name_entry.get()
        ip_address = ip_address_entry.get()
        netmask = netmask_entry.get()
        vnet_name = vnet_name_entry.get()

# Виклик функції створення віртуального моста
```

```

        create_virbr(network_name, bridge_name, ip_address, netmask, vnet_name)
def destroy():
# Отримання значень з полів введення
    bridge_name = bridge_name_entry.get()
    vnet_name = vnet_name_entry.get()
# Знищення віртуального моста та віртуального мережевого інтерфейсу
    subprocess.run(['sudo', 'virsh', 'net-destroy', bridge_name])
    subprocess.run(['sudo', 'ip', 'link', 'delete', vnet_name])
    messagebox.showinfo("Success", f"Virtual bridge '{bridge_name}' and
'{vnet_name}' destroyed successfully.")

root = tk.Tk()
root.title("Create Virtual Bridge")

network_name_label = tk.Label(root, text="Network name:")
network_name_label.pack()
network_name_entry = tk.Entry(root)
network_name_entry.pack()

bridge_name_label = tk.Label(root, text="Bridge name:")
bridge_name_label.pack()
bridge_name_entry = tk.Entry(root)
bridge_name_entry.pack()

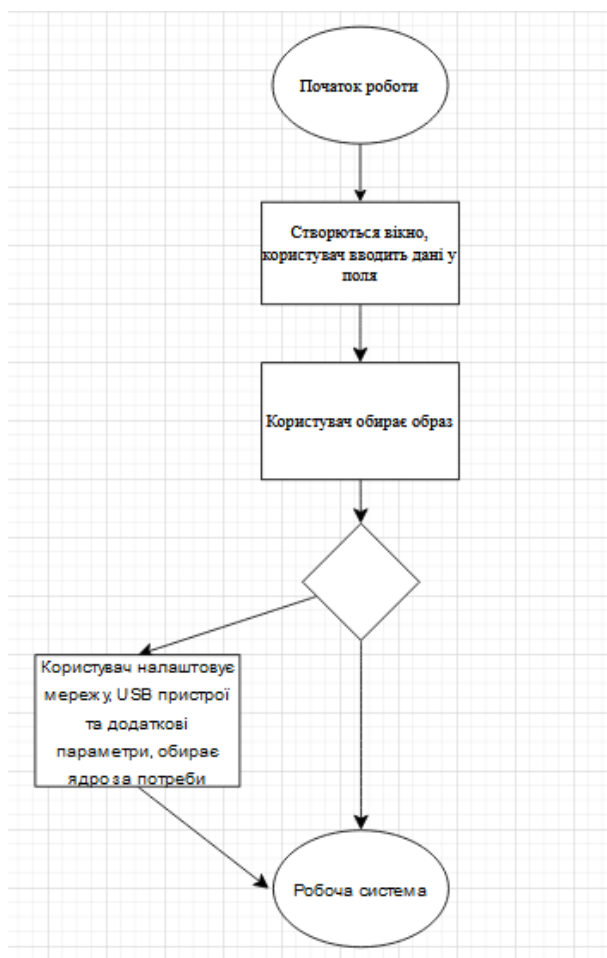
ip_address_label = tk.Label(root, text="IP Address:")
ip_address_label.pack()
ip_address_entry = tk.Entry(root)
ip_address_entry.pack()

netmask_label = tk.Label(root, text="Netmask:")
netmask_label.pack()
netmask_entry = tk.Entry(root)
netmask_entry.pack()

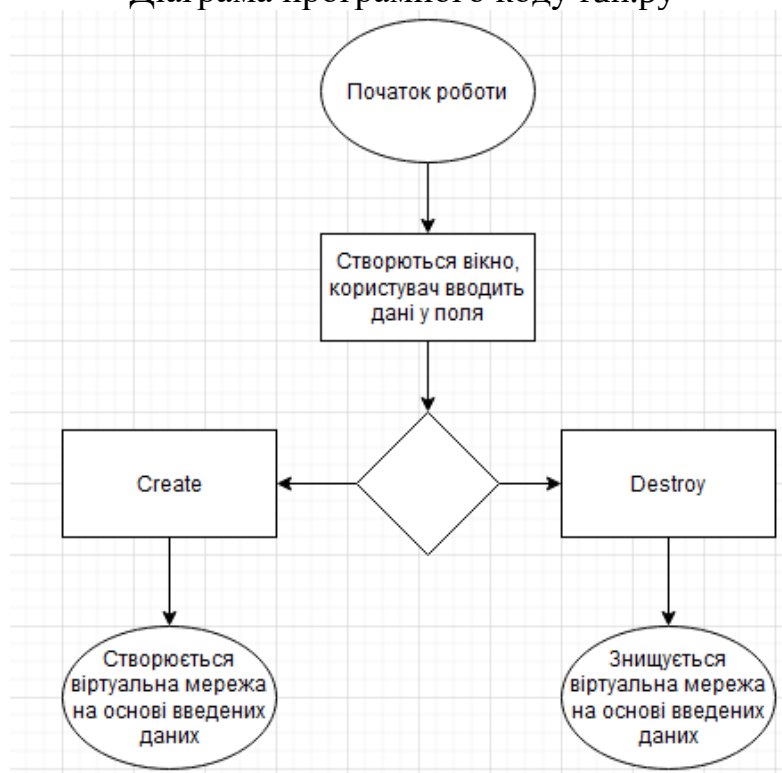
vnet_name_label = tk.Label(root, text="Vnet:")
vnet_name_label.pack()
vnet_name_entry = tk.Entry(root)
vnet_name_entry.pack()

submit_button = tk.Button(root, width=15, text="Create", command=submit)
submit_button.pack()
destroy_button = tk.Button(root, width=15, text="Destroy", command=destroy)
destroy_button.pack()
root.mainloop()
create_bridge()

```



Діаграма програмного коду run.py



Діаграма програмного коду network.py