ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Автономна платформа віддаленого керування робочим місцем на базі Raspberry Pi»

на здобуття освітнього ступеня магістра
зі спеціальності 123 Комп'ютерна інженерія
освітньо-професійної програми «Комп'ютерні системи та мережі»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Михайло ТОКАР
(підпис)

Виконав: здобувач вищої освіти групи КСДМ-62
Михайло ТОКАР

Керівник: Віталій ВОЛОХІН
доцент кафедри, к.т.н., доцент

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерна інженерія

Ступінь вищої освіти Магістр

Спеціальність 123 Комп'ютерна інженерія

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ Наталія ЛАЩЕВСЬКА
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Токару Михайлу Віталійовичу

1. Тема кваліфікаційної роботи: «Автономна платформа віддаленого керування робочим місцем на базі Raspberry Pi»

керівник кваліфікаційної роботи Віталій ВОЛОХІН, доцент кафедри, к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «___» _____ 2025 р. № _____.

2. Строк подання кваліфікаційної роботи «26» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, наукові статті у відкритому доступі, керівництва щодо написання скриптів для Raspberry Pi, вимоги щодо затримок передачі сигналів.

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

1. Дослідження сучасних способів віддаленого доступу до робочого місця.
2. Аналіз ринку готових рішень сфери віддаленого доступу.
3. Розробка та аналіз платформи віддаленого керування робочим місцем на базі Raspberry Pi.

5. Перелік ілюстративного матеріалу: *презентація*

1. Використання Raspberry Pi 4B.
2. Компонентна база проєкту.
3. Програмна частина системи.
4. Фізичне виконання проєкту.
5. Сценарії використання проєкту.
6. Підсистема трансляції відеосигналу.
7. Підсистема керування HID.
8. Підсистема керування живленням ПК.
9. Відеотрансляція через VPN.

6. Дата видачі завдання «01» вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	08.09.2025	Виконано
2	Пошук та аналіз матеріалів щодо розвитку технологій віддаленого доступу до ПК	17.09.2025	Виконано
3	Дослідження ринку доступних рішень	02.10.2025	Виконано
4	Аналіз особливостей розробки систем віддаленого доступу на базі Raspberry Pi 4B	12.10.2025	Виконано
5	Розробка та збірка системи віддаленого доступу	02.11.2025	Виконано
6	Економічний аналіз та розробка можливих модифікацій системи	20.11.2025	Виконано
7	Оформлення роботи	30.11.2025	Виконано
8	Розробка демонстраційних матеріалів	08.12.2025	Виконано
9	Попередній захист роботи	16.12.2025	Виконано

Здобувач вищої освіти _____ Михайло ТОКАР
(підпис)

Керівник
кваліфікаційної роботи _____ Віталій ВОЛОХІН
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 107 стор., 5 табл., 60 рис., 50 джерел.

Мета роботи – розробка та реалізація системи віддаленого управління та моніторингу комп'ютерів на базі одноплатного комп'ютера Raspberry Pi 4B.

Об'єкт дослідження – процес автоматизації віддаленого доступу до робочої станції з використанням одноплатного комп'ютера Raspberry Pi 4B.

Предмет дослідження – методи та підходи віддаленого керування для забезпечення одночасно низької затримки обробки сигналів, економічної ефективності та надійності функціонування 24/7.

Короткий зміст роботи: В роботі розглянуто розробку та збірку системи та модулів віддаленого управління та моніторингу комп'ютерів на базі одноплатного комп'ютера Raspberry Pi 4B.

Проведено аналіз сучасних методів розробки автоматизованих систем віддаленого керування робочим місцем на базі одноплатного комп'ютера Raspberry Pi.

Розроблено та оптимізовано модуль віддаленого керування робочим місцем.

Проведено експерименти для перевірки та подальшого аналізу майбутнього вдосконалення проєкту.

КЛЮЧОВІ СЛОВА: ВІДДАЛЕНИЙ ДОСТУП, KVM, ІНТЕРНЕТ РЕЧЕЙ (IOT), RASPBERRY PI, ОДНОПЛАТНИЙ КОМП'ЮТЕР, МІНІМІЗАЦІЯ ЗАТРИМКИ, АВТОМАТИЗАЦІЯ, ВІДМОВОСТІЙКІСТЬ, КАРТА ВІДЕОЗАХОПЛЕННЯ.

ABSTRACT

Text part of the master's qualification work: 107 pages, *60* pictures, 5 table, 50 sources.

The purpose of the work is to develop and implement a remote computer control and monitoring system based on the Raspberry Pi 4B single-board computer.

Object of research is the process of automating remote access to a workstation using the Raspberry Pi 4B single-board computer.

Subject of research covers remote control methods and approaches designed to ensure low signal processing latency, cost-efficiency, and reliable 24/7 operation simultaneously.

Summary of the work: The paper discusses the development and assembly of a remote computer control and monitoring system and its modules based on the Raspberry Pi 4B single-board computer. An analysis of modern methods for developing automated remote workstation control systems based on the Raspberry Pi single-board computer was conducted. A remote workstation control module was developed and optimized. Experiments were conducted to verify the system and analyze potential future improvements to the project.

KEYWORDS: REMOTE ACCESS, KVM, INTERNET OF THINGS (IOT), RASPBERRY PI, SINGLE-BOARD COMPUTER, LATENCY MINIMIZATION, AUTOMATION, FAULT TOLERANCE, VIDEO CAPTURE CARD.

ЗМІСТ

ВСТУП 10

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ВІДДАЛЕНИХ РОБОЧИХ МІСЦЬ, КОМУНІКАЦІЙНИХ ПРОТОКОЛІВ ТА ІОТ 12

1.1. Історія, концепції та архітектури IoT. Ключові поняття: датчики, мережа, обробка, керування. Трирівнева та п’ятирівнева моделі IoT, хмарні платформи 12

1.2. Принципи та схеми побудови систем моніторингу, віддаленого керування та автоматизації 23

1.3. Raspberry Pi як платформа для IoT-проектів та розумного дому. Його характеристики, переваги та обмеження 32

1.4. Протоколи безпеки віддаленого доступу 40

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ, АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ ТА ВИБІР КОМПОНЕНТІВ 44

2.1 Постановка задачі проєкту, його функціональні вимоги, модульність системи, обмеження, основні сценарії використання. 44

2.2 Методи віддаленого увімкнення ПК. Огляд існуючих систем віддаленого керування та моніторингу робочого місця. Порівняння створюваної системи з існуючими рішеннями 52

2.3 Архітектура автономних платформ на базі одноплатних комп’ютерів. Принцип реалізації таких платформ на базі Raspberry Pi. Загальна архітектура створюваної системи 64

2.4 Обґрунтування вибору апаратних рішень 68

2.5 Обґрунтування вибору програмних рішень 74

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ ТА ЇЇ РЕЗУЛЬТАТИ 79

3.1 Підготовка робочого середовища, налаштування RPi 4B, встановлення ОС, драйверів, фізична інсталяція оптопар та сенсора 79

3.2 Реалізація підсистеми захоплення та трансляції відеопотоку, а також миші та клавіатури 89
3.3 Створення веб-серверу для керування кнопками на передній панелі ПК 99
3.4 Організація захищеного віддаленого доступу до системи IP-KVM 105
3.5 Аналіз часових затримок системи 108
РОЗДІЛ 4. ПЕРСПЕКТИВИ ПОДАЛЬШОГО РОЗВИТКУ 111
ВИСНОВКИ 115
ПЕРЕЛІК ПОСИЛАНЬ 118
ДОДАТОК А 124
ДОДАТОК Б 129
ДОДАТОК В. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 132

ВСТУП

Глобальні виклики останніх років, і особливо війна в нашій країні, перетворили віддалену роботу з привілею для айтівців на життєву необхідність для мільйонів людей. Люди навчилися керувати бізнесом та складними процесами через екрани моніторів, іноді перебуваючи за сотні кілометрів від своїх робочих місць. Проте за цією цифровою свободою ховається серйозна вразливість: ми стали заручниками стабільності програмного забезпечення. Звісно, інструменти на кшталт TeamViewer, RDP чи SSH стали для нас звичною буденністю, але вони мають одну суттєву ваду – вони працюють лише доти, доки «жива» операційна система. Уявіть ситуацію: завис Windows, збилися налаштування BIOS після оновлення або просто комп'ютер не зміг коректно завантажитися. У цей момент ваша потужна робоча станція миттєво перетворюється на недосяжну купу «заліза». Цифровий міст зникає, і щоб просто натиснути фізичну кнопку перезавантаження, доводиться витрачати години на дорогу.

У світі великих корпоративних серверів ця проблема «останньої милі» давно вирішена. Там використовують спеціалізовані апаратні інтерфейси IPMI, iLO, iDRAC, які дозволяють адміністратору буквально «бачити» екран сервера і керувати ним, навіть якщо операційна система взагалі відсутня. Але звичайні комп’ютери – потужні станції дизайнерів, інженерів, лабораторне обладнання чи сервери малого бізнесу – такої розкоші зазвичай позбавлені. Ринок пропонує комерційні IP-KVM рішення, які могли б врятувати ситуацію, але вони часто коштують невиправдано дорого для масового використання. До того ж, це переважно закриті «чорні скриньки», які неможливо перевірити на безпеку чи адаптувати під свої специфічні потреби.

Тут на допомогу приходять сучасні технології Інтернету речей IoT [1]. Маленькі, але потужні одноплатні комп'ютери, такі як Raspberry Pi, вже давно переросли статус іграшок для ентузіастів. Вони відкривають нам шлях до

створення власних, доступних та ефективних інструментів. Логіка проста: якщо розумна розетка може вмикатися зі смартфона, то чому критично важлива IT-інфраструктура не може мати «розумного помічника», який дозволить перезавантажити завислий сервер чи провести діагностику обладнання, незалежно від стану основної системи?

Ця магістерська робота присвячена створенню саме такої автономної платформи. Ідея полягає в тому, щоб винести керування на окремий, незалежний пристрій – створити своєрідний «запасний вхід» для адміністратора. Такий підхід, як Edge Computing, дозволяє організувати надійний канал управління, який працюватиме навіть тоді, коли основний комп'ютер повністю не відповідає. Проєкт передбачає не просто дистанційне натискання кнопок, а й створення захищеного середовища з власним безперебійним живленням та безпечним доступом. Мета мого дослідження – довести, що професійний рівень надійності та контролю може бути доступним, гнучким і не обов'язково вимагати колосальних бюджетів.

Апробація результатів та публікації, які були опубліковані в ході проведення дослідження:

1. Токар М. В., Волохін В. В., СУЧАСНІ МЕТОДИ АВТЕНТИФІКАЦІЇ ТА РОЗМЕЖУВАННЯ ДОСТУПУ В ФІЗИЧНИХ СИСТЕМАХ ВІДДАЛЕНОГО АДМІНІСТРУВАННЯ. *Науковий простір: аналіз, сучасний стан, тренди та перспективи* : IX Міжнар. студент. наук. конф., м. Вінниця, 26 груд. 2025 р. URL: https://doi.org/10.62732/liga-inter-26.12.2025 (дата звернення: 24.12.2025).

2. Токар М. В., Волохін В. В. АПАРАТНА РЕАЛІЗАЦІЯ ДИСТАНЦІЙНОГО КЕРУВАННЯ ЖИВЛЕННЯМ ПК З ВИКОРИСТАННЯМ ГАЛЬВАНІЧНОЇ РОЗВ’ЯЗКИ. *Традиційні та інноваційні підходи до наукових досліджень* : X Міжнар. наук. конф., м. Луцьк, 26 груд. 2025 р. С. 313-316. URL: https://doi.org/10.62731/mcnd-26.12.2025 (дата звернення: 24.12.2025).

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ВІДДАЛЕНИХ РОБОЧИХ МІСЦЬ, КОМУНІКАЦІЙНИХ ПРОТОКОЛІВ ТА ІОТ

1.1 Історія, концепції та архітектури IoT. Ключові поняття: датчики, мережа, обробка, керування. Трирівнева та п’ятирівнева моделі IoT, хмарні платформи.

Визначення та еволюція IoT.

Сучасний етап науково-технічного прогресу характеризується безпрецедентною конвергенцією цифрових обчислювальних систем та фізичних об'єктів, що знайшло своє відображення у парадигмі Інтернету речей IoT. Згідно з визначенням Сектору стандартизації електрозв'язку Міжнародного союзу електрозв'язку ITU-T, IoT інтерпретується як глобальна інфраструктура інформаційного суспільства, що уможливлює надання просунутих послуг шляхом фізичного та віртуального взаємозв'язку речей на основі існуючих та перспективних інформаційно-комунікаційних технологій [2]. Це мережева структура, в якій об'єкти, оснащені сенсорами, вбудованим програмним забезпеченням та комунікаційними інтерфейсами, здатні здійснювати збір та обмін даними з іншими системами через глобальну мережу Інтернет, приймаючи рішення без безпосереднього втручання людини. Ключова відмінність IoT від традиційного Інтернету полягає у зміні суб'єкта генерації даних: якщо у класичній мережі контент створювали люди, то в екосистемі IoT генераторами та споживачами трафіку стають автономні пристрої, здатні приймати рішення без безпосереднього втручання оператора [3].

Історичне виникнення цієї концепції бере свій початок задовго до появи самого терміну. Технологічні передумови сформувалися наприкінці XX століття з розвитком технологій міжмашинної взаємодії Machine-to-Machine, M2M.

Технології M2M, що домінували у 1990-х та початку 2000-х років, забезпечували зв'язок між пристроями через промислові протоколи, наприклад, Modbus, Profibus або стільниковий зв'язок «точка-точка», залишаючись ізольованими від глобальної мережі та фокусуючись на вузькоспеціалізованих задачах телеметрії. Головна відмінність M2M від сучасного IoT полягала в ізольованості систем та відсутності вертикальної інтеграції з глобальною мережею Інтернет. Сам термін «Інтернет речей» був вперше введений у науковий обіг Кевіном Ештоном у 1999 році в контексті оптимізації управління ланцюгами постачання з використанням радіочастотної RFID-ідентифікації. Це ознаменувало перехід до парадигми, де кожен фізичний об'єкт отримує свій цифровий ідентифікатор та IP-адресу, стаючи повноцінним вузлом мережі. Перехід до парадигми IoT став можливим завдяки конвергенції кількох технологічних трендів: масового впровадження протоколу IPv6, що вирішило проблему дефіциту адресного простору для мільярдів пристроїв; мініатюризації та здешевленню мікроелектроніки MEMS-сенсорів; та розвитку технологій хмарних обчислень, що надали необмежені ресурси для обробки великих даних Big Data [4].

Функціонування сучасної екосистеми IoT базується на чотирьох фундаментальних принципах. По-перше, це підключення, що забезпечує надійний канал передачі даних незалежно від фізичного середовища, дротового чи бездротового. По-друге, автоматизований збір даних через гетерогенні сенсорні мережі для оцифрування фізичних параметрів. По-третє, інтелектуальна аналітика, яка перетворює масиви «сирих» даних на корисну інформацію за допомогою алгоритмів машинного навчання та Big Data. Четвертим принципом є автоматизація – здатність системи виконувати дії (актуацію) на основі алгоритмічних рішень, замикаючи кібернетичний контур керування.

Статистичні дані свідчать про експоненціальне зростання цього сегменту ринку. За прогнозами аналітичного агентства Statista [5], кількість підключених IoT-пристроїв у світі зросте з 15 мільярдів у 2023 році до більш ніж 40 мільярдів у 2034 році. В той же час інші джерела, такі як IDC, прогнозують ще більш

агресивний ріст, вказуючи на потенціал досягнення позначки у 50 мільярдів пристроїв за рахунок масового впровадження 5G та супутникового інтернету, наприклад, найвідомішого Starlink. Така динаміка створює значне навантаження на мережеву інфраструктуру і вимагає перегляду традиційних централізованих підходів до обробки даних.

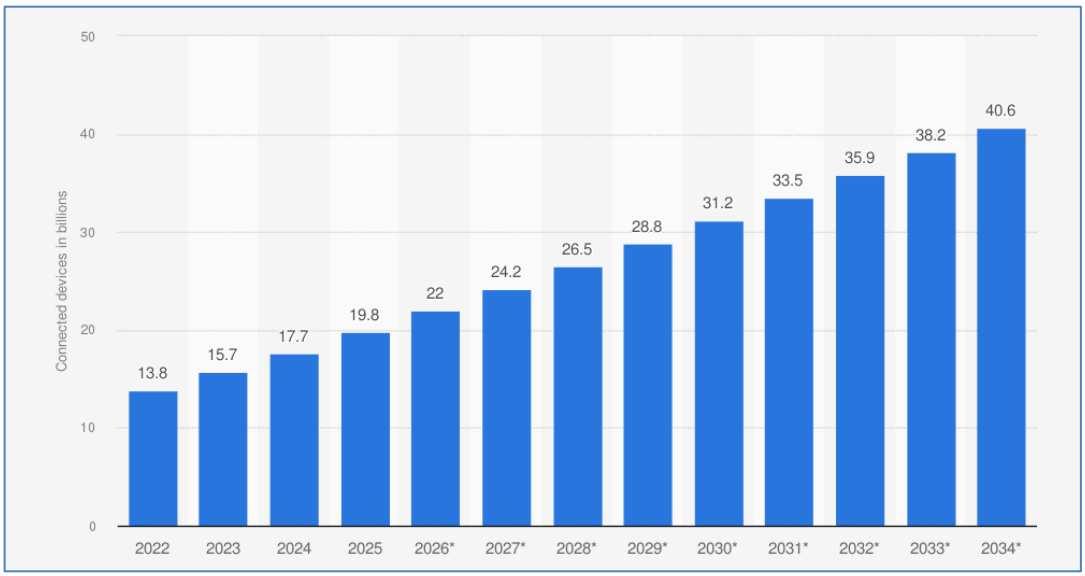


Рисунок 1.1 – Статистика зростання кількості IoT пристроїв

Сферу застосування IoT зазвичай можна класифікувати на три великі групи:

- Промисловий Інтернет речей Industrial IoT фокусується на автоматизації виробництва та надійності критичних систем.
- Споживчий Інтернет речей Consumer IoT спрямований на підвищення комфорту користувача через екосистеми «розумного будинку», носимі пристрої (wearables) та підключені побутові прилади.
- Розумні міста Smart Cities, де технології використовуються для оптимізації управління муніципальною інфраструктурою, включаючи транспортні потоки, енергомережі та екологічний моніторинг.
- Smart Home стосується автоматизації житлового простору, наприклад, клімат-контроль, освітлення, безпека, управління енергоспоживанням.

Ключові поняття IoT. Архітектура IoT систем.

Проєктування ефективних IoT-рішень неможливе без використання стандартизованих архітектурних моделей, що визначають ієрархію компонентів та регламент їх взаємодії. Фундаментальною моделлю для розуміння базових принципів побудови таких систем є трирівнева архітектура.

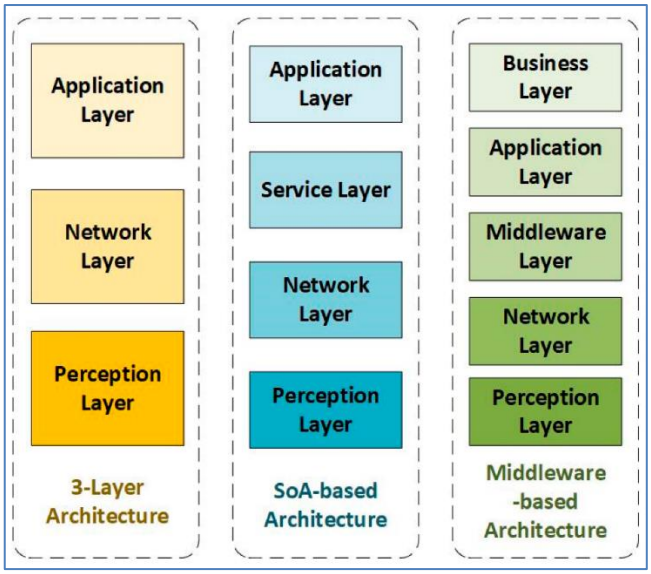


Рисунок 1.2 – Трирівнева архітектура IoT

Базовим рівнем цієї моделі є *Рівень сприйняття (Perception Layer)*, який виступає фізичним інтерфейсом системи з навколишнім середовищем. Він включає сенсори, що здійснюють перетворення фізичних величин в електричні сигнали, та актуатори, які реалізують зворотний керуючий вплив. Наступним є *Мережевий рівень (Network Layer)*, який забезпечує передачу даних від периферійних вузлів до центрів обробки, відповідаючи за маршрутизацію та комутацію пакетів. Верхнім рівнем є *Прикладний рівень (Application Layer)*, де здійснюється кінцева обробка інформації та візуалізація даних для користувача [6].

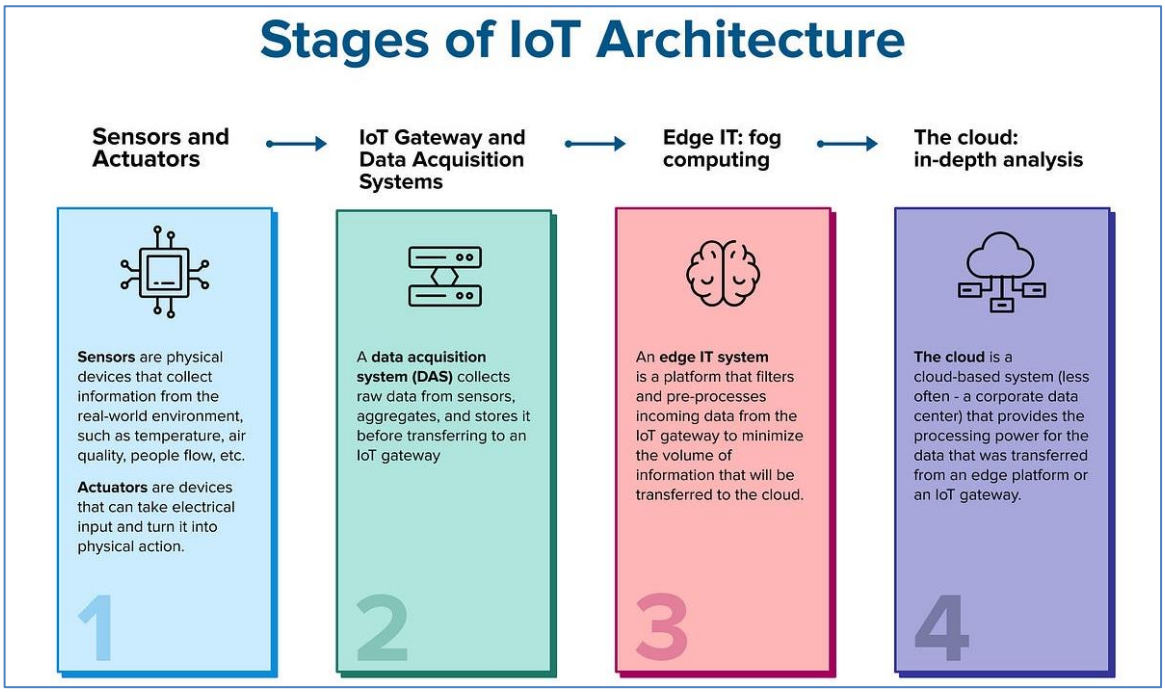


Рисунок 1.3 – Рівні архітектури IoT

Однак для складних, географічно розподілених систем трирівнева модель є недостатньою, тому в інженерній практиці застосовується розширена п'ятирівнева архітектура або сервіс-орієнтована архітектура SoA, що деталізує процеси обробки даних. Вона включає:

1) Бізнес-рівень, Business Layer. Це найвищий рівень абстракції, який керує всією системою IoT, включаючи бізнес-моделі, візуалізацію KPI (Key Performance Indicators) та стратегічне планування. Він відповідає за аналіз агрегованих даних для прийняття управлінських рішень та монетизацію послуг.

2) Прикладний рівень, Application Layer. Забезпечує безпосередню взаємодію з кінцевим користувачем або іншими системами. На цьому рівні реалізуються конкретні сценарії використання, наприклад, «розумний будинок», «розумне місто», надаючи інтерфейси для моніторингу та керування.

3) Рівень обробки, Service Support / Processing Laye або ж Middleware. Цей шар відповідає за зберігання SQL/NoSQL бази даних, автоматизований аналіз та обробку величезних масивів інформації. Тут працюють алгоритми хмарних

обчислень та штучного інтелекту для перетворення сирих даних на корисну інформацію.

4) Транспортний рівень, Network Layer. Відповідає за безпечну передачу даних від сенсорів до центрів обробки. Він забезпечує маршрутизацію та конвергенцію різних мережевих середовищ (бездротових сенсорних мереж, мобільного зв'язку 4G/5G, Інтернету).

5) Рівень сприйняття, Perception Layer. Базовий фізичний рівень, що складається з сенсорів збору даних та актуаторів виконання дій. Його основна функція – ідентифікація об'єктів та перетворення фізичних величин у цифрові сигнали.

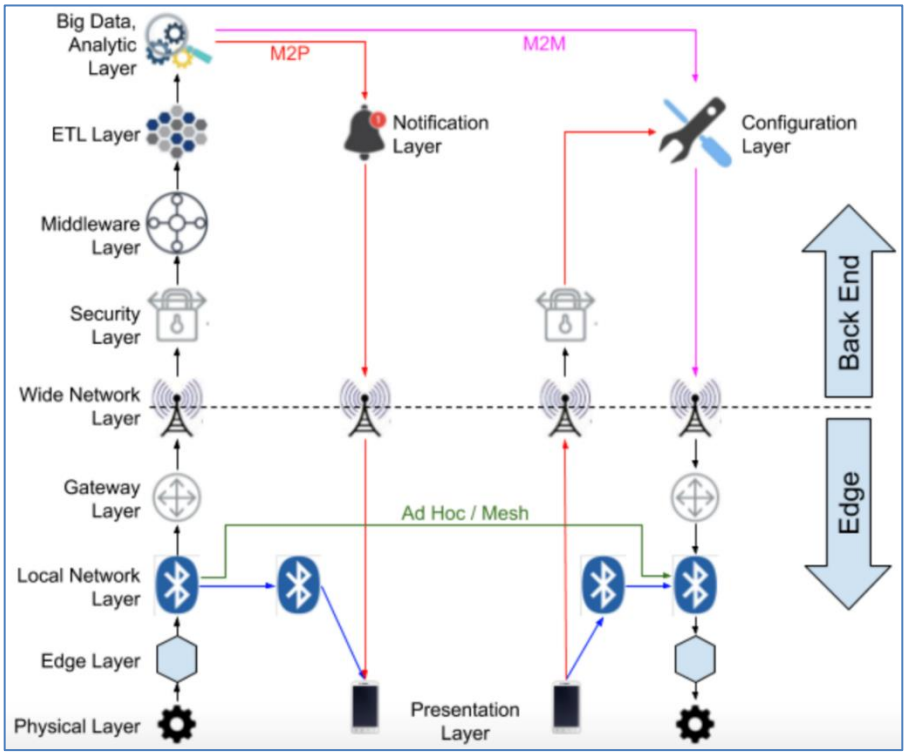


Рисунок 1.4 – Розширена п'ятирівнева архітектура

Інша поширена модель, орієнтована на обчислення, виділяє Edge Tier (кінцеві пристрої та датчики, що генерують дані), Fog Tier (локальні шлюзи та мережеве обладнання, що дозволяє фільтрувати інформацію, агрегувати показники та приймати миттєві рішення без звернення до хмари) та Cloud Tier (централізовані дата-центри для довгострокового зберігання великих даних, глибокої аналітики та

навчання моделей ШІ). Взаємодія компонентів усередині цих архітектур реалізується через специфічні патерни проектування [7].

Найпоширенішим патерном для домашніх систем є Thing – Gateway – Cloud. У цій топології ресурсно-обмежені пристрої, наприклад, датчики на протоколі Zigbee або Bluetooth LE, не мають прямого виходу в Інтернет. Вони підключаються до локального інтелектуального шлюзу Gateway, роль якого може виконувати Raspberry Pi. Шлюз агрегує дані, виконує трансляцію протоколів, наприклад, Zigbee в MQTT, шифрує трафік та передає його у хмару. Це дозволяє реалізувати локальну автоматизацію навіть при втраті зовнішнього з'єднання.

Важливим патерном є Cloud-in-the-Loop, де дані завжди проходять через хмарний сервіс для прийняття рішень. Це спрощує кінцеві пристрої, але створює критичну залежність від якості інтернет-з'єднання. Патерн Device-to-Device, D2D передбачає пряму взаємодію пристроїв, часто в mesh-мережах, минаючи центральний вузол. Концепція туманних обчислень Fog Computing передбачає розподілену архітектуру, де обчислення виконуються на рівні мережевого обладнання (роутерів, комутаторів), що знаходяться між кінцевими пристроями та хмарою.

Порівняння архітектурних підходів показує, що централізовані Cloud-centric системи забезпечують необмежену масштабованість та простоту обслуговування, проте мають єдину точку відмови та високу затримку. Натомість децентралізовані Edge-centric системи гарантують автономність, приватність та швидку реакцію, що є критичним для систем керування в реальному часі, таких як IP-KVM платформи.

Ключові компоненти IoT екосистеми.

Апаратна реалізація IoT-вузлів базується на взаємодії чотирьох категорій компонентів:

1) Сенсори (Sensors) є первинними перетворювачами фізичних величин:

- Температурні сенсори, а саме: цифрові датчики DS18B20 з точністю ±0.5°C й інтерфейсом 1-Wire є стандартом для моніторингу обладнання завдяки можливості побудови шинної топології.
- Сенсори середовища, а саме: датчики вологості, тиску BME280, освітленості BH1750.
- Сенсори безпеки, а саме: PIR-датчики руху, ультразвукові далекоміри HC-SR04.

2) Актуатори (Actuators) є виконавчими механізмами:

- Електромеханічні реле та MOSFET-транзистори для комутації навантаження (ATX Power/Reset).
- Сервоприводи для механічних дій.
- Засоби взаємодії між людиною і машиною HMI, світлодіоди, TFT/OLED дисплеї.

3) Контролери, що функціонують обчислювальними ядрами:

- Мікроконтролери MCU це енергоефективні чіпи (ESP32, STM32) для простих задач збору даних.
- Мікрокомп'ютери SBC це одноплатні комп'ютери (Raspberry Pi 4/5, Orange Pi), що працюють під управлінням повноцінних ОС Linux. Вони необхідні для реалізації веб-серверів, обробки відео та баз даних.

4) Комунікаційні модулі:

- Бездротові Wi-Fi 5/6 (широка смуга пропускання), Bluetooth LE (низьке споживання), Zigbee/Thread (mesh-мережі), LoRaWAN (далекий радіус).
- Дротові Gigabit Ethernet, RS-485, CAN, або навіть оптоволокно для гарантованої якості обслуговування QoS.



Рисунок 1.5 – Типи сенсорів IoT

Протоколи комунікації в IoT.

Для забезпечення продуктивності, надійності та енергоефективності IoT-системи вкрай необхідним стає правильний вибір стеку протоколів, які умовно можна класифікувати за рівнем моделі OSI на транспортні «прикладний рівень» та фізичні «апаратний рівень».

Транспортні протоколи:

- Протокол HTTP/HTTPS (REST API) базується на моделі «запит-відповідь», є універсальним стандартом для інтеграції веб-інтерфейсів. Він забезпечує високу сумісність та простоту використання, але має значні накладні витрати (overhead costs) через текстовий формат заголовків та необхідність встановлення нового TCP-з'єднання для кожного запиту, що робить його малоефективним для частих оновлень телеметрії в реальному часі.
- MQTT (Message Queuing Telemetry Transport) це легковагий бінарний протокол, що працює за архітектурою «видавник-підписник» Pub/Sub через центральний брокер. Він оптимізований для нестабільних мереж з обмеженою пропускною здатністю. Ключовою перевагою є підтримка різних рівнів якості

обслуговування, що дозволяє гарантувати доставку повідомлень, а також механізм LWT (Last Will and Testament) для відстеження стану підключення пристроїв.

- Спеціалізований протокол CoAP (Constrained Application Protocol) призначений для ресурсно обмежених пристроїв, що працює поверх UDP. Він підтримує асинхронний обмін повідомленнями та механізм спостереження за ресурсами, мінімізуючи навантаження на мережу та процесор.

- WebSocket це протокол, що забезпечує повнодуплексний канал зв'язку поверх одного TCP-з'єднання. Він критично важливий для інтерактивних застосунків, таких як KVM-системи, дозволяючи передавати команди керування миші/клавіатури та потокове відео в реальному часі з мінімальною затримкою, оминаючи обмеження класичного HTTP.

Таблиця 1.1

Порівняльна таблиця прикладних протоколів.

Протокол	Модель	Передача даних	Overhead costs	Real-time	Застосування
HTTP	Req/Res	TCP	Високі	Ні	Web API, конфігурація
MQTT	Pub/Sub	TCP	Низькі	Так	Телеметрія, IoT події
CoAP	Req/Res	UDP	Дуже низькі	Частково	Сенсори з батарейним живленням
WebSocket	Full Duplex	TCP	Середні	Так	Стрімінг, KVM керування

Протоколи фізичного рівня для Inter-chip communication:

- 1-Wire це низькошвидкісний протокол, розроблений компанією Maxim Integrated. Його головна особливість – можливість передачі даних та живлення по одному сигнальному провіднику. Кожен пристрій має унікальну 64-бітну адресу, що дозволяє підключати безліч датчиків (наприклад, DS18B20) до одного піна мікроконтролера.

- Синхронна шина I2C, Inter-Integrated Circuit створена для підключення низькошвидкісної периферії на коротких відстанях. Використовує дві лінії (SDA для даних та SCL для синхронізації) та підтримує адресацію до 127 пристроїв. Це стандарт для підключення датчиків середовища, акселерометрів та простих цифро-буквенних дисплеїв.

- Високошвидкісна повнодуплексна синхронна шина SPI (Serial Peripheral Interface) використовує чотири лінії: MOSI, MISO, SCK, CS. Вона забезпечує значно вищу пропускну здатність порівняно з I2C, що робить її ідеальною для підключення LCD-дисплеїв, SD-карт пам'яті та Ethernet-контролерів.

- UART (Universal Asynchronous Receiver-Transmitter) – це синхронний послідовний інтерфейс «точка-точка», що не вимагає лінії синхронізації. Широко використовується для налагодження пристроїв через системну консоль, підключення GPS-модулів та GSM-модемів.

Таблиця 1.2

Порівняльна таблиця протоколів фізичного рівня.

Протокол	Архітектура	Кількість ліній	Швидкість	Відстань	Застосування
1-Wire	Master-Slave	1 (Data/Power)	Низька (16 kbps)	До 100 м	Температурні датчики, iButton
I2C	Multi-Master	2 (SDA, SCL)	До 3.4 Mbps	< 1 м (на платі)	Сенсори, RTC, EEPROM
SPI	Master-Slave	4 (MOSI, MISO, SCK, CS)	Висока (>10 Mbps)	< 1 м (на платі)	LCD-дисплеї, SD-карти
UART	Peer-to-Peer	2 (TX, RX)	Змінна (до 1 Mbps)	До 15 м	GPS, GSM, Debug Console

1.2 Принципи та схеми побудови систем моніторингу, віддаленого керування та автоматизації.

Сучасні системи моніторингу, віддаленого керування та автоматизації базуються на багаторівневій архітектурі, що забезпечує ефективну взаємодію між фізичними пристроями, мережами передачі даних та інтелектуальними системами прийняття рішень.

Історія та класифікація систем віддаленого керування.

Еволюційний шлях засобів дистанційного керування можна умовно розділити на кілька етапів, кожен з яких якісно змінював підходи до архітектури та функціональних можливостей систем [8].



Рисунок 1.6 Частини сучасної промислової системи віддаленого керування

Розвиток систем віддаленого керування розпочався в другій половині XX століття з появою перших промислових систем автоматизації. Початкові механічні перемикачі для керування декількома комп'ютерами з'явилися у 1980-х роках та використовували багатополюсні обертові перемикачі для з'єднання однієї клавіатури та монітора з кількома комп'ютерами. Ці ранні пристрої не використовували жодної електроніки, а працювали шляхом фізичного перенаправлення електричних сигналів від портів клавіатури, відео та миші. Однак

механічні перемикачі мали суттєві обмеження: комп'ютери, не обрані перемикачем, не могли виявити клавіатуру або мишу, що часто унеможливлювало виявлення проблеми при завантаженні або змушувало комп'ютер знижувати роздільну здатність відео, а також вони мали проблеми зі стабільністю з’єднання.

У 1990-х роках виробники почали замінювати механічні перемикачі електронними схемами, що відкрило нову еру цифрових систем віддаленого керування. Ці нові цифрові пристрої могли емулювати присутність клавіатури та миші для всіх підключених комп'ютерів, вирішуючи проблему неможливості завантаження або втрати налаштувань відео. Термін «KVM switch» (Keyboard, Video, Mouse) був вперше введений у 1995 році Ремігіусом Шатасом, засновником компанії Cybex, що показало зростаючу важливість миші в комп'ютерних системах. Впровадження технології гарячих клавіш та екранного меню OSD значно спростило перемикання між комп'ютерами та зробило KVM-перемикачі більш популярними в офісах і центрах обробки даних.

На початку 2000-х років KVM-перемикачі почали підтримувати USB-з'єднання та вищі роздільні здатності відео. Розробка технології KVM-over-IP дозволила користувачам керувати комп'ютерами віддалено через мережу. Системи на основі IP надали безпрецедентну гнучкість для віддаленого доступу та керування, усуваючи фізичні обмеження традиційних KVM-перемикачів.

З появою мікропроцесорів відбулася цифрова революція переходу до систем диспетчерського управління та збору даних SCADA. Це дозволило не лише керувати процесами, а й накопичувати історичні дані для аналізу трендів, прогнозування аварій та оптимізації режимів роботи.

Сучасний етап Індустрії 4.0 характеризується переходом до парадигми Інтернету речей, де кожен пристрій стає автономним інтелектуальним агентом мережі, здатним до самодіагностики та M2M взаємодії без участі людини [9, 10].

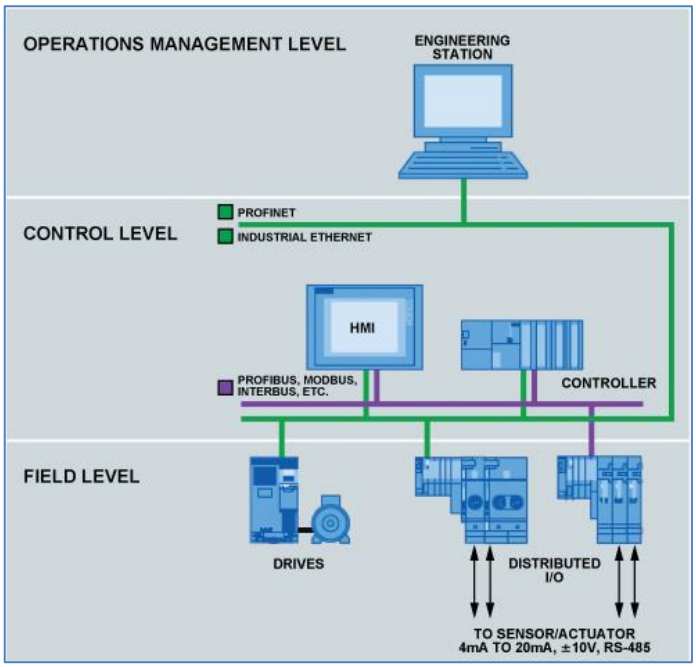


Рисунок 1.7 – Мережевий рівень промислової системи віддаленого керування

Класифікація сучасних систем віддаленого керування охоплює декілька основних категорій. Програмовані логічні контролери PLC виконують локальне керування дискретними та деякими аналоговими процесами [11]. PLC є захищеними комп'ютерами, розробленими для роботи в умовах електромагнітного шуму, вібрації та екстремальних температур. Вони виконують драбинну логіку, а дедалі частіше – функціональні блоки або структурований текст, причому рішення мають бути детермінованими та відбуватися в строгих часових межах.

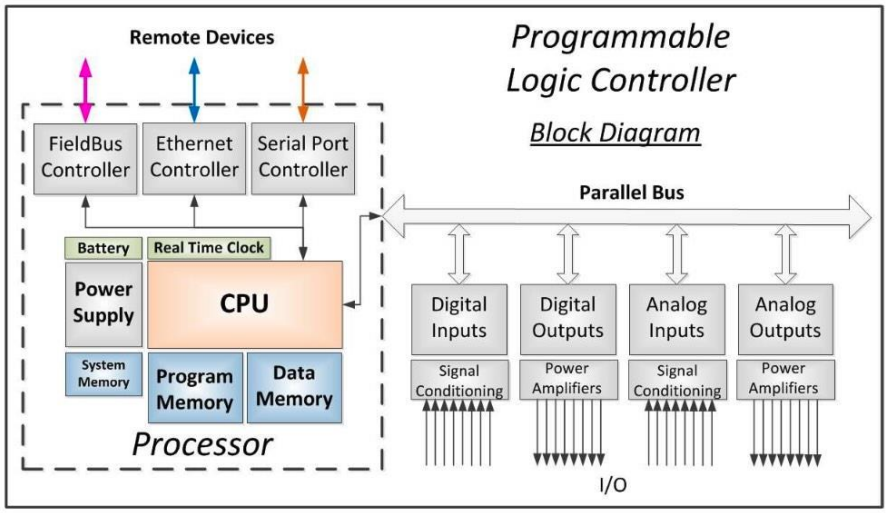


Рисунок 1.8 – Блок-схема PLC

Розподілені системи керування DCS знаходять застосування в управлінні та контролі складних промислових процесів, часто в локалізованій області [12]. DCS складаються з розподілених контролерів, об'єднаних у мережу по всьому заводу, з централізованими інтерфейсами для операторів. На відміну від SCADA, DCS орієнтовані на децентралізоване керування взаємопов'язаними процесами з високим рівнем автоматизації та взаємодії в реальному часі. DCS особливо корисні для безперервних процесів, таких як хімічне виробництво та фармацевтична промисловість.

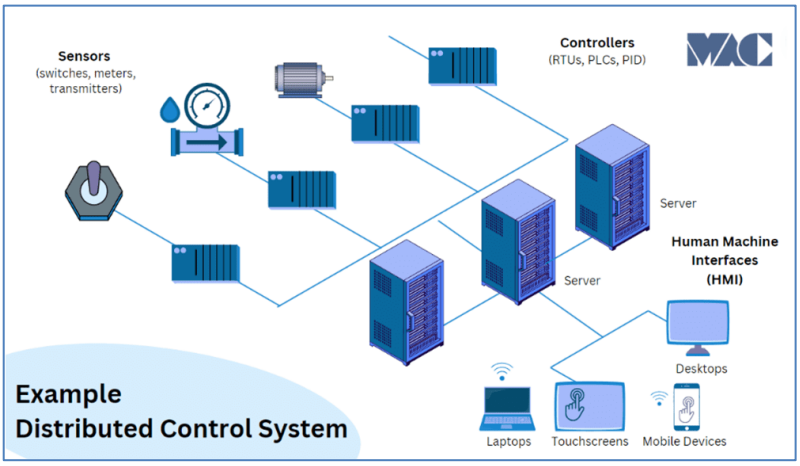


Рисунок 1.9 – Схема DCS

Системи диспетчерського контролю та збору даних SCADA використовуються для моніторингу та керування географічно розподіленими активами на великих відстанях [13]. SCADA-системи забезпечують централізований моніторинг та керування великомасштабними промисловими процесами, зазвичай з обмеженою взаємодією в реальному часі. Системи SCADA складаються з головної станції, віддалених термінальних блоків RTU, мереж зв'язку та інтерфейсів людина-машина HMI. Застосування SCADA охоплює нафтогазову промисловість, очищення води, електроенергетику та транспортні системи.

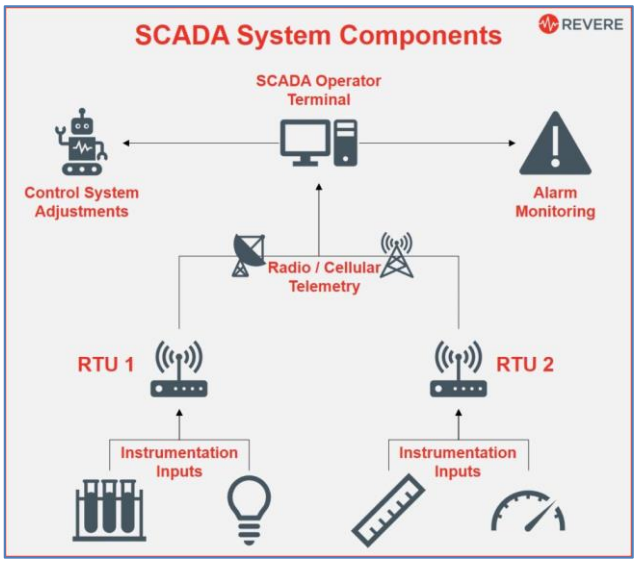


Рисунок 1.10 – Схема SCADA

Ще одну класифікацію сучасних систем віддаленого керування слід провести за рівнем взаємодії з об'єктом, що визначає їхню надійність та сферу застосування:

- Внутрішньосмугове керування, In-band, передбачає взаємодію з керованим вузлом через його основні мережеві інтерфейси LAN/WAN та засоби операційної системи. Це найпоширеніший та найдешевший метод, який не вимагає додаткового обладнання. Однак він має суттєвий архітектурний недолік: його працездатність повністю залежить від стану програмного забезпечення вузла. У разі критичного збою ОС, наприклад, «синій екран», помилки конфігурації фаєрволу або відмови мережевого адаптера, контроль над об'єктом втрачається безповоротно.

- Позасмугове керування, Out-of-band, використовує фізично або логічно незалежний канал керування, відокремлений від основної мережі передачі даних та операційної системи. Такий підхід, реалізований через окремі мережеві порти, гарантує повний доступ до обладнання навіть у критичних ситуаціях: зависання ядра ОС, відсутність завантажувального диска, збій BIOS/UEFI або вимкнене живлення. Це робить Out-of-band стандартом де-факто для серверних кімнат, центрів обробки даних ЦОД та критичної інфраструктури, дозволяючи виконувати «холодне» перезавантаження та діагностику на рівні заліза bare-metal.

Програмні системи віддаленого доступу.

У сегменті *In-band* рішень домінують програмні комплекси, які функціонують на прикладному рівні моделі OSI. Їхня робота базується на перехопленні потоків введення (клавіатура, миша) та виведення (відеоінтерфейс, консоль) безпосередньо в середовищі операційної системи та їх трансляції через мережу [14].

Для системних адміністраторів Unix-подібних систем та активного мережевого обладнання золотим стандартом залишається протокол SSH (Secure Shell). Він забезпечує криптографічно захищений тунель для передачі текстових команд, що дозволяє ефективно керувати системою навіть при низькій пропускній здатності каналу або високих затримках. Окрім того, SSH часто використовується як транспортний рівень для сучасних систем автоматизації конфігурацій Infrastructure as Code, таких як Ansible, дозволяючи масово керувати тисячами серверів одночасно.

Популярний для Windows систем RDP (Remote Desktop Protocol) це пропрієтарна розробка Microsoft, яка базується на семантичному підході. Замість передачі готового зображення екрану, RDP передає інструкції для його відмальовування на стороні клієнта – графічні примітиви GDI/DirectX. Це забезпечує високу плавність роботи, чіткість шрифтів та низьке споживання трафіку, але робить протокол менш ефективним при передачі складного растрового контенту, як відео чи 3D-графіка [15].

Також існує універсальний кросплатформенний протокол VNC/RFB (Remote FrameBuffer), що передає зміни пікселів екрану – растрові зображення. Сервер VNC аналізує фреймбуфер відеокарти, розбиває його на прямокутники та надсилає клієнту лише ті зони, що змінилися. Хоча VNC є сумісним з будь-якою ОС, він значно більш вимогливий до ширини каналу та чутливий до затримок мережі.

Комерційні хмарні P2P-сервіси, як TeamViewer чи AnyDesk, це рішення, які спеціалізуються на обході мережевих обмежень. Вони використовують проміжні

сервери для встановлення з'єднання, автоматизуючи проходження через NAT (Network Address Translation) та корпоративні фаєрволи, що робить їх незамінними для швидкої технічної підтримки кінцевих користувачів без необхідності складного налаштування мережі.

Апаратні системи віддаленого керування.

Апаратні засоби реалізують концепцію *Out-of-band* управління у найбільш повному обсязі, надаючи адміністратору рівень доступу, еквівалентний фізичній присутності біля пристрою. До цього класу насамперед належать вбудовані сервісні процесори BMC (Baseboard Management Controller), інтегровані в материнські плати серверного обладнання. BMC працює від резервного живлення материнської плати, функціонуючи постійно, незалежно від стану сервера. Контролер забезпечує позасмугове управління, моніторинг стану системи, управління живленням та доступ до віддаленої консолі.

Такі рішення, як iDRAC від Dell, iLO від HPE або загальний галузевий стандарт IPMI (Intelligent Platform Management Interface), функціонують незалежно від центрального процесора сервера. Вони мають власну мережеву карту, пам'ять та ОС реального часу, що дозволяє моніторити фізичні показники температури, напруги, оберти кулерів, переглядати журнали апаратних помилок, керувати живленням та монтувати віртуальні носії для встановлення ОС. Керування відновленням після збоїв здійснюється через локальні або віддалені перезавантаження системи та операції включення/вимикання живлення.

Одначе, коли мова йде про керування різнорідним обладнанням, яке не має вбудованих BMC, наприклад, застарілі сервери, промислові ПК, медичне обладнання, POS-термінали, банкомати, незамінними стають зовнішні KVM-рішення.

KVM-системи та IP-KVM.

Абревіатура KVM (Keyboard, Video, Mouse) позначає клас комутаційних пристроїв, що дозволяють керувати декількома комп'ютерами за допомогою

одного комплекту периферії [16]. Еволюція цих пристроїв від простих механічних перемикачів привела до появи IP-KVM, які не просто перемикають електричні сигнали, а оцифровують їх, інкапсулюють у TCP/IP пакети та передають локальною або глобальною мережею. Ключові переваги IP-KVM систем включають розподілену архітектуру без необхідності центрального комутатора, що усуває єдину точку відмови та підвищує надійність системи. Масштабованість досягається шляхом додавання нових вузлів без необхідності переконфігурації всієї системи.

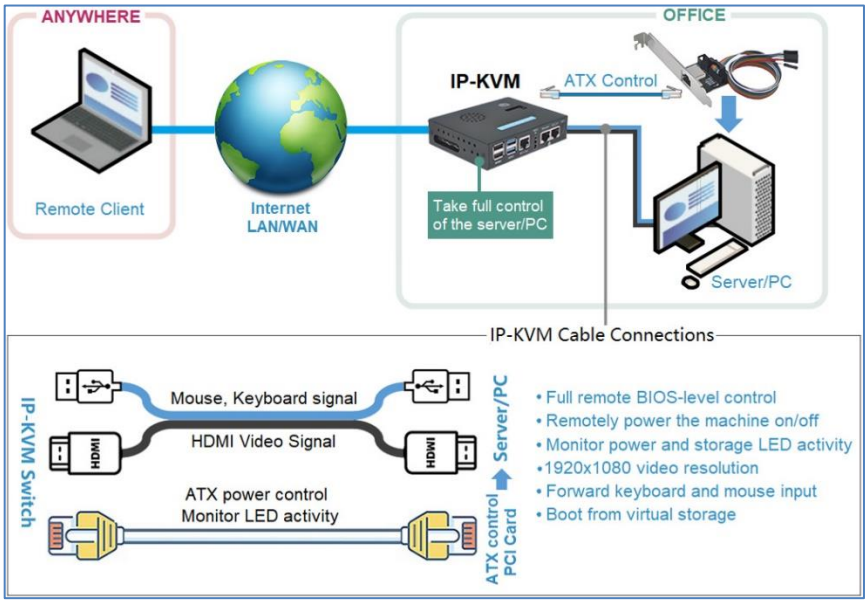


Рисунок 1.11 – Схема IP-KVM

Архітектура IP-KVM будується на принципі «людина посередині» (Man-in-the-Middle) на апаратному рівні. Пристрій фізично підключається до відеовиходу (HDMI, VGA, DVI) та портів введення (USB) цільового комп'ютера.



Рисунок 1.12 – Типовий сучасний IP-KVM свіч

Для комп'ютера IP-KVM виглядає як звичайний монітор та стандартна клавіатура/миша, що гарантує стовідсоткову сумісність з будь-якою операційною системою та BIOS/UEFI без встановлення драйверів [17]. Це дозволяє виконувати найскладніші низькорівневі операції: від конфігурації RAID-контролерів до повної реінсталяції операційної системи з нуля на віддаленому об'єкті.

Технології захоплення відео та USB-емуляція клавіатури/миші.

Серцем будь-якого IP-KVM є складні апаратно-програмні алгоритми обробки сигналів, які й будуть визначати якість користувацького досвіду та придатність системи для професійного використання.

Завдання підсистеми відео полягає у перетворенні високошвидкісного відеопотоку у компактний формат для передачі мережею з мінімальною затримкою. Наприклад, відеопотік 1080p при 60 Гц, що в залежності від бітрейту генерує гігабіти даних на секунду стає нерентабельним для передачі мережею. Підсистема відео складається з двох частин:

- *Апаратний інтерфейс*. Для первинного перетворення сигналу використовують спеціалізовані мости. Найвищу продуктивність забезпечують рішення на базі інтерфейсу CSI-2, наприклад, чіп Toshiba TC358743, які підключаються безпосередньо до шини процесора, мінімізуючи накладні витрати. Бюджетною альтернативою є USB-грабери на базі чіпів MacroSilicon MS21xx, Actions AM8352, де якість та затримка суттєво залежать від покоління шини. Так USB 2.0 вимагає агресивного стиснення на стороні чіпа і дозволяє якість відеопотоку лише 1080p30, тоді як USB 3.0 дозволяє передавати нестиснуте відео YUY2/NV12 в 1080p60.
- *Алгоритми компресії.* Золотим стандартом для KVM-систем залишається MJPEG. На відміну від потокових кодеків H.264/H.265, які використовують міжкадрове передбачення та субдискретизацію кольору, що призводить до розмиття дрібного тексту та артефактів на контрастних елементах інтерфейсу, MJPEG обробляє кожен кадр як окреме статичне зображення. Це гарантує читабельність термінальних шрифтів та коду, хоча і створює більше

навантаження на мережевий канал. У сучасних реалізаціях сумарна затримка не перевищує 100-150 мс, яку можна назвати порогом комфортної інтерактивної роботи.

Для передачі команд керування IP-KVM емулює пристрої класу HID, себто миша й клавіатура. Ключовим технічним викликом тут є синхронізація курсору миші. Традиційна відносна позиція Relative Mouse, яку використовують звичайні фізичні миші – передача зміщення dx/dy, часто призводить до розсинхронізації локального та віддаленого курсорів через різні налаштування прискорення в ОС. Тому сучасні IP-KVM використовують режим абсолютного позиціонування Absolute Mouse, емулюючи оцифровувач або сенсорний екран. Це дозволяє передавати точні координати X та Y, гарантуючи ідеальну точність наведення «піксель-в-піксель».

Окремої уваги заслуговує технологія Virtual Media, яка дозволяє IP-KVM емулювати USB-накопичувач. Це дає можливість адміністратору дистанційно монтувати ISO-образи дисків або локальні папки як віртуальні CD-ROM/Flash-накопичувачі на віддаленому сервері, що є необхідним для встановлення драйверів, оновлення прошивок або розгортання ПЗ на «голе залізо».

1.3 Raspberry Pi як платформа для IoT-проектів та розумного дому. Його характеристики, переваги та обмеження.

За неперервного інтенсивного розвитку технологій IoT та автоматизації житлових приміщень, обґрунтований вибір апаратної платформи для реалізації центрального керуючого вузла є визначальним етапом проектування системи. Одноплатні комп'ютери SBC (Single Board Computers) набули статусу де-факто стандарту для впровадження парадигми граничних обчислень Edge Computing у системах «розумного будинку». Дана стратегія передбачає децентралізацію обчислювальних потужностей шляхом їх перенесення безпосередньо до джерел

генерації даних, що дозволяє мінімізувати затримки реакції системи та знизити залежність від стабільності з'єднання з глобальною мережею. Серед наявного спектру апаратних рішень лідируючі позиції займає сімейство мікрокомп'ютерів Raspberry Pi [18], яке пройшло еволюційний шлях від спеціалізованого освітнього інструменту до потужної платформи для індустріальних та побутових застосувань.

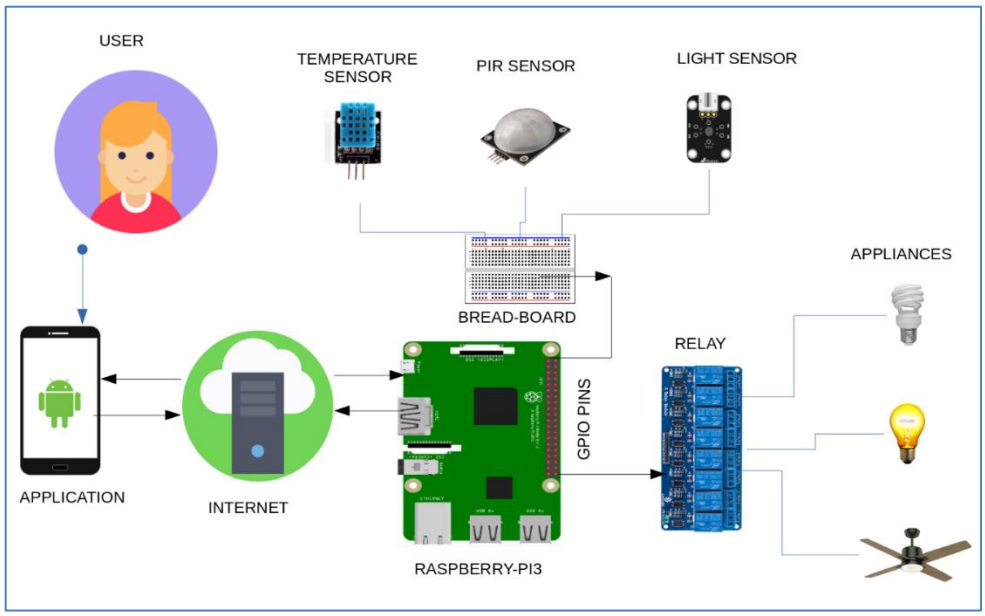


Рисунок 1.13 – Схема розумного дому на базі Raspberry Pi

Широке розповсюдження даної платформи зумовлене не лише її високими технічними характеристиками, але й наявністю масштабної спільноти розробників та розвиненої екосистеми сумісних модулів розширення HATs.

Огляд модельного ряду Raspberry Pi.

Номенклатура виробів Raspberry Pi охоплює широкий спектр моделей, адаптованих під різні сценарії експлуатації. Модельний ряд розпочинається з ультракомпактних рішень Raspberry Pi Zero та Zero 2 W, які, завдяки низькому енергоспоживанню та мініатюрним габаритам, є оптимальними для вбудованих автономних сенсорів та портативних пристроїв. Найбільш продуктивними представниками сімейства є моделі Raspberry Pi 4 та 5 на SoC Broadcom BCM2711 з чотирма ядрами Cortex-A72 та BCM2712 з чотирма ядрами Cortex-A76 відповідно,

архітектура та швидкодія яких дозволяють їм конкурувати з десктопними бюджетними ПК.

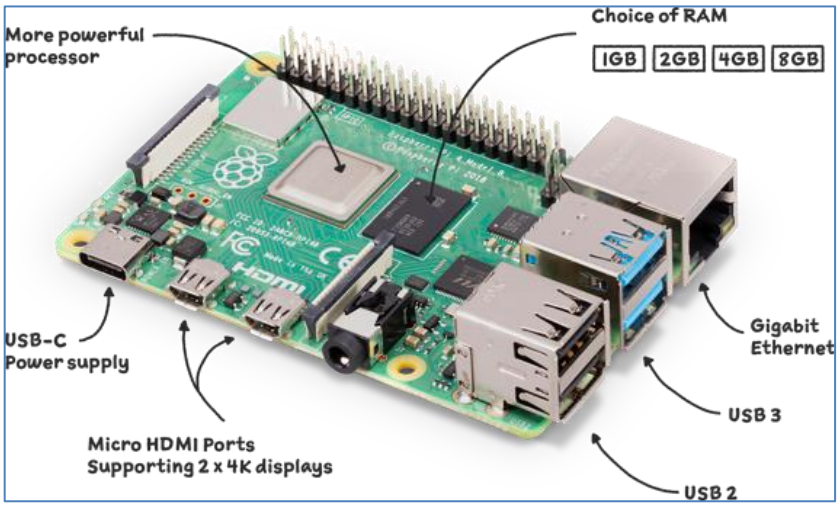


Рисунок 1.14 – Модулі Raspberry Pi 4B

Така продуктивність дозволяє виконувати локальну обробку даних, включаючи роботу з базами даних часових рядів, наприклад, InfluxDB та візуалізацію аналітики, наприклад, Grafana, без необхідності передачі чутливої інформації у хмарні сховища. Окремої уваги заслуговує Compute Module – індустріальна модифікація плати, позбавлена стандартних інтерфейсних роз'ємів та призначена для інтеграції у власні друковані плати кінцевих пристроїв. Розвиток платформи характеризується стійкою тенденцією до зростання обчислювальної потужності при збереженні архітектурної спадкоємності периферії та програмного забезпечення. Усе це надає можливість масштабування IoT-рішень: від простих сенсорних вузлів на базі Zero W до комплексних шлюзів обробки відеоданих та серверів домашньої автоматизації на базі старших моделей.

Таблиця 1.3

Технічні характеристики Raspberry Pi 4B

Параметр	Значення
Процесор (CPU)	Broadcom BCM2711, 4-ядерний ARM Cortex-A72 (ARMv8, 64-bit), тактова частота до 1.8 ГГц
Графіка (GPU)	Broadcom VideoCore VI, підтримка OpenGL ES 3.0

Продовження таблиці 1.3

Технічні характеристики Raspberry Pi 4B

Параметр	Значення
Оперативна пам’ять	4 ГБ LPDDR4 SDRAM
Бездротова мережа (Wi-Fi)	Двохдіапазонний 802.11 b/g/n/ac, 2.4 ГГц і 5 ГГц
Bluetooth	Bluetooth 5.0 з BLE
Проводова мережа (Ethernet)	Gigabit Ethernet через спеціальний контролер, підтримка PoE за допомогою окремого PoE HAT
Порти USB	2 × USB 3.0, 2 × USB 2.0
Відеовиходи	2 × micro-HDMI, підтримка двох дисплеїв до 4Kp60
Підтримка відео	Апаратне декодування H.265 (HEVC) до 4Kp60; H.264 до 1080p60 декодування, 1080p30 кодування
Аудіовихід	Комбінований 4-контактний 3.5 мм роз’єм (стерео аудіо + композитне відео)
Інтерфейс GPIO	40-контактний GPIO-роз’єм, 28 користувацьких GPIO, повна зворотна сумісність із попередніми моделями
Порти камери та дисплея	1 × 2-lane MIPI CSI (камера), 1 × 2-lane MIPI DSI (дисплей)
Носій даних	Слот microSD для ОС і зберігання даних
Живлення (вхід)	5 В DC через USB-C (рекомендовано ≥3 А); 5 В DC через GPIO; підтримка PoE через PoE HAT
Споживання струму	Типово до приблизно 3 А при повному навантаженні, залежить від периферії
Робоча температура	Від 0 до 50 °C
Габарити	Приблизно 85 × 56 × 17 мм

Інтерфейс введення-виведення загального призначення GPIO.

Визначальною архітектурною перевагою платформи, що відрізняє її від традиційних комп'ютерних систем та перетворює з простого комп'ютера на універсальну платформу для IoT-проектів, є наявність 40-контактного інтерфейсу введення-виведення загального призначення GPIO. Цей апаратний інтерфейс забезпечує безпосередню взаємодію програмного забезпечення з фізичними об'єктами шляхом низькорівневого керування електричними сигналами.

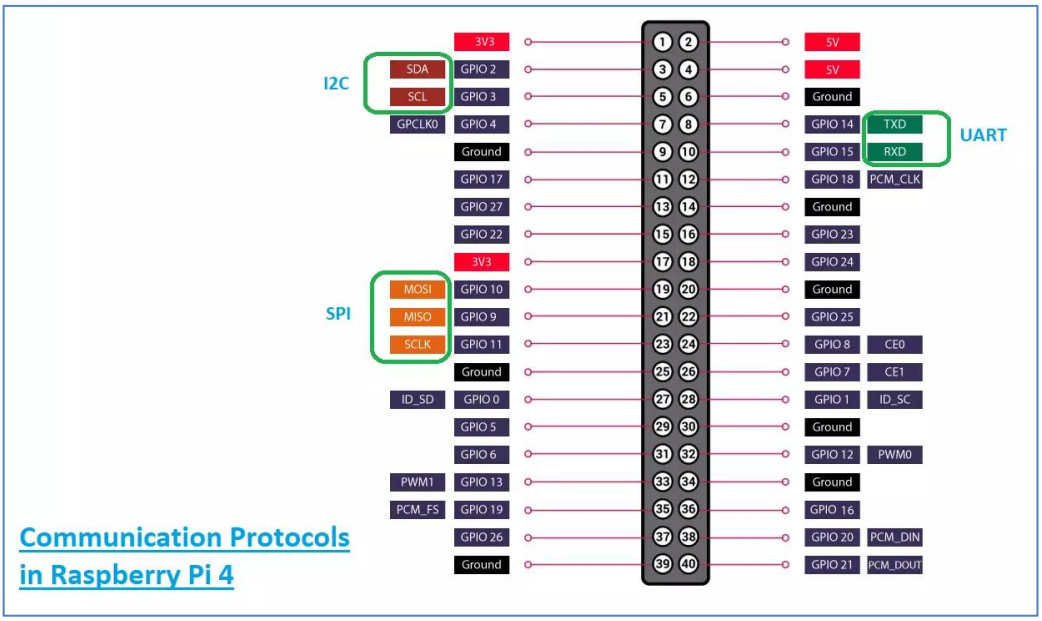


Рисунок 1.15 – Розпіновка контактів GPIO

Контакти GPIO підлягають динамічній конфігурації як на вхід для зчитування дискретних станів з датчиків, комутаційних елементів, інфрачервоних приймачів, так і на вихід для керування реле, світлодіодною індикацією, твердотільними перемикачами.

Важливою функціональною особливістю є апаратна підтримка широтно-імпульсної модуляції ШІМ/PWM на окремих контактах, що дозволяє здійснювати прецизійне керування потужністю навантаження, наприклад, регулювання яскравості освітлення або швидкості обертання електродвигунів, без створення додаткового навантаження на центральний процесор. Слід зауважити, що логічні рівні інтерфейсу GPIO становлять 3.3 В, що вимагає застосування перетворювачів

рівнів напруги level shifters при інтеграції з периферією, що працює за стандартом 5 В. Розуміння вимог до напруги є вкрай важливим для забезпечення сумісності та запобігання пошкодженню компонентів.

Вже готові бібліотеки Python для роботи з GPIO значно спрощують процес керування GPIO-пінами. GPIO Zero є зручною бібліотекою для керування GPIO-пінами Raspberry Pi, ідеальною для початківців у апаратному програмуванні. RPi.GPIO є більш просунутою бібліотекою, що пропонує детальний контроль над апаратними пінами Raspberry Pi. Модуль RPi.GPIO дозволяє легко керувати GPIO-пінами, забезпечуючи простий інтерфейс для конфігурації пінів як входів або виходів, читання їх стану та управління ними.

Інтерфейси передачі даних.

Raspberry Pi забезпечує апаратну підтримку стандартних промислових протоколів обміну даними, що є критично важливим для інтеграції широкого спектру цифрової периферії без необхідності розробки спеціалізованих драйверів:

- I2C, Inter-Integrated Circuit, є напівдуплексним протоколом послідовної передачі даних, призначеним для комунікації на коротких відстанях між платами, модулями та периферійними пристроями. Послідовна асиметрична шина, що дозволяє підключати значну кількість сенсорів (температури, тиску, вологості, акселерометрів) та дисплеїв, використовуючи лише дві сигнальні лінії (SDA та SCL). Адресація пристроїв дозволяє спростити схемотехніку підключення. I2C працює швидше, ніж UART, але не так швидко, як SPI.
- Інтерфейс SPI, Serial Peripheral Interface, забезпечує високошвидкісний повнодуплексний обмін даними. Його використовують для периферії, що вимагає високої пропускної здатності, зокрема графічних дисплеїв, швидкодіючих АЦП або RFID-зчитувачів.
- UART, Universal Asynchronous Receiver-Transmitter використовує один провід для однонаправленої передачі або два дроти для двонаправленої комунікації плюс заземлюючий провід. Хоча й швидкість передачі даних не є його сильною стороною, цей асинхронний інтерфейс призначений для послідовного зв'язку з

навігаційними модулями GPS, комунікаційними модемами GSM/GPRS, сканерами штрих-кодів або для низькорівневої діагностики системи через послідовну консоль.

- CSI (Camera Serial Interface) та DSI (Display Serial Interface) це два спеціалізовані високошвидкісні інтерфейси, підключені безпосередньо до графічної підсистеми GPU. Вони забезпечують пряму передачу даних з камер та на дисплеї, оминаючи шину USB та центральний процесор, що радикально знижує затримки та навантаження на CPU при обробці відеопотоку.

Операційне середовище та програмне забезпечення.

Основним програмним середовищем є Raspberry Pi OS (раніше Raspbian), що базується на дистрибутиві Debian Linux [19]. Вона гарантує гарну стабільність роботи, високий рівень безпеки та доступ до обширних репозиторіїв відкритого програмного забезпечення. Підтримка технологій контейнеризації, зокрема Docker, на сучасних ядрах Linux дозволяє ізолювати сервіси IoT та спростити процедуру їх оновлення та розгортання.

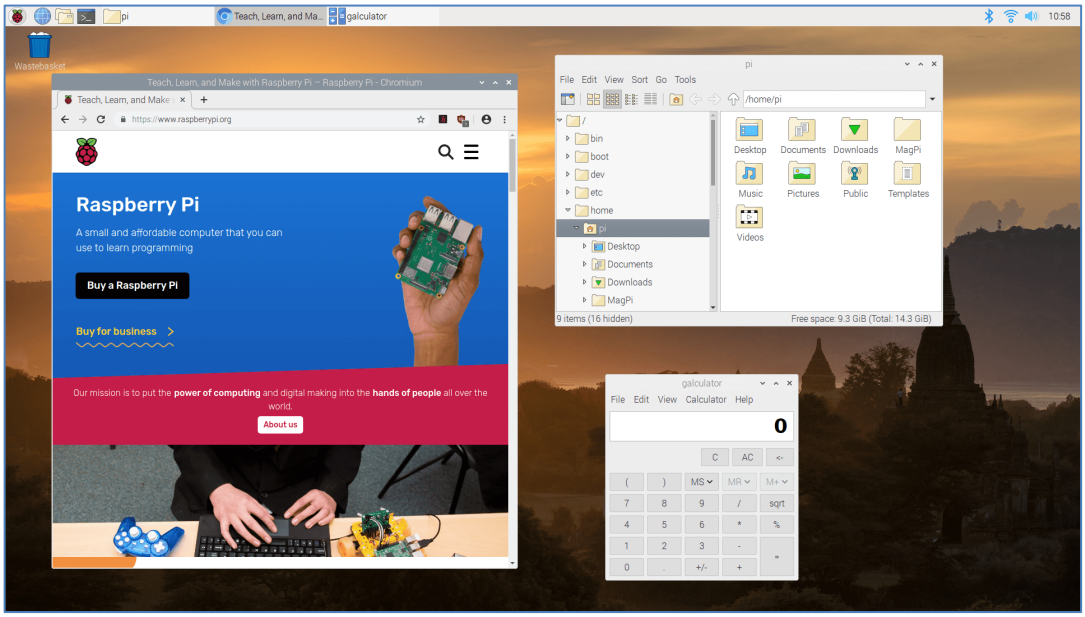


Рисунок 1.16 – Вигляд Raspberry Pi OS

Мова програмування Python набула статусу основного інструменту розробки для Raspberry Pi у всьому світі завдяки своїй синтаксичній лаконічності та

наявності потужних бібліотек для апаратної взаємодії. Бібліотеки, такі як RPi.GPIO та gpiozero, надають високорівневі об'єктно-орієнтовані абстракції, що дозволяє реалізовувати складні алгоритми керування з мінімальними витратами часу на написання коду [20]. Екосистема Python також включає інструменти для роботи з комунікаційними протоколами IoT, реалізація MQTT через paho-mqtt, створення легких веб-серверів для адміністративних панелей Flask, Django, FastAPI та аналізу сенсорних даних Pandas, NumPy.

Systemd як інструмент управління службами.

Для забезпечення надійності автономного функціонування IoT-пристроїв, які часто працюють у режимі безпосереднього нагляду оператора headless mode, критично важливим є використання системного менеджера ініціалізації systemd. Systemd включає набір інструментів для різноманітних завдань, але його основна мета – ініціалізувати, керувати та відстежувати системні сервіси та демони як під час запуску, так і під час роботи системи. На відміну від застарілих механізмів, таких як cron або rc.local, systemd дозволяє конфігурувати складні залежності запуску служб. Переваги використання Systemd для управління сервісами на Raspberry Pi включають: автоматичний запуск під час завантаження системи; управління залежностями для забезпечення правильного порядку запуску сервісів; автоматичний перезапуск у разі збою програми; контроль ресурсів, включаючи обмеження пам'яті та процесора; стандартизоване ведення журналу через journald; просте управління за допомогою команди systemctl.

Raspberry Pi вже став не просто іграшкою для ентузіастів, а потужним інструментом інженерії IoT. Найкращим застосуванням для нього є рівень верхньої ланки керування High-Level Control, агрегації даних та забезпечення інтерфейсу користувача, тоді як задачі низькорівневого керування апаратурою краще делегувати спеціалізованим мікроконтролерам.

1.4 Протоколи безпеки та віддаленого доступу.

Найважливішими аспектами у проектуванні систем віддаленого керування та IoT є забезпечення цілісності, конфіденційності та доступності інформації, тріада CIA, особливо коли йдеться про пристрої, що мають прямий доступ до апаратного забезпечення цільових серверів, зокрема IP-KVM. Вразливість каналів зв'язку може призвести до несанкціонованого перехоплення керування, витоку чутливих даних або використання пристрою як плацдарму для атак на внутрішню мережу підприємства. У даному розділі розглянуто ключові протоколи, що забезпечують захищений віддалений доступ, шифрування трафіку та автентифікацію користувачів у розроблюваній системі.

Протокол SSH, Secure Shell. Основним інструментом для безпечного адміністративного доступу до операційної системи Raspberry Pi та тунелювання трафіку є протокол SSHv2. Цей протокол прикладного рівня забезпечує криптографічний захист термінальних сесій, замінюючи незахищені протоколи Telnet та rlogin.

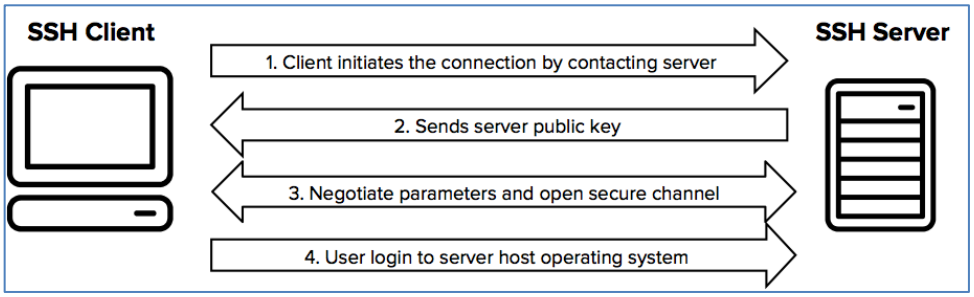


Рисунок 1.17 – Алгоритм роботи SSH

Архітектура SSH базується на клієнт-серверній моделі, де захищений канал формується після процедури взаємної автентифікації та узгодження параметрів шифрування, так званого handshake. Для забезпечення найвищого рівня безпеки в системі рекомендується відмова від парольної автентифікації на користь асиметричної криптографії з використанням пар ключів. Найбільш сучасним та продуктивним алгоритмом для генерації ключів на сьогодні є Ed25519, який забезпечує вищу криптостійкість та швидкість роботи порівняно з традиційним

RSA, що важливо для обмежених ресурсів одноплатних комп'ютерів. Крім того, SSH забезпечує можливість перенаправлення портів port forwarding, що дозволяє створювати зашифровані тунелі для доступу до внутрішніх сервісів, які не мають власного захисту.

Протоколи захисту веб-інтерфейсу TLS 1.3 та HTTPS. Беручи до уваги взаємодію користувача з графічним інтерфейсом IP-KVM через веб-браузер, фундаментальним елементом безпеки є протокол TLS (Transport Layer Security). У розроблюваній мною системі доцільно використовувати актуальну версію TLS 1.3, затверджену IETF у RFC 8446.

TLS 1.3 має ряд переваг, порівняно з попередніми версіями. Зменшення кількості циклів обміну повідомленнями (1-RTT handshake) знижує затримки при первинному підключенні і пришвидшує саме з’єднання. Вилучення застарілих та вразливих криптографічних алгоритмів, таких як MD5, SHA-1, RC4, та обов'язкове використання досконалої прямої секретності PFS (Perfect Forward Secrecy), що гарантує неможливість розшифрування перехопленого трафіку навіть у разі компрометації приватного ключа сервера в майбутньому.

Використання HTTPS, HTTP over TLS, реалізує автентифікацію сервера за допомогою цифрових сертифікатів, наприклад, від Let's Encrypt, та захист даних від атак типу «людина посередині».

WebRTC для передачі потокового відео. Для реалізації функції передачі відео з екрану цільового ПК з мінімальною затримкою, low latency streaming, використовується технологія WebRTC (Web Real-Time Communication). Безпека WebRTC є вбудованою на рівні стандартів і базується на двох ключових протоколах:

- DTLS (Datagram Transport Layer Security) використовується для шифрування метаданих та сигналізації. Так як відеотрафік передається через протокол UDP (для швидкості), використання стандартного TLS неможливе. DTLS адаптує механізми безпеки TLS для дейтаграмних протоколів.

- SRTP (Secure Real-time Transport Protocol) відповідає безпосередньо за шифрування аудіо та відео. SRTP забезпечує конфіденційність, автентифікацію повідомлень та захист від атак повторного відтворення replay protection.

За допомогою WebRTC відеопотік з IP-KVM не може бути переглянутий третіми особами навіть при перехопленні пакетів у локальній мережі.

Організація захищеного каналу зв'язку VPN та WireGuard. Надання безпечного доступу до пристрою з глобальної мережі Інтернет без необхідності відкриття портів на зовнішньому маршрутизаторі (що створює значні ризики безпеки), найефективніше вирішує технологія VPN (Virtual Private Network).

Серед існуючих рішень, як OpenVPN, IPsec/IKEv2, найбільш перспективним для IoT-систем на базі Linux можна назвати протокол WireGuard. Це сучасний VPN-протокол, який працює на рівні ядра Linux (Layer 3 OSI), що забезпечує значно вищу пропускну здатність та менші затримки порівняно з OpenVPN, який працює у просторі користувача userspace. Завдяки кодовій базі WireGuard близько 4000 рядків коду, проти сотень тисяч у OpenVPN, мінімізується поверхня кібератаки та спрощується аудит безпеки. Протокол не відповідає на пакети від неавторизованих пірів, роблячи сервер VPN невидимим для сканерів мережі. Він підтримує зміну IP-адреси клієнта без розриву з'єднання, що ідеально підходить для мобільного керування системою зі смартфона.

Автентифікація та авторизація. На додаток до шифрування каналів, вкрай важливим є впровадження надійних механізмів автентифікації. Для доступу до веб-інтерфейсу IP-KVM необхідна реалізація двофакторної автентифікації 2FA. Стандартом де-факто тут є алгоритм TOTP (Time-based One-Time Password), описаний у RFC 6238.TOTP базується на генерації одноразових цифрових кодів за допомогою хеш-функції HMAC-SHA1, де вхідними параметрами є секретний ключ та поточний час UNIX-time. Це дозволяє використовувати апаратні токени або мобільні додатки Google Authenticator, Microsoft Authenticator як другий фактор, захищаючи систему від атак перебору паролів brute-force та крадіжки облікових даних.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ, АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ ТА ВИБІР КОМПОНЕНТІВ

2.1 Постановка задачі проєкту, його функціональні вимоги, модульність системи, обмеження, основні сценарії використання.

Розробка автономної платформи віддаленого моніторингу та керування робочим місцем на базі Raspberry Pi потребує вичерпного формулювання задачі проекту з урахуванням функціональних вимог, архітектурних принципів, технічних обмежень та сценаріїв використання. Перший крок до цього, це постановка задачі проекту, бо визначаються межі проекту, очікувані результати та критерії успіху.

Формулювання проблеми та контекст проекту.

В даний момент найпопулярніші системи віддаленого керування та моніторингу робочих станцій є програмними, а тому стикаються з кількома фундаментальними проблемами. Обмежений доступ на рівні BIOS/UEFI є критичною проблемою, оскільки програмні рішення віддаленого доступу, такі як RDP, VNC або TeamViewer, стають недоступними, якщо операційна система не завантажилася або вимагає втручання на етапі POST, Power-On Self-Test. Високу вартість комерційних рішень демонструють промислові KVM-over-IP системи від виробників, таких як ATEN, Raritan або Dell, які коштують від $500 до $3000 за одиницю, що робить їх недоступними для малого бізнесу, домашніх лабораторій та освітніх закладів [21]. Водночас комерційні системи для домашнього використання та малих офісів, на кшталт Pi-KVM, TinyPilot, NanoKVM мають строго обмежену функціональність, хоча і співставні за ціною з розроблюваною системою.



Рисунок 2.1 – Промисловий KVM-свіч компанії ATEN

Відсутність інтеграції з IoT-моніторингом у традиційних KVM-системах обмежує можливості проактивного управління, оскільки більшість комерційних рішень зосереджені виключно на функціях клавіатура-відео-миша, ігноруючи телеметрію навколишнього середовища, моніторинг температури та можливості автоматизації. Складність розгортання та обслуговування enterprise-рівня систем вимагає спеціалізованих знань та інфраструктури, що створює бар'єр для впровадження в середовищах з обмеженими ресурсами.

Контекст використання системи охоплює декілька сценаріїв, а саме: віддалене управління домашньою робочою станцією, за якою виконується деяка робота; повне перевстановлення операційної системи робочої станції; екстрені ситуації та відновлення після збоїв, коли необхідно швидко діагностувати та виправити критичні проблеми без фізичної присутності.

Функціональні вимоги системи.

Функціональні вимоги визначають задачі, які система повинна робити, представляючи собою конкретні сценарії поведінки системи.

Підсистема захоплення та передачі відео повинна забезпечувати наступні можливості: захоплення HDMI-сигналу з роздільною здатністю до 1920×1080 пікселів при частоті оновлення мінімум 30 кадрів на секунду для забезпечення плавного відображення інтерфейсу; підтримку кодеків H.264/H.265 для ефективної компресії відеопотоку з метою мінімізації пропускної здатності мережі з цільовим

бітрейтом 2-8 Мбіт/с; затримку відображення (end-to-end latency) не більше 200 мілісекунд для забезпечення комфортної інтерактивної роботи; можливість запису відеопотоку для аудиту та відтворення критичних операцій.

Підсистема емуляції клавіатури та миші має реалізувати: емуляцію USB HID для клавіатури з підтримкою всіх стандартних клавіш, включаючи функціональні та мультимедійні; емуляцію USB HID миші з абсолютним позиціюванням курсору для точного управління; затримку обробки вводу input lag не більше 50 мілісекунд від натискання клавіші до реєстрації події на цільовому комп'ютері; підтримку комбінацій клавіш, включаючи Ctrl+Alt+Del, для доступу до системних функцій.

Підсистема керування живленням повинна забезпечувати програмне вмикання та вимикання цільового комп'ютера через апаратну емуляцію натискання кнопки живлення з використанням оптронної розв'язки; програмну емуляцію короткого (1 секунда) та тривалого (5 секунд) утримання кнопки живлення для програмного вимкнення та примусового перезавантаження відповідно; скидання налаштувань BIOS/UEFI цільового комп'ютера у разі відсутності картинки від ОС.

Веб-інтерфейс користувача повинен надавати адаптивний інтерфейс, сумісний з настільними браузерами Chrome, Firefox, Edge та мобільними пристроями; час завантаження головної сторінки не більше 10 секунд при швидкості з'єднання 10 Мбіт/с; інтуїтивне розташування елементів керування з можливістю швидкого доступу до критичних функцій включення/вимикання, перезавантаження не більше ніж за 2 кліки.

Підсистема безпеки та автентифікації має реалізувати багатофакторну автентифікацію користувачів з підтримкою мінімум двох методів (пароль + одноразовий код); шифрування всіх мережевих з'єднань з використанням TLS 1.3; рольову модель доступу (RBAC) з мінімум трьома рівнями привілеїв: адміністратор, оператор, спостерігач.

Функціонал системи має включати підтримку віртуальних носіїв інформації Virtual Media, тобто можливість дистанційного монтування образів дисків формату

.iso через веб-інтерфейс. Пристрій повинен емулювати зовнішній USB-накопичувач, що дозволяє виконувати низькорівневі задачі: розгортання операційних систем з нуля, завантаження LiveCD для відновлення даних, оновлення прошивок BIOS або RAID-контролерів без фізичної присутності.

Модульність архітектури системи.

Архітектурне рішення проекту базується на принципах модульності та слабкої зв'язності компонентів. Такий підхід забезпечує гнучку масштабованість системи, спрощує процеси діагностики несправностей та дозволяє проводити модернізацію окремих вузлів без повної заміни пристрою. Декомпозиція системи здійснюється на двох рівнях: апаратному та програмному.

Структура апаратних модулів системи починається з модуля захоплення відео, який інкапсулює всю логіку роботи з HDMI-USB адаптером, включаючи ініціалізацію пристрою MJPEG, налаштування параметрів захоплення та обробку відеопотоку; модуль емуляції HID відповідає за створення віртуальних USB-пристроїв клавіатури та миші через kernel gadget API, забезпечуючи трансляцію команд з веб-інтерфейсу в події HID; модуль GPIO-керування надає уніфікований інтерфейс для роботи з пінами загального призначення та забезпечуючи функції керування оптронами та читання стану сигналів.

Веб-сервер та API модуль реалізує RESTful API для взаємодії клієнтського інтерфейсу з backend-сервісами, забезпечуючи кінцеві точки для керування системою, отримання телеметрії та управління відеопотоком. Модуль автентифікації та авторизації централізовано обробляє всі аспекти безпеки, включаючи валідацію облікових даних, управління сесіями та перевірку дозволів.

Модульність системи даного проекту означає можливість легкої модернізації. Переваги модульності очевидні: спрощене тестування коли кожен модуль може бути протестований ізольовано; полегшена паралельна розробка модулів; легка заміна компонентів, наприклад, можливість швидко перейти з одного HDMI-USB адаптера на інший, змінивши лише модуль захоплення

відео; масштабованість системи, що дозволяє додавати нові модулі, наприклад, підтримку додаткових датчиків або інтеграцію з системами розумного дому без переробки існуючого коду.

Економічні та фізичні обмеження.

Успішна реалізація проекту вимагає врахування низки жорстких обмежень, які визначають межі проектних рішень та вибір елементної бази.

Бюджетні обмеження проекту визначають доступні фінансові ресурси та впливають на вибір компонентів. Цільовий бюджет проекту встановлено на рівні $100-150 (приблизно 4000-6000 грн), що виступає ключовою вимогою для демонстрації економічної ефективності рішення порівняно з комерційними KVM-over-IP системами аналогічного класу вартістю $100-500. Задля збереження універсальності та модульності системи, порівняно з трохи дешевшими, але вузькоспеціалізованими аналогами, в нагоді стане використання усіх можливостей закупівлі компонентів. Саме тому доречно використати маркетплейс Aliexpress, оскільки майже всі компоненти родом з Китаю, а купувати напряму без посередників в Україні виходить в рази дешевше. Так наприклад, карта захоплення відео HDMI-to-USB в Україні коштує 400-600 грн., тоді як аналогічну на Aliexpress можна взяти за 200 грн. Недоліком такого способу придбання стає великий час очікування доставки в межах від двох до шести тижнів. Далі буде описана детальна структура бюджету.

Основна обчислювальна платформа: Raspberry Pi 4 Model B 4GB RAM – $55-65 (2200-3000 грн), що становить близько 45% від загального бюджету та є найдорожчим компонентом системи; карта пам’яті micro SD Kingston SDCG3 64 Gb – $8-15 (350-600 грн). Компоненти захоплення відео: HDMI-to-USB capture card на базі MS2109 чипсету – $5-15 (200-600 грн), що забезпечує захоплення відео до 1080p@30fps. Дисплей та інтерфейс: 3.5" TFT LCD дисплей 320×480 з резистивним тачскріном та GPIO-сплітером – $12-18 (480-720 грн). Компоненти GPIO та датчики: три одноканальні оптопари PC817 для ізоляції сигналів – $2-3 (80-120 грн); один датчик температури DS18B20 – $2-3 (80-120 грн). Кабелі та з'єднувачі:

HDMI кабель, USB кабелі, перемички Dupont для GPIO – $8-12 (300-500 грн). Живлення та аксесуари: USB-C блок живлення 5V/3A для Raspberry Pi – $5-10 (200-400 грн); корпус для Raspberry Pi з вентиляцією – $5 (200 грн).

Апаратні обмеження визначаються характеристиками обраних компонентів: обчислювальна потужність Raspberry Pi 4B обмежена чотирьохядерним процесором ARM Cortex-A72 на частоті 1.5 ГГц, що накладає обмеження на складність обробки відеопотоку в реальному часі та кількість одночасних користувацьких сесій (практична межа 2-3 одночасних з'єднання з прийнятною якістю); оперативна пам'ять 4 ГБ обмежує можливості кешування та одночасного запуску сервісів, вимагаючи оптимізації використання пам'яті; пропускна здатність USB 3.0 становить теоретичний максимум 5 Гбіт/с, але реальна продуктивність HDMI-USB адаптера обмежена 1080p@30fps або 720p@60fps через специфіку чіпсету MS2109.

GPIO піни Raspberry Pi обмежені 40-контактною гребінкою з лише 26 GPIO-пінами загального призначення, що вимагає ретельного планування розподілу пінів між дисплеєм, датчиками та оптронами. Швидкість зчитування 1-Wire протоколу для датчика DS18B20 обмежена мінімальним часом конвертації 750 мілісекунд при максимальній роздільності 12 біт, що визначає частоту оновлення температурних даних. Відеопам'ять GPU обмежена спільним розділом з системною RAM, зазвичай 64-128 МБ, що впливає на можливості апаратного кодування відео.

Енергетичні обмеження включають споживання електроенергії системою: типове споживання Raspberry Pi 4B під навантаженням становить 3-4 Вт без периферії та 6-8 Вт з підключеними USB-пристроями та активним дисплеєм; максимальна потужність джерела живлення 15 Вт визначає загальну межу споживання всієї системи, включаючи USB-периферію. Теплові обмеження вимагають забезпечення адекватного охолодження для запобігання тротлінгу процесора при температурі вище 80°C, що особливо критично при інтенсивному кодуванні відеопотоку.

Програмні обмеження включають: відсутність апаратної підтримки H.265 кодування в Raspberry Pi 4B, що обмежує вибір кодеків до H.264 або MJPEG; обмеження Raspberry Pi OS у відсутності real-time kernel за замовчуванням, що може призводити до коливань часу доставки пакетів jitter у критичних за часом операціях; ліміти на кількість одночасно відкритих файлових дескрипторів, звичайно 1024, що впливає на можливість обробки великої кількості одночасних мережевих з'єднань.

Сценарії експлуатації.

Моя система проектується як універсальний інструмент, здатний покрити широкий спектр задач в IT-інфраструктурі [22]. Нижче буде наведено детальний опис найвірогідніших сценаріїв використання.

Сценарій 1. Нормальна робота системи при стабільній роботі всіх компонентів. Діячі: користувач системи, адміністратор. Передумови: всі компоненти функціонують без помилок; цільовий комп'ютер успішно завантажився; користувач автентифікований. Алгоритм дій: користувач відкриває веб-інтерфейс та спостерігає головну панель керування зі статусом комп'ютера; система безперервно передає HDMI-відеопотік з затримкою менше 200 мс; користувач виконує звичайну роботу на цільовому комп'ютері через емульовану USB-клавіатуру та мишу; весь трафік зашифрований TLS 1.3; система записує журнали всіх дій для аудиту.

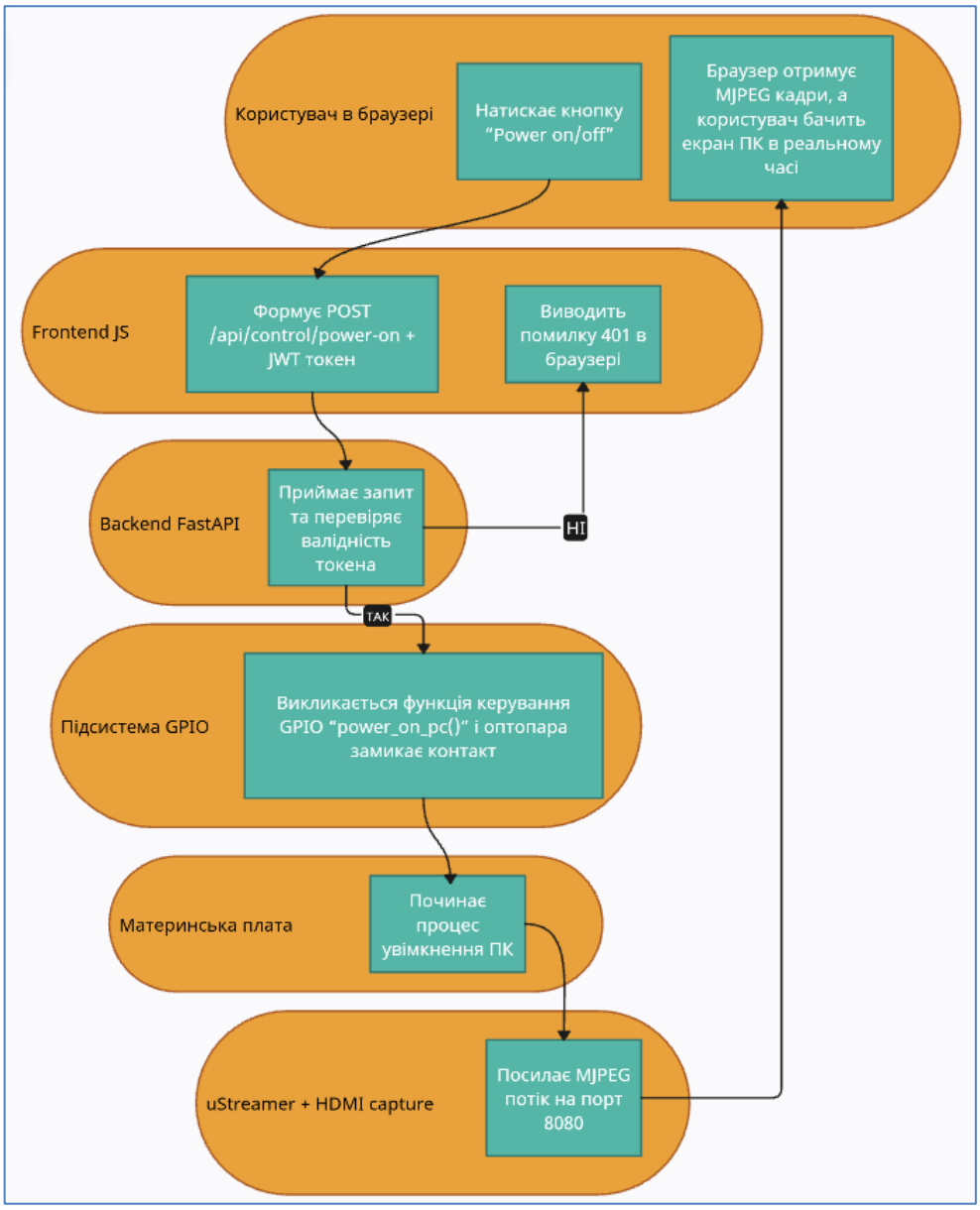


Рисунок 2.2 – Узагальнена блок-схема процесу віддаленого увімкнення ПК

Сценарій 2. Перевстановлення ОС Windows з Raspberry Pi як USB-накопичувача. Діячі: системний адміністратор. Передумови: Raspberry Pi налаштована в режимі USB OTG mass storage; на SD-карті завантажений ISO-образ операційної системи; цільовий комп'ютер підключений до Raspberry Pi через USB-C. Основний потік: адміністратор підключає Raspberry Pi через USB-C до цільового комп'ютера; цільовий комп'ютер розпізнає Raspberry Pi як USB mass storage device; адміністратор перезавантажує комп'ютер та вибирає Raspberry Pi як пристрій завантаження у BIOS; комп'ютер завантажується з ISO-образу на Raspberry Pi; адміністратор спостерігає процес через HDMI-відеопоток; проводить розбиття

диску та встановлення ОС за допомогою емульованої клавіатури; після завершення розмонтує Raspberry Pi та система повертається до звичайного режиму KVM.

Сценарій 3. Віддалене завантаження та налаштування BIOS/UEFI. Діячі: системний адміністратор, віддалений користувач. Передумови: цільовий комп'ютер вимкнений або перебуває в стані POST; користувач автентифікований в системі та має права оператора або адміністратора; мережеве з'єднання з Raspberry Pi стабільне. Алгоритм дій: користувач відкриває веб-інтерфейс системи в браузері та переходить до розділу керування живленням; користувач натискає віртуальну кнопку «Power On», система активує GPIO-пін, підключений до оптрона PC817, який замикає контакти кнопки живлення на материнській платі на 1 секунду; цільовий комп'ютер починає процес POST, система автоматично починає захоплення HDMI-сигналу та відображення його у веб-інтерфейсі; користувач спостерігає екран POST та натискає відповідну клавішу (наприклад, Del або F2) через емульовану USB-клавіатуру для входу в BIOS; система транслює натискання клавіші через USB HID, комп'ютер входить в меню BIOS/UEFI; користувач працює меню BIOS, використовуючи емульовану клавіатуру, змінює необхідні налаштування (порядок завантаження, параметри CPU, RAM, VT-x/AMD-V) та зберігає зміни; користувач вибирає опцію «Save and Exit», комп'ютер перезавантажується з новими налаштуваннями.

Сценарій 4. Діагностика та відновлення після критичного збою. Діячі: технічний спеціаліст, користувач. Передумови: цільовий комп'ютер завис або не відповідає на мережеві запити; програмні методи віддаленого доступу SSH, RDP недоступні. Алгоритм дій: система моніторингу виявляє, що цільовий комп'ютер не відповідає на ping-запити протягом 5 хвилин; він підключається до веб-інтерфейсу Raspberry Pi та спостерігає поточний стан екрану через відеопотік; користувач виявляє, що система завантажилася до «kernel panic» або «BSOD»; користувач виконує примусове вимкнення, утримуючи віртуальну кнопку живлення протягом 5 секунд або натискає на кнопку перезавантаження; після вимкнення він чекає 10 секунд та знову вмикає комп'ютер; він спостерігає процес завантаження через відеопотік, входить в BIOS для налаштування завантаження з

USB-накопичувача для відновлення або діагностики; після завершення діагностики та усунення проблеми користувач перезавантажує систему в нормальному режимі та перевіряє, що всі сервіси працюють коректно.

У цьому підрозділі було проведено постановку задачі проекту, визначено чіткі межі розробки, встановлено функціональні вимоги, обґрунтовано модульний підхід до архітектури системи та враховано реальні обмеження бюджету та обладнання. Основні сценарії використання демонструють практичну застосовність системи для вирішення конкретних задач віддаленого моніторингу та керування робочими станціями.

2.2 Методи віддаленого увімкнення ПК. Огляд існуючих систем віддаленого керування та моніторингу робочого місця. Порівняння створюваної системи з існуючими рішеннями.

Ринок апаратних засобів віддаленого адміністрування та керування серверною інфраструктурою тривалий час залишався консервативним та сфокусованим переважно на корпоративному Enterprise сегменті. У цій ніші домінуючі позиції історично займали високовартісні пропрієтарні рішення від таких виробників, як Aten, Raritan, Avocent та Dell iDRAC, вартість яких часто перевищувала бюджет обладнання, що обслуговується. Однак, глобальна трансформація робочих процесів у бік дистанційної взаємодії, децентралізація обчислювальних потужностей Edge Computing та стрімкий розвиток інфраструктури домашніх лабораторій збільшили попит на економічні, але функціональні пристрої класу SOHO (Small Office/Home Office). У даному підрозділі буде здійснено детальний аналіз існуючих методів дистанційного керування живленням обчислювальної техніки, проведено поглиблений огляд найбільш розповсюджених бюджетних апаратних рішень та виконано їх порівняльний аналіз із проектованою системою.

Методи дистанційного керування живленням персональних комп'ютерів.

Забезпечення повноцінного та надійного віддаленого доступу неможливе без наявності механізмів керування фізичним станом живлення. Це включає можливість виконання старту вимкненої системи та примусового апаратного перезавантаження у разі критичних програмних збоїв, наприклад, Kernel Panic або BSOD, коли стандартні програмні засоби керування стають недоступними. Далі буде наведено аналіз основних методів віддаленого увімкнення ПК.

Стандартизований мережевий протокол канального рівня *Wake-on-LAN (WoL)* дозволяє ініціювати увімкнення комп'ютера, який перебуває у режимі сну або гібернації, шляхом надсилання спеціалізованого широкомовного кадру даних, відомого як «Magic Packet». Цей пакет містить 6 байт зі значенням FF, за якими слідує 16 повторень MAC-адреси цільового адаптера. Його реалізація вимагає обов'язкової підтримки на рівні BIOS/UEFI материнської плати та мережевого контролера, який має залишатися під напругою блока живлення Standby Power для моніторингу трафіку. Функціонування через бездротові мережі Wi-Fi зазвичай є обмеженим через відключення радіомодуля в режимі сну WoWLAN. Недоліком є низька надійність доставки пакетів через маршрутизатори особливо при проходженні через NAT та різні підмережі, що вимагає налаштування спеціальних ретрансляторів Relay Agents. Крім того, протокол є одностороннім fire-and-forget і не дозволяє виконати примусове вимкнення системи у стані зависання.

Використання WoL через Інтернет вимагає додаткової конфігурації мережевої інфраструктури. Оскільки magic packet є широкомовним broadcast пакетом, він не може безпосередньо маршрутизуватися через зовнішню мережу. Типовими рішеннями цього є налаштування directed broadcast або subnet directed broadcast на маршрутизаторі для переадресації UDP-пакетів на порт 7 або 9 (стандартні порти WoL) на broadcast-адресу локальної мережі (наприклад, 192.168.1.255); використання VPN-підключення для входу безпосередньо в локальну мережу перед надсиланням magic packet; розгортання спеціалізованих

WoL-relay сервісів або пристроїв усередині локальної мережі, які приймають команди ззовні та ретранслюють magic packets всередину мережі.

Інтелектуальні розетки Smart Plugs з функцією AC Recovery базуються на використанні керованих через Wi-Fi або ZigBee розеток у поєднанні з налаштуванням політики живлення в BIOS «Restore on AC Power Loss» (автоматичне увімкнення при подачі напруги). Вони дозволяють здійснити повний цикл перезавантаження шляхом фізичного переривання та відновлення ланцюга живлення 220В. Це єдиний метод, що гарантовано виводить систему з будь-якого стану зависання. Проте, раптове знеструмлення несе значні ризики пошкодження файлової системи, втрати незбережених даних у кеші запису дисків та скорочення ресурсу компонентів блоку живлення через перехідні процеси. У зв'язку з війною, саме через постійні відключення електроенергії в мене вийшли з ладу SSD та блок живлення. А ще відсутня можливість виконання коректного завершення роботи операційної системи.



Рисунок 2.3 – Приклад розумної розетки

Електромеханічні актуатори (Fingerbots, SwitchBot) це компактні роботизовані пристрої з сервоприводами, призначені для фізичної імітації натискання кнопки живлення на корпусі комп'ютера. Керування здійснюється через Bluetooth або шлюз Wi-Fi. Вони забезпечують абсолютно неінвазивний метод керування, сумісний з будь-яким типом корпусу без необхідності втручання у внутрішню схемотехніку ПК. Серед недоліків у них низька надійність механічного кріплення, ризик відклеювання під навантаженням, необхідність періодичного обслуговування або заміни елементів живлення, а також затримки у спрацюванні через хмарні сервіси.



Рисунок 2.4 – Розумна кнопка SwitchBot

Спеціалізовані карти розширення PCIe (Tuya, eWeLink), що встановлюються у вільний слот PCI-e для отримання чергового живлення та підключаються паралельно до контактів кнопки Power на материнській платі. Вони забезпечують повністю незалежний Out-of-Band канал керування живленням через власні модулі Wi-Fi або LTE, що дозволяє керувати ПК навіть при несправності основної мережевої карти. Але у них є критична залежність від сторонньої хмарної інфраструктури китайських виробників, потенційні ризики інформаційної безпеки, а також повна відсутність можливості передачі відеосигналу, обмеження функціоналу виключно керуванням живленням.

Метод прямого апаратного керування ATX Direct GPIO можна назвати золотим стандартом для систем класу IP-KVM. Передбачає гальванічно розв'язане

підключення контролера через оптопари або реле безпосередньо до контактних груп POWER_SW та RESET_SW на материнській платі. Він забезпечує повний та гнучкий контроль над процесом: дозволяє емулювати як короткочасне натискання для увімкнення або коректного вимкнення через ACPI, так і тривале утримання 5 секунд для примусового вимкнення. Також дозволяє здійснювати зворотний моніторинг реального статусу живлення шляхом зчитування сигналу з контактів POWER_LED.

Огляд бюджетних апаратних KVM-рішень для сегмента SOHO.

Останні декілька років ринок доступних IP-KVM пристроїв почав активно розвиватися, пропонуючи альтернативи вживаним дорогим enterprise-рішенням для домашніх робочих станцій та малих офісів. Нижче представлено детальний огляд найбільш доступних варіантів, в тому числі створювану систему.

Створювана система на базі Raspberry Pi 4B. Система розробляється як універсальна платформа, що поєднує функціональність IP-KVM з можливостями IoT-хабу для розумного дому. Апаратна основа складається з Raspberry Pi 4 Model B 4GB з чотирьохядерним процесором ARM Cortex-A72 на частоті 1.5 ГГц, що забезпечує достатню обчислювальну потужність для паралельного виконання множини задач. Захоплення відео здійснюється через зовнішній HDMI-USB адаптер на базі чіпсету MS2109 або MS2130, що підтримує роздільну здатність 1920×1080 при 30 або 60 кадрах на секунду з апаратним кодуванням H.264 або MJPEG. Емуляція USB HID-пристроїв реалізується через функціональність USB OTG (On-The-Go) Raspberry Pi 4B, використовуючи kernel gadget framework Linux. Унікальною особливістю створюваної системи є інтеграція 3.5" TFT LCD дисплею 320×480 з резистивним тачскріном, підключеного через GPIO-сплітер, що дозволяє зберегти доступ до решти GPIO-пінів для розширення функціональності.

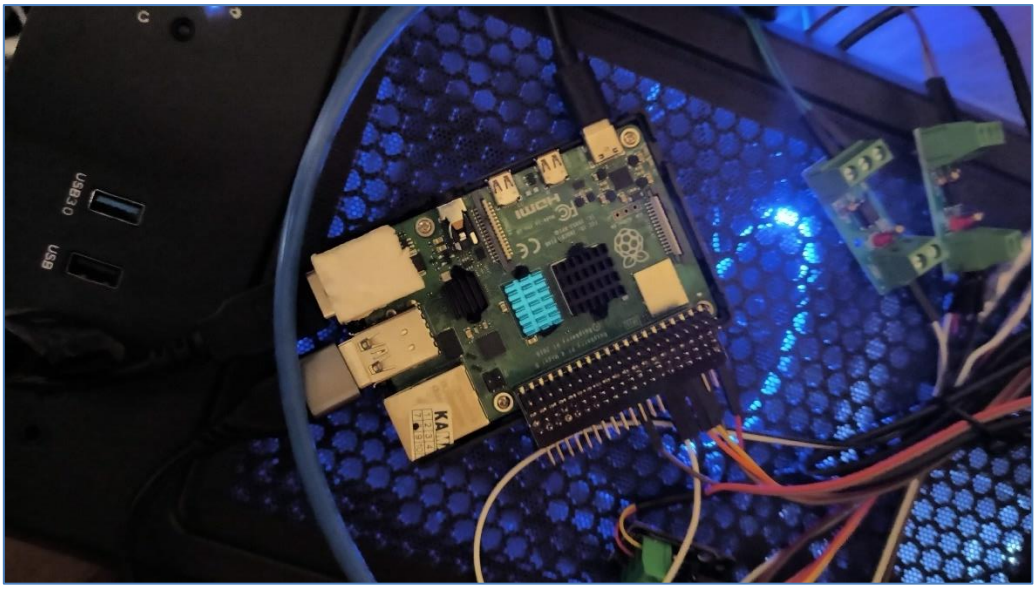


Рисунок 2.5 – Створювана система на базі Raspberry Pi 4B

Керування живленням реалізовано через три оптопари PC817, що забезпечують повну гальванічну розв'язку між Raspberry Pi та материнською платою керованого комп'ютера. Система підтримує як апаратну емуляцію кнопки живлення через GPIO-керування оптопарами, так і програмний метод Wake-on-LAN для випадків, коли мережевий адаптер цільового ПК підтримує відповідну функцію [23]. Вкрай важливою перевагою створюваної системи є можливість використання Raspberry Pi як USB mass storage device для віддаленого встановлення операційних систем. Потенціал як хабу розумного дому реалізується через 40-контактний GPIO-інтерфейс Raspberry Pi, що дозволяє підключати широкий спектр датчиків та актуаторів: температурні та датчики вологи (DHT11, DHT22, BME280, DS18B20) для моніторингу мікроклімату; датчики руху PIR для автоматизації включення систем при виявленні присутності; реле для керування освітленням, вентиляцією або іншим обладнанням; датчики відкриття дверей/вікон для систем безпеки; можливість інтеграції з протоколами розумного дому через USB-адаптери (Zigbee через ConBee II, Z-Wave через Aeotec Z-Stick). Програмне забезпечення може бути інтегроване з популярними платформами автоматизації (Home Assistant, OpenHAB, Node-RED) через REST API або MQTT-брокер, що дозволяє створювати складні сценарії автоматизації. Собівартість системи становить приблизно $100-150, що включає всі необхідні компоненти.

Sipeed NanoKVM Pro, випущена наприкінці 2024 року, вирішує суттєві проблеми з безпекою базової моделі, такі як наявність прихованого мікрофону, можливості віддаленого прослуховування та закритість частини програмного забезпечення. NanoKVM Pro використовує двоядерний ARM-процесор замість RISC-V, що забезпечує значно вищу продуктивність. Пристрій підтримує захоплення відео 4K@60fps з H.264 кодуванням та має затримку навіть нижчу, ніж PiKVM. Версія Pro оснащена 1 ГБ LPDDR4X RAM, 32 ГБ швидкої eMMC та Gigabit Ethernet. Базова версія коштує $79, версія з Wi-Fi та PoE – $105. Важливою перевагою є сумісність з програмним забезпеченням PiKVM, що дозволяє встановити перевірений open-source стек замість пропрієтарної прошивки. Ця модель є значно безпечнішою альтернативою базовому NanoKVM.



Рисунок 2.6 – Sipeed NanoKVM Pro

JetKVM позиціонується як компактний та безпечний IP-KVM з активною командою розробників. Пристрій використовує SoC Rockchip RV1106G3 з одноядерним ARM Cortex-A7 на частоті 1.0 ГГц та апаратним кодером H.264/H.265. Він оснащений 256 МБ DDR3L RAM та 8 ГБ eMMC. JetKVM підтримує захоплення HDMI-відео з апаратним кодуванням H.264/H.265, що забезпечує ефективне використання пропускної здатності. Унікальною особливістю є вбудований 1.69" сенсорний дисплей для локального керування та відображення статусу. Базова ціна

становить $69, з додатковою платою розширення ATX за $15 для керування живленням. Пристрій має відкритий вихідний код та активну спільноту розробників. Недоліком є мала кількість оперативної пам'яті (256 МБ), що може обмежувати можливості при складних сценаріях використання.



Рисунок 2.6 – JetKVM

Luckfox PicoKVM є легким інструментом IP-KVM управління на базі процесора RV1106G3. Пристрій підтримує захоплення HDMI-відео 1920×1080@60fps з кодуванням H.264 та затримкою 80-200 мс. Він оснащений 256 МБ DDR3 RAM та 8 ГБ eMMC для операційної системи. Унікальною особливістю є вбудований 1.54" сенсорний дисплей для відображення IP-адреси, статусу підключення та стану системи. PicoKVM підтримує емуляцію USB-клавіатури, миші та USB-накопичувача, а також має GPIO-інтерфейс для керування живленням хост-комп'ютера. Базова версія без дисплея та корпусу коштує $38.99, повна версія з металевим корпусом та дисплеєм – $61.99-89. Платформа має 100 Мбіт/с Ethernet порт та підтримує розширення через TF/SD карту. Це робить PicoKVM привабливим варіантом для бюджетних проєктів домашньої автоматизації.



Рисунок 2.7 – Luckfox PicoKVM

GL.iNet Comet Pro є покращеною версією базового Comet з суттєво підвищеними характеристиками. Пристрій використовує чотириядерний ARM Cortex-A53 процесор, 1 ГБ DDR3 RAM та 32 ГБ eMMC без можливості розширення. Важливою відмінністю є подвійний HDMI з функцією passthrough, що дозволяє одночасно захоплювати відео та підключати локальний монітор. Comet Pro підтримує відео до 4K@30 FPS або 1080p@60 FPS з H.264 кодуванням та затримкою ~30-60 мс. Пристрій оснащений 2.22" кольоровим ємнісним сенсорним дисплеєм для налаштування та локального керування. Він підтримує як провідне Gigabit Ethernet, так і бездротове підключення Wi-Fi 6. Споживання енергії в режимі простою становить 1.5-5 Вт. Порівняно з JetKVM, Comet Pro має вищу ціну, але пропонує значно кращі характеристики, більш відшліфований користувацький досвід та можливість бездротового розгортання. Орієнтовна ціна близько $150-200.



Рисунок 2.8 – GL.iNet Comet Pro

PiKVM V4 Plus є флагманською моделлю лінійки з максимальною функціональністю: підтримка 1920×1080@60Hz та 1920×1200@60Hz для максимальної сумісності з BIOS/UEFI; канали зв'язку Wi-Fi з опціональною зовнішньою антеною, Gigabit Ethernet; внутрішній USB 3.0 слот для flash-накопичувача; OLED дисплей 128×32 0.91" для відображення статусу; розширені опції живлення (DC 5.5x2.1mm barrel connector, 8V-24V); сталевий корпус з безшумною системою охолодження; MicroSD locker, Kensington security slot та два локаторних LED. Споживання енергії до 24 Вт (2A/12V). Ціна V4 Plus становить $300-400. Офіційна документація детально описує використання GPIO для підключення реле, світлодіодів, датчиків та інших компонентів. Підтримуються драйвери для pwm (ШІМ-керування), ezcoo (керування KVM-перемикачами), anelpwr (керування IP-PDU), та інші. Існує офіційна інтеграція PiKVM з Home Assistant, що дозволяє моніторити температуру CPU, швидкість вентилятора, статус тротлінгу, стан MSD та інші параметри. Планується розширення функціональності для керування GPIO, надсилання команд клавіатури та інших функцій безпосередньо з Home Assistant. Спільнота активно розробляє інтеграції з розумними розетками, системами освітлення та іншими пристроями автоматизації.



Рисунок 2.9 – PiKVM V4 Plus

Таблиця 2.1

Порівняння універсальності платформ та потенціалу IoT-інтеграції.

Критерій	**Створювана система**	**PiKVM V4+**	**Інші спеціалізовані KVM**
GPIO для розширення	40 пінів, повна документація	Доступно, документовано	Відсутньо або не документовано
USB-порти для адаптерів	4 порти USB 3.0/2.0	2-3 порти	Обмежено (1-2 порти, часто зайняті)
Підтримка GPIO-датчиків	Повна (temp, вологість, PIR тощо)	Можливо через розробку	Неможливо
Підключення реле	Пряме через GPIO	Можливо через GPIO	Неможливо або USB (не документовано)
Локальний дисплей	3.5" 320×480 touch	OLED 0.91" (V4+), інші	0-2.22" (обмежена функціональність)
MQTT-брокер	Легко встановити	Можливо встановити	Обмежені ресурси або закрита ОС
Home Assistant інтеграція	Можлива через API/MQTT	Офіційна наявна	Неможлива
Zigbee/Z-Wave підтримка	Через USB-адаптери	Через USB-адаптери	Відсутня або обмежена
REST API для автоматизації	Власний розширюваний	Повний	Обмежений (лише KVM)
Повноцінна Linux ОС	Raspberry Pi OS	Кастомна на базі Arch	Закрита або обмежена
Можливість встановлення ПЗ	apt, pip, Docker	Обмежено (readonly)	Закрита система

Продовження таблиці 2.1

Порівняння універсальності платформ та потенціалу IoT-інтеграції.

Критерій	**Створювана система**	**PiKVM V4+**	**Інші спеціалізовані KVM**
Спільнота розробників	Велика (RPi)	Активна (PiKVM)	Обмежена або відсутня
KVM + контроль освітлення	Через GPIO/реле	Можливо	Неможливо
KVM + система безпеки	PIR, камери, сигналізація	Можливо	Неможливо
KVM + автоматизація клімату	Датчики + керування HVAC	Можливо	Неможливо
Центральний IoT-хаб	Повна функціональність	Можливо	Неможливо

Висновки порівняльного аналізу.

Створювана мною система на базі Raspberry Pi 4B демонструє найвищий рівень універсальності серед усіх розглянутих IP-KVM рішень, трохи поступаючись надійністю PiKVM, який коштує у 2-3 рази дорожче. За рахунок ліпших технічних характеристик вона краще ніж PiKVM підходить на роль хабу розумного дому. PiKVM переважає за: надійністю апаратного захоплення вбудований відео TC358743 проти зовнішнього USB-адаптера; зрілістю програмного забезпечення та офіційною підтримкою; готовністю до промислового використання. Всі інші бюджетні та середньоцінові IP-KVM (NanoKVM, JetKVM, Luckfox, Comet) є виключно спеціалізованими KVM-пристроями без можливості розширення, хоча вони є готовими до використання з коробки.

2.3. Архітектура автономних платформ на базі одноплатних комп’ютерів. Принцип реалізації таких платформ на базі Raspberry Pi.

Принцип реалізації платформи IP-KVM та моніторингу на Raspberry Pi.

Реалізація спеціалізованих функцій, таких як IP-KVM, вимагає використання специфічних апаратних можливостей Raspberry Pi, які відрізняють його від звичайних ПК [24].

Фундаментальною основою для роботи IP-KVM є механізм USB OTG та емуляція периферії – здатність Raspberry Pi виступати не лише як USB-хост до якого підключають мишу, але і як USB-пристрій. Це реалізується завдяки наявності OTG-контролера (On-The-Go) в чіпах BCM283x/BCM2711. В ядрі Linux ця функція реалізована через підсистему USB Gadget API, зокрема модуль libcomposite та configfs. Це дозволяє програмно конфігурувати USB-ідентифікатори Vendor ID, Product ID та класи пристроїв, змушуючи цільовий комп'ютер розпізнавати Raspberry Pi як: стандартну клавіатуру та мишу HID, зовнішній накопичувач Mass Storage для монтування ISO-образів, мережеву карту Ethernet over USB для передачі даних.

Для передачі зображення робочого столу з мінімальною затримкою важливо уникнути програмного кодування відео на CPU, яке призводить до високого навантаження та перегріву. Архітектура Raspberry Pi включає потужний графічний процесор VideoCore IV або VI, який має виділені блоки для апаратного кодування H.264 OpenMAX або V4L2 M2M API.

Процес обробки відео виглядає наступним чином:

1) Захоплення «сирого» потоку (YUYV/MJPEG) з HDMI-грабера через підсистему V4L2 (Video4Linux2);

2) Передача кадрів у пам'ять GPU через механізм DMA (Direct Memory Access) без участі центрального процесора;

3) Апаратне стиснення у H.264;

4) Трансляція потоку клієнту через WebRTC або HTTP.

Оскільки система працює у віддаленому режимі, архітектура повинна передбачати механізми захисту від збоїв та пошкодження даних:

- *Файлова система тільки для читання Read-Only RootFS.* SD-карти пам'яті мають обмежений ресурс запису і чутливі до раптового зникнення живлення. Для вирішення цієї проблеми використовується технологія OverlayFS. Вона створює віртуальний шар поверх основної файлової системи, де всі зміни записуються в оперативну пам'ять (RAM), а не на диск. Після перезавантаження зміни зникають, а системний розділ залишається неушкодженим.
- *Апаратний сторожовий таймер Hardware Watchdog.* Система на чіпі BCM2711 містить вбудований апаратний таймер Watchdog. Принцип його роботи полягає у необхідності періодичного скидання таймера спеціальним системним демоном [25]. Якщо система зависає і демон перестає надсилати сигнали, таймер відраховує заданий інтервал, наприклад, 15 секунд, і надсилає сигнал апаратного скидання Reset на процесор, автоматично перезавантажуючи пристрій.
- *Безпечний віддалений доступ Secure Tunneling.* Архітектура передбачає інтеграцію VPN-клієнта, наприклад, WireGuard, на рівні мережевої ініціалізації systemd-networkd. Це дозволяє пристрою автоматично встановлювати захищене з'єднання з керуючим сервером одразу після завантаження, обходячи NAT та фаєрволи провайдера без необхідності білої IP-адреси.

Для реалізації функцій керування живленням ATX та моніторингу середовища використовується підсистема GPIO. Архітектура Raspberry Pi дозволяє керувати пінами через прямий доступ до регістрів пам'яті Memory Mapped I/O, що забезпечує високу швидкість реакції в наносекундах.

Для підключення сенсорів використовується шина I2C, яка дозволяє адресувати декілька датчиків на одній лінії, або 1-Wire для простих термометрів, що спрощує схемотехніку кінцевого пристрою.

Архітектура створюваної системи.

Загальна архітектура системи була організована по принципу IoT у вигляді чотирьох основних рівнів. Чітке розділення відповідальності між компонентами всередині архітектури дозволяє незалежно розробляти чи модифікувати кожен рівень без впливу на решту системи.

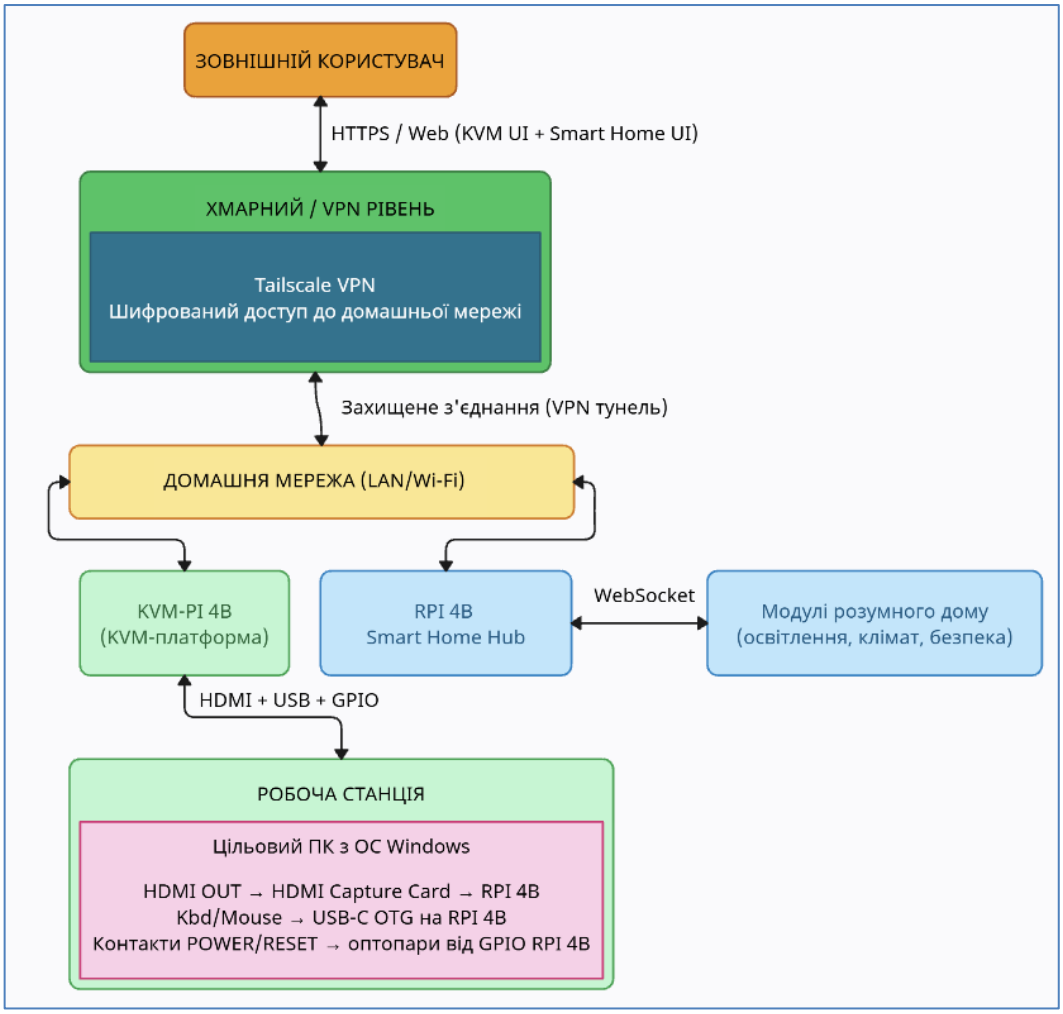


Рисунок 2.10 – Загальна архітектура системи

Рівень пристроїв, Device Layer, включає фізичні апаратні компоненти системи. Raspberry Pi 4 Model B виступає як центральний обчислювальний вузол та шлюз між всіма пристроями. USB-грабер MacroSilicon MS2109/MS2130 забезпечує захоплення HDMI-відео від керованого комп'ютера. Датчик температури DS18B20 моніторить температуру бекплейту відеокарти та може бути інтегрований в екосистему розумного дому Home Assistant. Оптопари PC817 забезпечують

гальванічно розв'язане керування кнопками живлення материнської плати. Цей рівень також включає потенційні точки розширення для підключення додаткових датчиків через GPIO (PIR-сенсори руху, датчики вологості DHT22, датчики відкриття дверей), актуаторів для керування освітленням та вентиляцією через реле-модулі, USB-адаптерів протоколів розумного дому (Zigbee ConBee II, Z-Wave Aeotec Z-Stick).

Рівень підключення, Connectivity Layer, забезпечує зв'язок між компонентами системи та зовнішніми клієнтами. Фізичні інтерфейси включають Gigabit Ethernet для провідного підключення з мінімальною затримкою та високою пропускною здатністю для передачі відео; Wi-Fi 802.11ac для бездротового доступу до веб-інтерфейсу з мобільних пристроїв; GPIO, I²C, SPI для підключення периферійних датчиків та актуаторів на рівні апаратного забезпечення; USB 3.0/2.0 для захоплення відео та можливого підключення додаткових пристроїв.

Протоколи комунікації реалізують багатошарову модель: HTTP/HTTPS для REST API endpoints, що надають доступ до конфігурації системи та телеметрії; WebSocket для низькозатримкової двосторонньої передачі відеопотоку та команд керування в режимі реального часу; MQTT для асинхронного обміну повідомленнями між компонентами системи та інтеграції з платформами розумного дому; 1-Wire для комунікації з датчиками температури DS18B20.

Рівень edge обробки, Edge Computing Layer, виконує локальну обробку даних, аналітику та прийняття рішень без залежності від хмарних сервісів. Підсистема захоплення та кодування відео через драйвер V4L2 забезпечує доступ до USB-грабера, FFmpeg виконує транскодування MJPEG у H.264 при необхідності, апаратний кодер VideoCore GPU Raspberry Pi прискорює H.264 кодування.

Підсистема керування живленням інтерпретує команди з веб-інтерфейсу або API, керує GPIO-пінами для активації оптопар PC817, реалізує Wake-on-LAN для програмного пробудження комп'ютера.

Рівень додатків, Application Layer, містить бізнес-логіку системи та інтерфейси взаємодії з користувачем. FastAPI backend надає RESTful API endpoints

для конфігурації системи, отримання телеметрії, керування живленням комп'ютера; WebSocket endpoints для потокової передачі відео та двосторонньої комунікації. В майбутньому можлива інтеграція в екосистему розумного дому з виведенням показників датчика DS18B20 в Home Assistant, а також інші варіанти модифікацій, які дозволяють можливості заліза [26].

Веб-інтерфейс користувача реалізовано як Single Page Application (SPA) на HTML5/JavaScript, надає віртуальну клавіатуру та мишу для керування комп'ютером, відображає панель керування живленням кнопками Power, Reset. MQTT брокер Mosquitto забезпечує publish-subscribe механізм для асинхронного обміну повідомленнями, інтеграцію з Home Assistant та іншими платформами автоматизації, підтримку множинних клієнтів без блокування основного функціоналу.

Висновок до підрозділу.

Архітектура створюваної автономної платформи на базі Raspberry Pi являє собою складний симбіоз апаратних можливостей SoC Broadcom та гнучкості ядра Linux. Використання апаратної акселерації відео, підсистеми USB Gadget та механізмів захисту файлової системи дозволяє перетворити одноплатний комп'ютер загального призначення на спеціалізований, надійний інструмент для задач IP-KVM та розумного дому.

2.4. Обґрунтування вибору апаратури.

Мій вибір апаратних компонентів для створюваної системи віддаленого керування базується на комплексному аналізі технічних характеристик, економічної доцільності, доступності на ринку та сумісності з іншими елементами системи. Кожен компонент відбирався з урахуванням специфічних вимог проєкту, включаючи надійність роботи в автономному режимі, можливість розширення функціональності та оптимальне співвідношення ціна-якість.

Вибір Raspberry Pi 4 Model B з 4 ГБ оперативної пам'яті як центрального обчислювального модуля системи обґрунтовується кількома головними факторами. Порівняння з альтернативними моделями лінійки дозволяє чітко визначити оптимальність цього вибору.

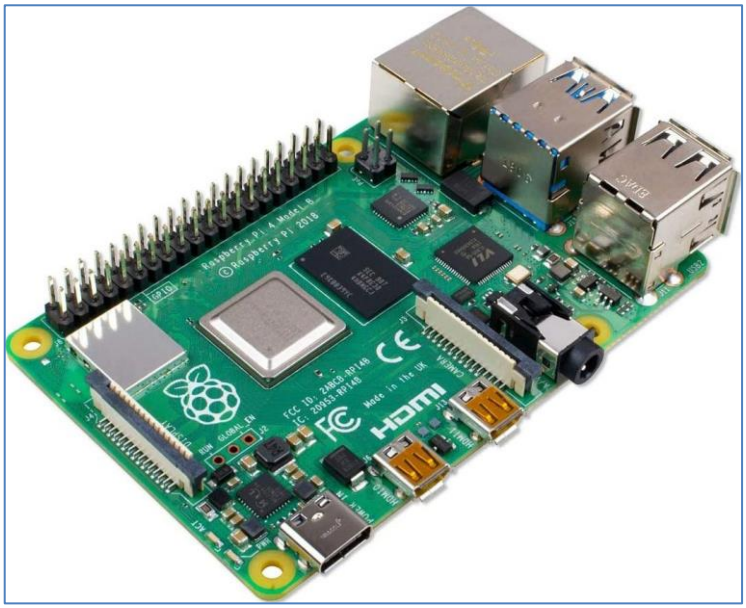


Рисунок 2.11 – Raspberry Pi 4 Model B

Raspberry Pi 5, випущений у 2023 році, демонструє суттєве зростання продуктивності: чотирьохядерний ARM Cortex-A76 процесор на частоті 2.4 ГГц забезпечує 2-3-кратне прискорення в порівнянні з Cortex-A72 1.5 ГГц у Pi 4.

Проте для створюваної IP-KVM системи ця перевага не важлива через кілька причин. Обробка відео 1080p@30 або 60 fps з H.264/MJPEG кодуванням не така складна обчислювальна задача для Pi 4 [27], який впевнено впорається з цим навантаженням. Додаткові задачі системи (збір телеметрії, веб-сервер, локальний дисплей) не створюють значного навантаження, яке виправдовувало б 2-3-кратне зростання продуктивності. Переломним недоліком Pi 5 є вартість $60-100 за версію 4 ГБ проти $45-65 за Pi 4, що становить підвищення на 30-60%. Енергоспоживання Pi 5 становить до 12-15 Вт під навантаженням проти 6-8 Вт у Pi 4. Необхідність активного охолодження для Pi 5 додає $5-10 до вартості системи. Сумісність компонетів також є проблемою: Pi 5 використовує нові камерні роз'єми MIPI замість

CSI/DSI, відсутній 3.5 мм аудіовихід, змінені позиції портів вимагають нових корпусів.

Тепер слід провести порівняння з молодшими моделями. Raspberry Pi 3 Model B+ має значно нижчу продуктивність з процесором Cortex-A53 на 1.4 ГГц, обмежений 1 ГБ RAM LPDDR2, що недостатньо для паралельного виконання KVM-функцій, телеметрії та веб-сервера. Мережеве обладнання Pi 3B+ обмежене Ethernet 300 Мбіт/с через USB 2.0 bridge замість Gigabit у Pi 4, що може створювати bottleneck при передачі відео. USB-порти обмежені USB 2.0, що знижує швидкість роботи з HDMI-захопленням. Raspberry Pi Zero 2 W з процесором на 1.0 ГГц та 512 МБ RAM взагалі не підходить для KVM-застосувань через недостатню обчислювальну потужність для обробки відео та обмежені можливості підключення периферії. Compute Module 4 теоретично підходить, але вимагає додаткової carrier board, що збільшує загальну вартість системи до $80-120 без очевидних переваг для даного застосування.

Саме тому вибір Raspberry Pi 4 Model B 4GB надає оптимальний баланс продуктивності, вартості та енергоефективності для створюваної системи.

Надважливий вибір карти пам'яті проводився крізь аналіз пропозицій надійних виробників на ринку України, бо якість та автентичність карток на Aliexpress викликає багато питань.

Kingston Canvas Go! Plus SDCG3 64 ГБ має сертифікації UHS-I, U3, V30 та A2. Заявлені швидкості становлять до 170 МБ/с для читання та 70 МБ/с для запису 64 ГБ версії. Реальні бенчмарки показують швидкість читання 89-93 МБ/с та запису 84-90 МБ/с у CrystalDiskMark. Робочий температурний діапазон від -25°C до +85°C робить карту придатною для промислових застосувань. Тести на довговічність показують, що карта витримує 14000 циклів перезапису перед першою помилкою, що є хорошим показником для споживчого класу.



Рисунок 2.12 – Kingston Canvas Go! Plus SDCG3 64 ГБ

Вибір модуля захоплення відео (Video Capture), який відповідає за оцифрування HDMI-сигналу від сервера, був виконаний з врахуванням того, що створювана система це ніби «проба пера» і для початку роботи буде достатньо бюджетного варіанту. Для реалізації бюджетної версії системи обрано MS2109, як найбільш економічно доцільне та поширене рішення. Чіп Actions AM8352 є прямим конкурентом MS2109 у тій же ціновій категорії і демонструє схожі характеристики, проте MS2109 має кращу підтримку в спільноті Linux (зокрема виправлення проблем зі звуком та ініціалізацією в проектах типу PiKVM).



Рисунок 2.13 – Карта відеозахоплення

Хоча USB 2.0 рішення використовують стиснення MJPEG, що вносить незначні артефакти на дрібному тексті та додає затримку, цього цілком достатньо для типових задач адміністрування [29]. Чіп MS2130 я розглядаю в якості апгрейду за наявності потреби для отримання 60 FPS, а CSI-рішення відхиляю через високу вартість.

Таблиця 2.2

Порівняння карт відеозахоплення

Модель	MacroSilicon MS2109	Actions AM8352	MacroSilicon MS2130	Toshiba TC358743 (CSI)
Інтерфейс	USB 2.0	USB 2.0	USB 3.0	MIPI CSI-2
Формат даних	MJPEG (стиснутий)	MJPEG (стиснутий)	YUY2 / MJPEG	RAW (нестиснутий)
Макс. якість	1080p @ 30fps	1080p @ 30fps	1080p @ 60fps	1080p @ 60fps
Затримка	Середня (~100 мс)	Середня (~100 мс)	Низька (~50-80 мс)	Мінімальна (~20-40 мс)
Ціна	~$5	~$8-15	~$20-25	~$40-50

Для контролю теплового режиму компонентів ПК був вибраний контактний цифровий датчик DS18B20. Датчики навколишнього повітря типу DHT11/DHT22 більше підходять для контролю були умов у серверній шафі та визнані неефективними через неможливість вимірювання контактної температури. Порівняння буде серед спеціалізованих рішень:

- *NTC Термістори* це найпоширеніші аналогові датчики для вимірювання температури поверхні. Переваги: низька вартість та миттєва реакція. Недоліки: вимагають наявності АЦП (аналого-цифрового перетворювача), який відсутній у Raspberry Pi, або реалізації складної схеми вимірювання через час розряду конденсатора. Потребують індивідуального калібрування.
- *Термопари Type-K з підсилювачем MAX6675* це промисловий стандарт для контактного вимірювання. Переваги: Широкий діапазон температур. Недоліки: Висока вартість модуля узгодження, займають інтерфейс SPI, є функціонально надлишковими для діапазону температур ІТ-обладнання (0-100°C).
- *Цифровий датчик DS18B20,* гільза з нержавіючої сталі якого дозволяє прикріпити сенсор безпосередньо до радіатора процесора або корпусу жорсткого диска для моніторингу контактної температури з високою теплопровідністю. Плата

має інтегрований підтягуючий Pull-up резистор 4.7 кОм, необхідний для стабільної роботи шини 1-Wire [28]. Інтерфейс 1-Wire дозволяє використовувати довгі кабелі до 10-20 метрів для розміщення сенсора в будь-якій точці серверної шафи, використовуючи лише один GPIO пін RPi та не вимагаючи додаткових перетворювачів.



Рисунок 2.14 – Цифровий датчик DS18B20

Для емуляції натискання кнопок Power/Reset необхідно гальванічно ізолювати ланцюги Raspberry Pi 3.3В від материнської плати ПК 5В/12В для захисту дороговартісної електроніки. Порівняння варіантів реалізації:

- Транзистори MOSFET/BJT забезпечують комутацію, але вимагають об'єднання земель GND двох пристроїв, що створює ризик вигорання портів RPi при різниці потенціалів.
- Електромеханічні реле мають надійну ізоляцію, але великі габарити, механічний знос контактів, клацання при спрацьовуванні та індуктивні викиди.
- Оптопари PC817 здійснюють передачу сигналу через світловий промінь. Вони забезпечують напругу ізоляції до 5 кВ с.к.з., що гарантує повний захист RPi від несправностей блоку живлення ПК. Плата має резистори 220 Ом для обмеження струму світлодіодів всередині оптопар PC817 (захист GPIO).

З міркувань безпеки, надійності та компактності плати було обрано саме оптопари PC817 [30].

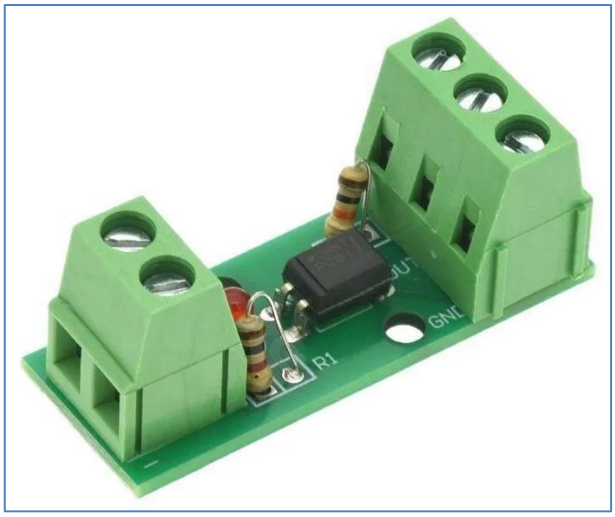


Рисунок 2.15 – Оптопара PC817

Також потрібно враховувати численні кабелі, перехідники та допоміжні плати, потреба в яких виникає в процесі фізичної інсталяції системи. Серед них є набір кабелів Dupont для підключення датчиків до GPIO. Плата-спліттер, що фізично розділяє лінії живлення (VBUS/GND до блоку живлення) та лінії даних (D+/D-/GND до ПК) необхідна, оскільки RPi 4 не може одночасно живитися і передавати дані через один порт від більшості материнських плат через нестачу струму. Кабелі-спліттери кнопок живлення ATX забезпечують можливість одночасного використання фізичної кнопки живлення на корпусі ПК та підключення до GPIO. А ще до переліку входять перехідники та кабелі USB-A та USB-C. За рахунок універсальності платформи є можливість легкої модифікації задач за рахунок додавання різних фізичних компонентів.

2.5. Обґрунтування вибору програмних компонентів.

Сама архітектура програмного забезпечення для системи IP-KVM вимагає знаходження оптимального балансу між продуктивністю, складністю розробки, надійністю функціонування та зручністю для кінцевого користувача. Враховуючи специфіку апаратної платформи Raspberry Pi 4, яка поєднує характеристики

вбудованої системи та персонального комп'ютера, мій вибір програмних компонентів впав на користь сучасних високорівневих фреймворків та оптимізованих інструментальних засобів, які мають низьку ресурсомісткість [31]. Подальший вміст підрозділу включає детальне техніко-економічне обґрунтування вибору кожного компонента та проведений порівняльний аналіз із найбільш поширеними альтернативами.

Операційна система. Для забезпечення базового функціонування апаратної платформи обрано операційну систему Raspberry Pi OS Desktop з графічним інтерфейсом, що базується на стабільному дистрибутиві Debian Linux.

Вибір на користь версії з графічним інтерфейсом користувача, незважаючи на підвищені вимоги до апаратних ресурсів порівняно з DietPi або Alpine, зумовлений необхідністю оперативного налагодження мультимедійних функцій безпосередньо на цільовому пристрої. Наявність попередньо встановлених пропрієтарних драйверів VideoCore гарантує коректне функціонування апаратного кодування відеопотоку, налаштування якого в Ubuntu або Alpine часто потребує складної конфігурації ядра.

Об’єктивно, core-елементом програмного комплексу є *підсистема захоплення відеосигналу HDMI* та його трансляції через веб-інтерфейс [32]. Для цієї мети обрано uStreamer – спеціалізований сервер, розроблений у рамках проекту PiKVM. Порівняльний аналіз з аналогами:

1) *MJPG-Streamer* це класичне рішення для роботи з веб-камерами. Недолік: застаріла архітектура та відсутність оптимізації для роботи з HDMI-граберами та апаратним кодером Raspberry Pi, що призводить до надмірного навантаження на центральний процесор.

2) *FFmpeg* у режимі стрімінгу це ніверсальний інструмент для обробки мультимедіа. Недоліками є висока затримка при трансляції в реальному часі внаслідок буферизації, а також складність конфігурування низькорівневих параметрів підсистеми V4L2.

3) *Потужний мультимедійний фреймворк GStreamer,* основний недолік якого, це висока складність інтеграції та надмірність функціоналу для задачі вбудовування відеопотоку у легкий веб-інтерфейс.

Обраний для проекту uStreamer, розроблений мовою C, використовує прямий доступ до інтерфейсу V4L2 M2M (Memory-to-Memory) для апаратної компресії у формати JPEG/H.264 засобами графічного процесора Raspberry Pi [33]. В ідеальних умовах він забезпечує стабільну частоту кадрів 60 FPS при роздільній здатності 1080p із затримкою менше 100 мс, а також підтримує динамічну зміну роздільної здатності, що стане важливим при відображенні екранів завантаження BIOS.

Серверна частина та API. Для реалізації бізнес-логіки керування (емуляція периферії, керування живленням, програмний інтерфейс) обрано асинхронний фреймворк FastAPI на мові програмування Python. Порівняльний аналіз з аналогами:

1) Flask – це синхронний мікрофреймворк, звідки і виникає його головний недолік: модель вводу-виводу може спричиняти затримки при одночасній обробці HTTP-запитів та керуванні GPIO. Також відсутня вбудована валідація вхідних даних.

2) Повнофункціональний веб-фреймворк Django. Його монолітна архітектура вкрай надмірна для задачі створення лаконічного REST API, що призводить до неефективного використання системних ресурсів.

3) Популярний JavaScript-фреймворк Express.js (Node.js). Головним недоліком є використання відмінної від системної мови програмування JavaScript для бекенду, тоді як взаємодія з апаратним інтерфейсом GPIO на Raspberry Pi має кращу документацію та реалізацію мовою Python.

Обраний для проекту FastAPI базується на стандарті ASGI та бібліотеці Pydantic. Фреймворк забезпечує автоматичну валідацію даних, автоматичну генерацію інтерактивної документації та високу продуктивність завдяки асинхронній моделі виконання async/await, що дозволяє ефективно обробляти WebSocket-з'єднання для передачі координат миші [34].

Для програмного керування контактами GPIO, для комутації реле ATX, індикації та зчитування сенсорів обрано *бібліотеку gpiozero*. Порівняльний аналіз з аналогами:

1) *RPi.GPIO* – базова бібліотека низького рівня. Недолік: вимагає написання значного обсягу надлишкового коду (ініціалізація, очищення ресурсів), має складну реалізацію обробки переривань та широтно-імпульсної модуляції (ШІМ).

2) *Pigpio* це бібліотека, що функціонує через системний демон. Недолік: забезпечує високу точність синхронізації, проте вимагає запуску окремого фонового процесу з правами суперкористувача, що ускладнює архітектуру системи.

3) *WiringPi* це портована версія бібліотеки з платформи Arduino. Недолік: припинення підтримки оригінальної бібліотеки автором створює ризики несумісності з новими версіями ядра Linux.

Обґрунтування вибору бібліотеки gpiozero засноване на її високорівневому декларативному інтерфейсі. Замість прямого маніпулювання номерами контактів та логічними рівнями, розробник оперує абстракціями (LED, Button, OutputDevice), що підвищує читабельність коду та мінімізує ймовірність помилок. Бібліотека також автоматично використовує найбільш ефективний доступний бекенд RPi.GPIO або lgpio.

Організація віддаленого доступу. Для забезпечення захищеного доступу до системи з глобальної мережі без наявності публічної IP-адреси було обрано Tailscale. Порівняльний аналіз з аналогами [35]:

1) Галузевий стандарт OpenVPN має низьку продуктивність (функціонує у просторі користувача), громіздку кодова база, складність налаштування інфраструктури відкритих ключів PKI, вимогу відкриття портів на маршрутизаторі.

2) Сучасний високопродуктивний протокол WireGuard у чистому вигляді. Для роботи в сценарії «сервер за NAT» непридатний без наявності зовнішнього VPS з публічною IP-адресою та ручного налаштування маршрутизації.

3) *ZeroTier,* аналог Tailscale, недоліком якого є контролер з відкритим вихідним кодом, складнішим у розгортанні, а пропрієтарна частина менш інтегрована в екосистему Linux порівняно з Tailscale.

Tailscale будує mesh-мережу на базі протоколу WireGuard [36, 37], забезпечуючи високу швидкість та криптографічну стійкість, до того ж автоматизує процеси проходження NAT (NAT traversal) та управління ключами. Це дозволяє розгорнути доступ до KVM за мінімальний час без необхідності модифікації налаштувань корпоративного міжмережевого екрану [38].

Інтерфейс користувача в браузері реалізується за допомогою поєднання *HTML5+CSS3+JavaScript*. Клієнтська частина створюється з використанням нативних веб-технологій без залучення ресурсомістких фреймворків. Порівняльний аналіз з аналогами:

1) *React*, популярна бібліотека від Meta. Недоліки: потребує етапу компіляції (build step), генерує значні за обсягом файли JavaScript, що збільшує час первинного завантаження у мережах з низькою пропускною здатністю.

2) *Angular*, повнофункціональний фреймворк від Google. Недолік: надмірна складність архітектури та великий розмір кінцевого пакету для задачі реалізації односторінкового додатка керування KVM.

3) *Vue.js*, легша альтернатива Angular/React. Недолік: використання абстракції Virtual DOM може вносити мікрозатримки при обробці інтенсивного потоку подій від пристроїв введення.

Отже, використання нативного JavaScript (ES6+) дозволяє досягти максимальної продуктивності клієнтської частини. Відсутність проміжного шару Virtual DOM та накладних витрат фреймворків гарантує миттєву передачу подій руху миші через WebSocket. Такий підхід також спрощує розгортання, оскільки сервер лише надає статичні файли без необхідності рендерингу на стороні сервера.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ ТА ЇЇ РЕЗУЛЬТАТИ

3.1. Підготовка робочого середовища, налаштування RPi 4B, встановлення ОС, драйверів, фізична інсталяція оптопар та сенсора.

Фізичне підключення компонентів системи.

Звісно ж, першочерговим етапом практичної реалізації проекту повинно стати фізичне підключення компонентів системи. Реалізація системи розпочинається з правильного монтажу та підключення всіх апаратних компонентів. Послідовність монтажу критична для забезпечення надійності та функціональності всієї системи.

Для перетворення аналогового або цифрового HDMI-сигналу від відеокарти цільового комп'ютера у цифровий потік даних, зрозумілий для операційної системи Raspberry Pi, використовується карта захоплення UVC05. Цей пристрій побудовано на базі спеціалізованого чипу AM8352, який забезпечує апаратне стиснення відео у формат MJPEG. Важливою перевагою цього пристрою є повна підтримка стандарту UVC (USB Video Class), що дозволяє інтегрувати його в середовище Linux без необхідності компіляції пропрієтарних драйверів ядра – пристрій автоматично розпізнається підсистемою V4L2. Карта підключається безпосередньо у синій порт USB 3.0 на Raspberry Pi 4B. Використання саме стандарту USB 3.0 є обов'язковим, оскільки пропускна здатність попереднього стандарту USB 2.0 теоретично максимум 480 Мбіт/с, але реально – близько 35-40 МБ/с, яка є недостатньою для стабільної передачі відеопотоку з роздільною здатністю 1080p або 720p при 60 кадрах на секунду. Нестача пропускної здатності призводить до пропуску кадрів, появи візуальних артефактів та значного зростання затримки вводу, що робить віддалене керування некомфортним. До HDMI-роз’єму карти захоплення підключається HDMI-кабель від відеокарти.

Для реалізації функцій апаратного ввімкнення та примусового перезавантаження сервера необхідно імітувати фізичне замикання контактів кнопок Power та Reset на материнській платі. Для цього використовується модуль гальванічної розв'язки на базі оптопари PC817. Гальванічна розв'язка є важливою вимогою безпеки: вона електрично ізолює логічні ланцюги керування Raspberry Pi 3.3В від ланцюгів материнської плати ПК 5V, захищаючи GPIO-порти від потенційних стрибків напруги, перешкод та статичної електрики з боку блоку живлення ПК. В обраній конфігурації з 5 контактами: +, -, VCC, GND, OUT, схема працює в режимі «відкритий колектор» (Open Collector). Це означає, що транзистор оптопари діє як ключ, який при спрацьовуванні замикає сигнальний провід на землю. Саме тому контакт VCC на вихідній стороні плати не використовується, бо не потрібна подача напруги на материнську плату, а лише комутація існуючого потенціалу на самій материнській платі.

Схема підключення до Raspberry Pi (Вхід/Керування) така:

- Контакти «+» (Anode) для Power Button підключається до GPIO17, Reset Button – GPIO27. Ці піни програмно переводитимуться у стан логічної одиниці (HIGH) для активації оптопари.
- Контакти «-» (Cathode) підключаються до загальної землі GND фізичного піна 20 на Raspberry Pi через спліттер.

Схема підключення до материнської плати ПК (Вихід/Виконання) така:

- Контакт «OUT» (Collector) підключається до плюсового (+) піна PWR_SW на материнській платі.
- Контакт «GND» (Emitter) підключається до мінусового (GND) піна PWR_SW на материнській платі.
- Контакт «VCC» залишається непідключеним.

При подачі керуючого сигналу з Raspberry Pi, внутрішній світлодіод оптопари освітлює базу фототранзистора, відкриваючи його перехід колектор-емітер. Це створює шлях для протікання струму від плюсового контакту

материнської плати на землю, що контролер материнської плати інтерпретує як натискання кнопки.

Для безперервного моніторингу температури відеокарти безпосередньо всередині корпусу керованого ПК використовується цифровий датчик температури DS18B20 у захищеному герметичному корпусі. Попередньо обраний модуль вже оснащений необхідною обв'язкою, зокрема підтягуючим резистором (pull-up resistor) номіналом 4.7 кОм, який забезпечує стабільний рівень логічної одиниці на лінії даних у стані спокою.

Підключення до Raspberry Pi здійснюється наступним чином:

- VCC (Red) підключається до піна живлення 1 або 17 на 3.3V . Хоча DS18B20 підтримує живлення 5V, для прямого підключення до логіки Raspberry Pi (яка працює на рівні 3.3V) безпечніше використовувати саме 3.3V, щоб уникнути пошкодження вхідних каскадів GPIO.
- GND (Black/Orange) підключається до піна GND – фізичний пін 6.
- DATA (Yellow/White) підключається до GPIO 4 – фізичний пін 7. Цей пін є стандартним для драйвера w1-gpio в ядрі Linux, хоча за необхідності може бути перепризначений.

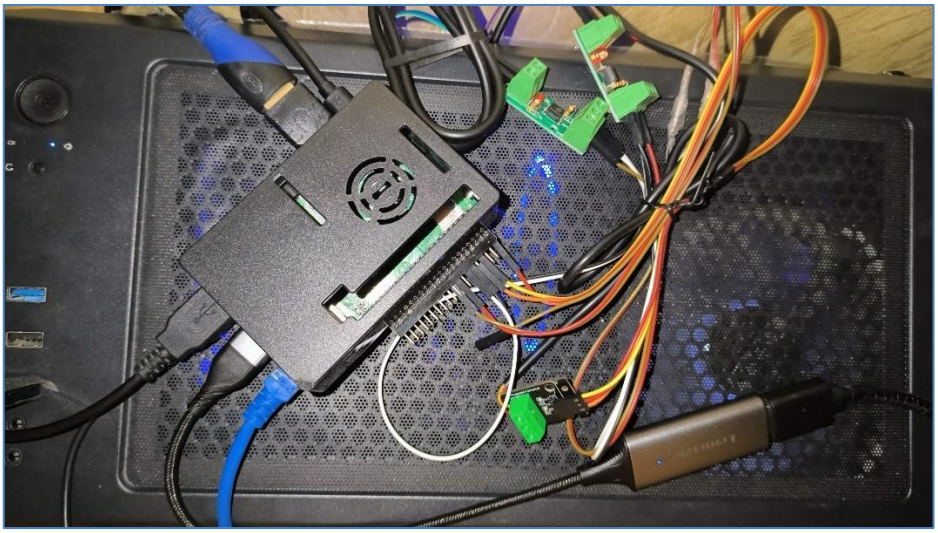


Рисунок 3.1 – Вигляд системи після підключення компонентів

Встановлення та налаштування операційної системи.

В якості програмної основи комплексу обрано Raspberry Pi OS Desktop 64-bit, яка побудована на базі Debian 12 Bookworm. Для запису образу на MicroSD карту використовується офіційна утиліта Raspberry Pi Imager, яка дозволяє виконати попередню конфігурацію системи ще до першого запуску.

Параметри конфігурації:

- Hostname: mikvm – унікальне ім'я пристрою в мережі.
- Username: mick – основний користувач з правами sudo.
- Увімкнено SSH для віддаленого адміністрування [39].

Після запису та встановлення карти в слот пристрою налаштовуємо статичну адресу 192.168.0.100, шлюз 192.168.0.1 та маску підмережі 255.255.255.0 в інтерфейсі налаштувань системи. Також вимикаємо DHCP. Після цього виконуємо перезавантаження командою *sudo reboot.*

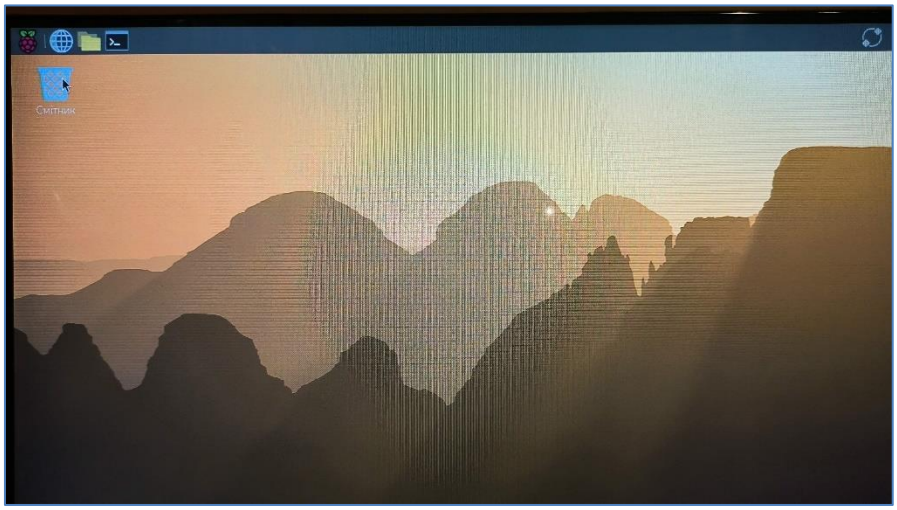


Рисунок 3.2 – Система після встановлення

Для продовження роботи необхідно з командної строки Windows підключитися за новою статичною адресою: *ssh mick@192.168.0.100* та ввести пароль.

Для роботи термодатчика DS18B20 необхідно активувати підтримку шини 1-Wire на рівні ядра ОС. Відкриваємо файл конфігурації у текстовому редакторі nano з правами суперкористувача [40]:

sudo nano /boot/firmware/config.txt.

У секцію [all] додаємо рядок для завантаження відповідного Device Tree Overlay:

dtoverlay=w1-gpio.

Ця директива вказує ядру завантажити драйвер w1-gpio при старті системи, використовуючи стандартний пін GPIO 4. Після збереження змін (Ctrl+O, Enter, Ctrl+X) необхідно перезавантажити систему: *sudo reboot*.

Перевіряємо, чи з'явився датчик DS18B20 на шині 1-Wire командою *ls /sys/bus/w1/devices/*.

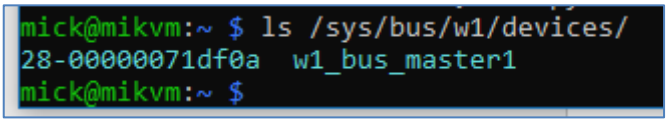


Рисунок 3.3 – Результат перевірки наявності датчика на шині 1-Wire

Далі налаштовуємо скрипт *check_temperature.sh* для відображення статусу температури датчика в консолі.

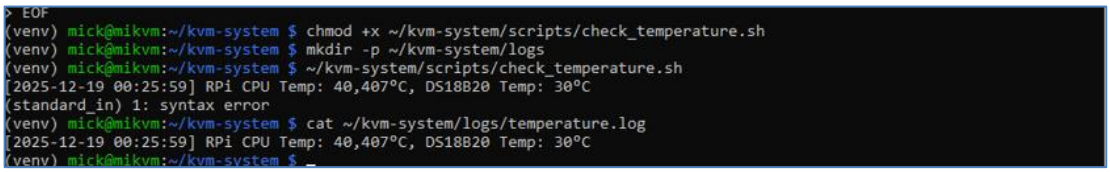


Рисунок 3.4 – Результат роботи датчика

Встановлення необхідних пакетів та залежностей.

Наступним етапом є підготовка програмного середовища для розробки та запуску backend-компонентів системи. Враховуючи специфіку роботи з відео та Python у середовищі Debian 12, цей процес має певні особливості.

Перед встановленням будь-якого ПЗ необхідно актуалізувати локальну базу пакетів та оновити встановлені компоненти до останніх версій для усунення вразливостей безпеки:

sudo apt update && sudo apt full-upgrade -y.

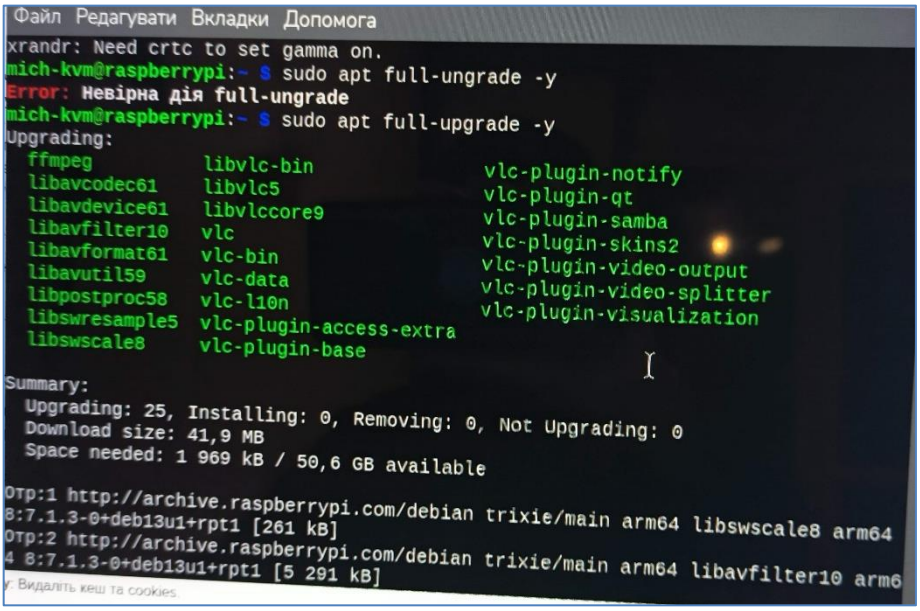


Рисунок 3.5 – Оновлення системи

Встановлюємо базовий набір інструментів, необхідних для адміністрування, моніторингу ресурсів та компіляції вихідного коду:

sudo apt install -y git curl wget vim build-essential htop net-tools.

build-essential містить компілятор GCC, утиліту make та інші інструменти, необхідні для збірки uStreamer та деяких Python-бібліотек. *htop* дозволяє в реальному часі відслідковувати навантаження на ядра процесора при трансляції відео.

Встановлення Python оточення та бібліотек для мультимедіа.

Сучасні версії Debian/Raspberry Pi OS впровадили стандарт PEP 668, який забороняє встановлення Python-пакетів через pip глобально в систему, щоб уникнути конфліктів з системним пакетним менеджером apt. Тому встановлення python3-venv є обов'язковим для створення ізольованих віртуальних середовищ для

проєкту. Також встановлюємо бібліотеки розробки (dev-пакети), необхідні для компіляції високопродуктивного стримера uStreamer:

sudo apt install -y python3-pip python3-venv python3-dev libjpeg-dev zlib1g-dev libevent-dev libbsd-dev

libjpeg-dev бібліотека для роботи зі стисненням JPEG, вкрай важлива для обробки MJPEG потоку. libevent-dev забезпечує механізми асинхронного вводу/виводу, що дозволяє uStreamer ефективно обробляти численні HTTP-запити.

Перевірка працездатності та діагностика периферії.

Після завершення налаштувань необхідно переконатися, що ядро операційної системи коректно ініціалізувало підключене обладнання.

Виконуємо команду *lsusb*, щоб перевірити, чи бачить система USB-пристрій:

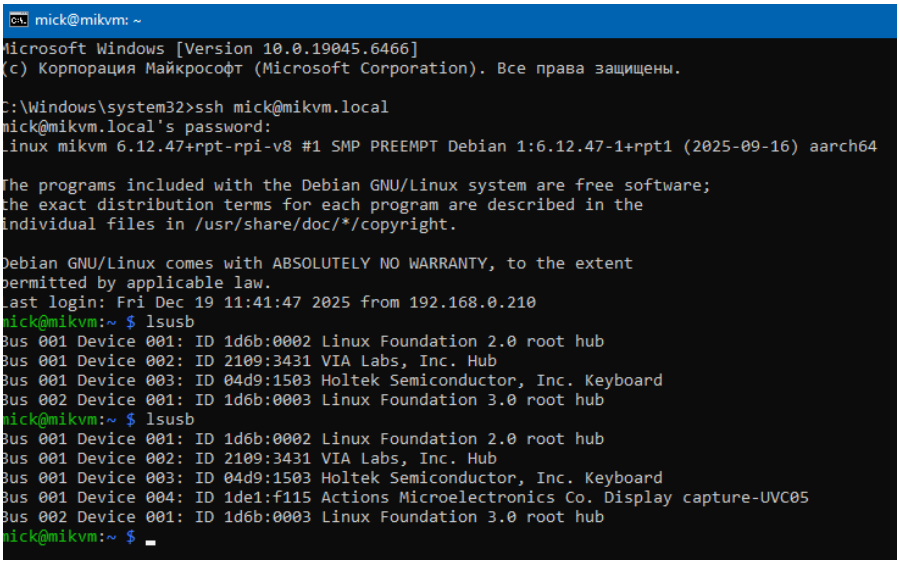


Рисунок 3.6 – Результат команди *lsusb*

У виводі є рядок з підключеною картою захоплення UVC05.

Далі перевіряємо наявність вузлів пристрою в каталозі /dev: *ls -l /dev/video**.

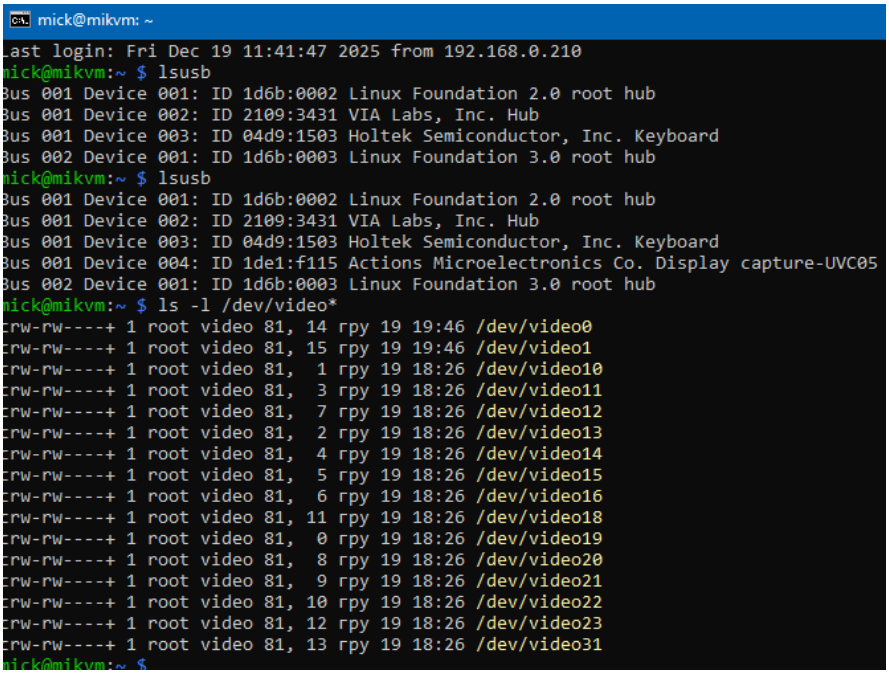


Рисунок 3.7 – Результат команди /dev: *ls -l /dev/video**

Створюється два пристрої /dev/video0 та /dev/video1, де перший використовується для захоплення відео, а другий – для метаданих.

Автоматичне перезавантаження системи.

Найголовнішим пріоритетом розроблюваної системи безперечно є стабільність її роботи. Для цієї цілі потрібна можливість перезапуску системи в будь-який момент часу. Кращим рішенням буде забезпечення автоматичного перезавантаження у разі критичних збоїв операційної системи або зависання системних процесів. Зокрема, мікропроцесор Raspberry Pi 4B Broadcom BCM2711 оснащено вбудованим модулем сторожового таймера Hardware Watchdog [41]. Інсталяцію потрібного демона було виконано командою:

sudo apt install watchdog

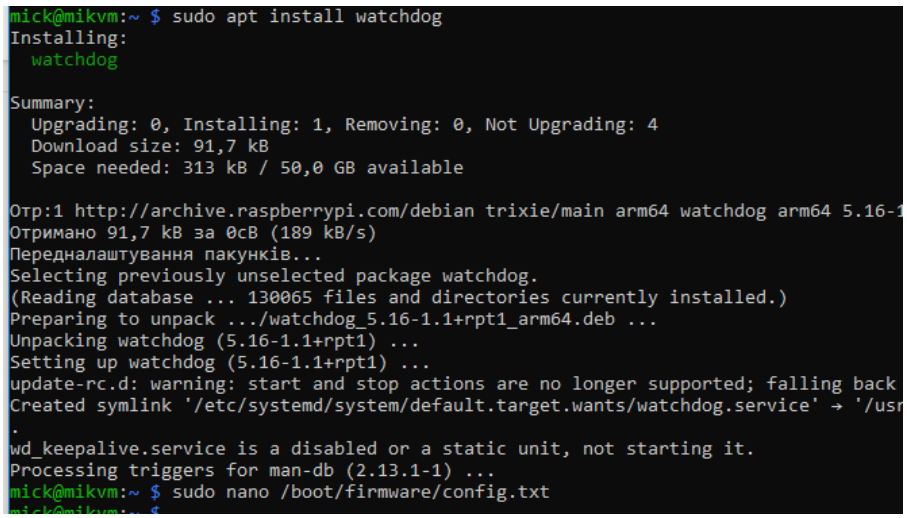


Рисунок 3.8 – Завантаження демона Hardware Watchdog

Для ініціалізації апаратного модуля було необхідно внести зміни до конфігураційного файлу завантажувача, а потім систему було перезавантажено.

sudo nano /boot/firmware/config.txt.

dtparam=watchdog=on

sudo reboot

Наступним етапом я налаштував логіку роботи служби моніторингу.

sudo nano /etc/watchdog.conf

Було виконано ідентифікацію пристрою, що вказує шлях до файлу сторожового таймера.

watchdog-device = /dev/watchdog

Встановлено тайм-аут, після спливання якого без отримання сигналу «keep-alive» відбудеться примусове апаратне перезавантаження.

watchdog-timeout = 15

Також було налаштовано моніторинг завантаження системи. Ця опція ініціює перезавантаження, якщо середнє навантаження на процесор перевищує допустимі межі протягом визначеного часу, наприклад, через некоректну роботу процесу uStreamer.

max-load-1 = 24

Для забезпечення безперервного захисту мною було налаштовано автоматичний запуск служби при завантаженні системи.

sudo systemctl enable watchdog

sudo systemctl start watchdog

sudo systemctl status watchdog

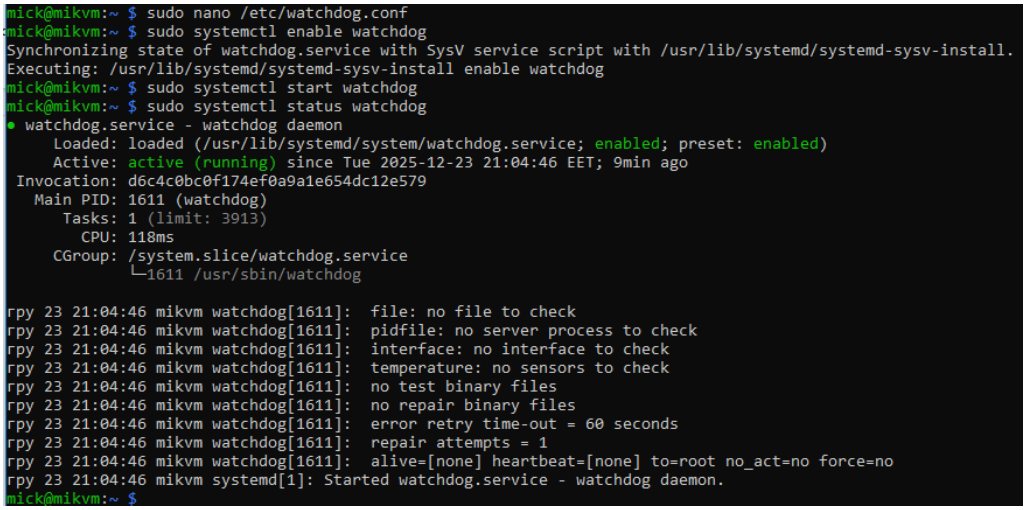


Рисунок 3.9 – Налаштування демона Hardware Watchdog

Для запобігання конфліктам між демоном watchdog та системним менеджером systemd, а також для підвищення надійності було налаштовано параметри systemd для взаємодії з апаратним таймером. Відкрито конфігураційний файл systemd:

sudo nano /etc/systemd/system.conf

У файлі конфігурації було активовано параметр, зобов'язує менеджер systemd періодично надсилати сигнали апаратному таймеру. У разі критичного збою самого systemd, таймер ініціює перезавантаження через 10 секунд.

RuntimeWatchdogSec=10

sudo reboot

Для перевірки функціональності таймера було застосовано команду типу «Fork bomb», яка рекурсивно створює нові процеси до повного вичерпання системних ресурсів. Після вичерпання таймеру 15 секунд демон автоматично перезавантажує систему.

:(){ :|:& };:

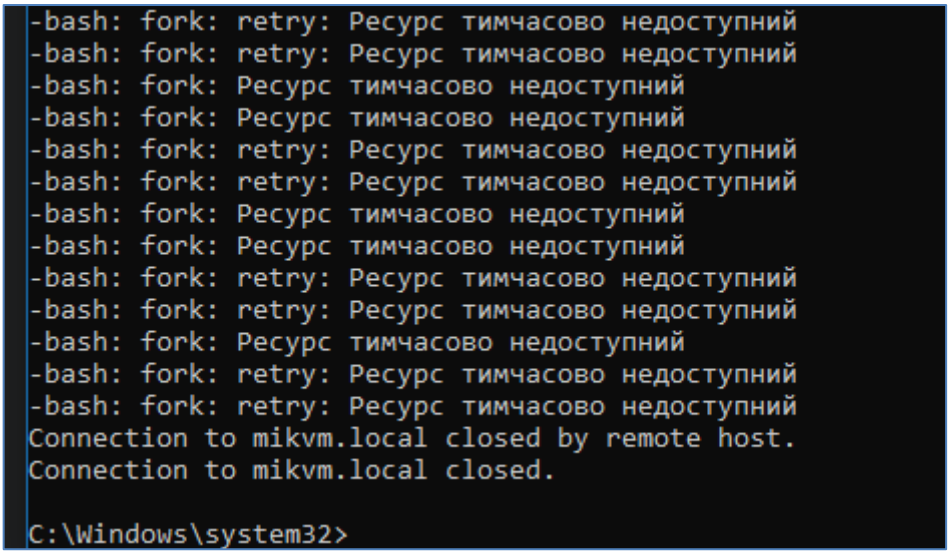


Рисунок 3.10 – Успішне перезавантаження системи

Виконання цих кроків гарантує, що апаратна платформа повністю готова до розгортання прикладного програмного забезпечення, а базові налаштування мережі та ОС забезпечать стабільність роботи майбутньої системи.

3.2 Реалізація підсистеми захоплення та трансляції відеопотоку, а також миші та клавіатури.

Центральним елементом програмно-апаратного комплексу KVM є підсистема медіа та керування, яка забезпечує два критичні процеси: трансляцію зображення екрана керованого комп'ютера та передачу команд керування. Враховуючи архітектуру Raspberry Pi 4B (ARM64) та використання карти захоплення бюджетного сегмента на чипі MacroSilicon MS2109/AM8352,

ключовим завданням реалізації стала мінімізація затримок та навантаження на центральний процесор.

Для забезпечення відеострімінгу було обрано програмне забезпечення uStreamer. Це спеціалізований сервер, розроблений спільнотою PiKVM, який значно перевершує класичний mjpg-streamer за стабільністю роботи з підсистемою ядра V4L2 та ефективністю використання пам'яті [42].

Враховуючи перехід на 64-бітну версію Raspberry Pi OS та проблеми з підтримкою застарілих бібліотек OpenMAX (OMX), було прийнято рішення виконувати компіляцію програми у чистій конфігурації. Це забезпечує максимальну стабільність, покладаючись на оптимізовані бібліотеки libjpeg та потужність 4-ядерного процесора Cortex-A72.

Для коректної роботи з пристроєм /dev/video0 користувачу були надані права доступу та додано до відповідних груп:

sudo usermod -a -G video,render mick

Процес встановлення починається з оновлення системи та встановлення залежностей. Для компіляції uStreamer необхідні бібліотеки для роботи з подіями (libevent), сумісністю BSD-систем (libbsd) та обробки JPEG:

sudo apt update

sudo apt install -y libevent-dev libbsd-dev libjpeg-dev build-essential git

Використання Git дозволяє отримувати найактуальнішу версію ПЗ. Сама компіляція виконується командою make без додаткових прапорів апаратного прискорення, що гарантує відсутність помилок сегментації пам'яті на нових ядрах Linux. Для цього в домашній директорії я створив папку для проєкту:

mkdir -p /home/mick/kvm_project/src

cd /home/mick/kvm_project/src

Використання прапорця --depth=1 дозволяє завантажити лише актуальну версію коду без історії змін, що пришвидшує процес:

git clone --depth=1 [https://github.com/pikvm/ustreamer] (https://github.com/pikvm/ustreamer)

Після переходу в створену директорію запущено компіляцію:

cd ustreamer

make

Тепер потрібно провести ряд тестових запусків трансляції. Тестовий запуск сервера для перевірки параметрів:

./ustreamer --device=/dev/video0 --host=0.0.0.0 --port=8080 \ --format=MJPEG --resolution=1280x720 --desired-fps=60 \ --drop-same-frames=30

Термінал вивів повідомлення Capturing … що означає успішний початок трансляції. Щоб побачити трансляцію було відкрито адресу http://192.168.0.100:8080 в браузері на комп'ютері.

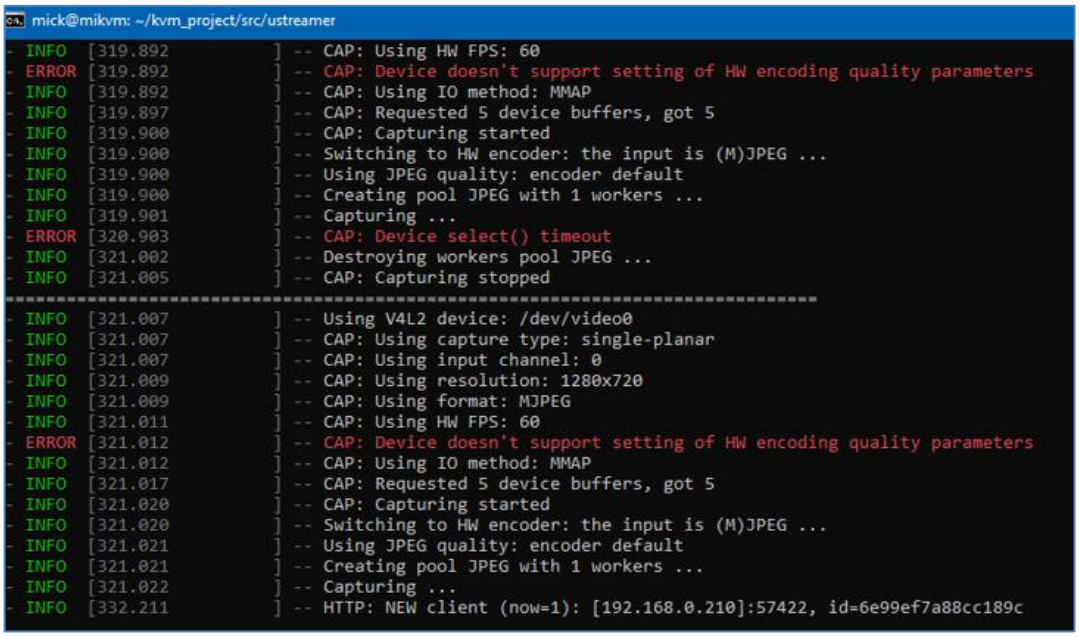


Рисунок 3.11 – Результат запуску трансляції

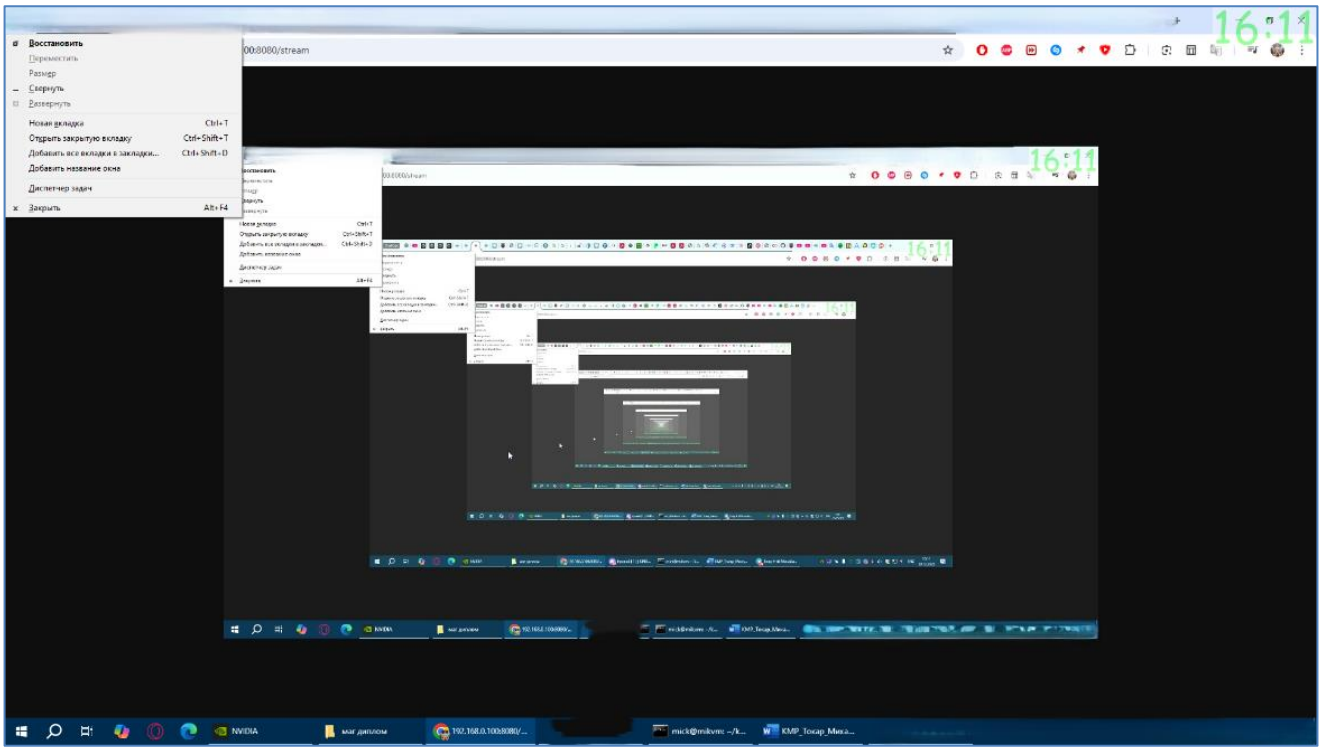


Рисунок 3.12 – Вигляд трансляції в браузері

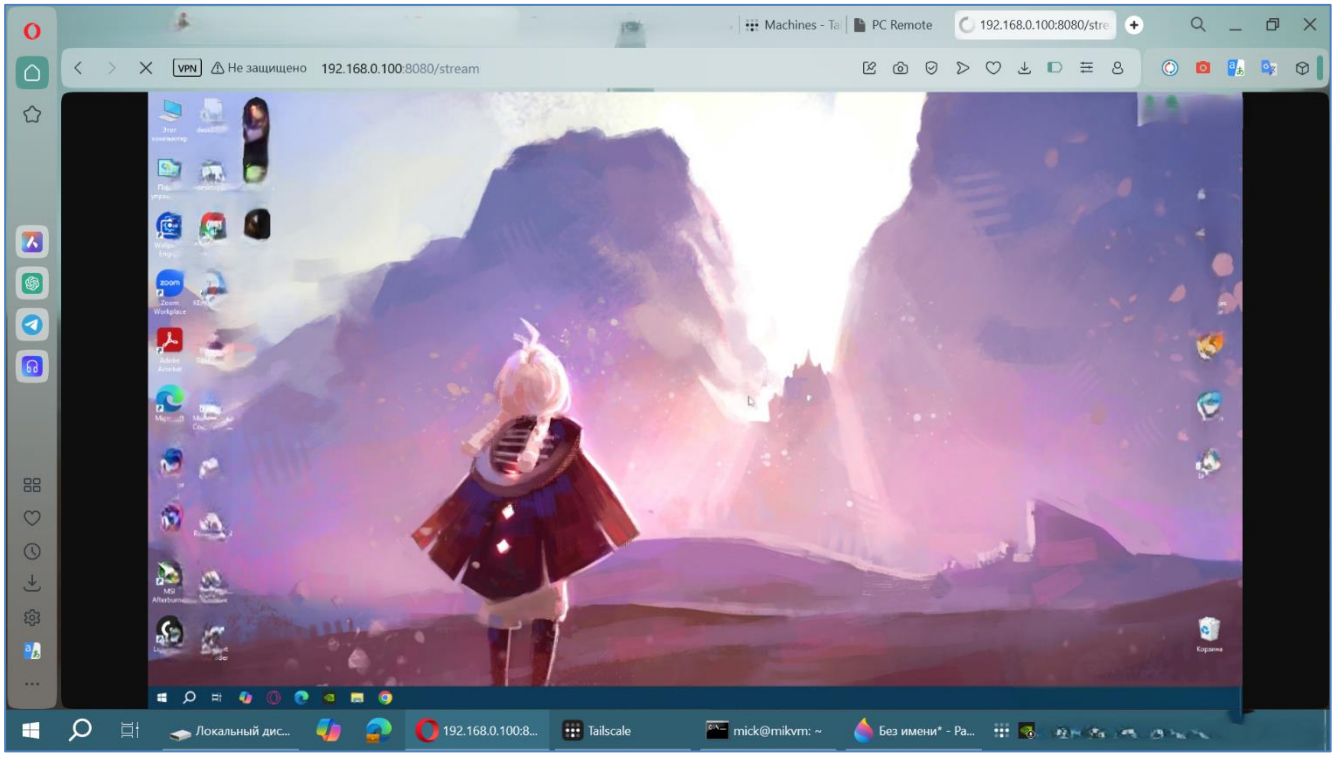


Рисунок 3.13 – Вигляд трансляції в браузері на іншому ПК

Зображення демонструє коректну передачу кольорів та чіткість тексту, що підтверджує правильність вибору формату MJPEG та роздільної здатності 720p для задач віддаленого адміністрування.

Аналіз роботи карти захоплення показав, що спроба програмного кодування відеосигналу з сирого формату YUYV у формат H.264 або JPEG на Raspberry Pi створює надмірну затримку >250 мс, через що потрібна деяка оптимізація. Тому було застосовано метод MJPEG Pass-through. Суть методу полягає в тому, що карта захоплення самостійно стискає відео у формат Motion JPEG. uStreamer отримує готові JPEG-кадри та транслює їх клієнту через HTTP, не витрачаючи ресурси CPU на стиснення.

Після багатьох тестувань різних конфігурацій було виявлено суттєві недоліки карти захоплення: мала пропускна спроможність інтерфейсу USB 2.0 та продуктивність вбудованого чіпу AM8352 дозволяють стабільну передачу картинки максимум 720p 60 кадрів, присутні проблеми з запуском апаратного прискорення, а також значний перегрів, що призводить до пропусків кадрів, рваних рухів мишею або крашу трансляції. В майбутньому для модифікації системи доцільно буде обрати більш потужний чіп відеозахоплення MS2130 або його аналог.

Для того щоб система працювала автономно 24/7, необхідно налаштувати uStreamer як системну службу (daemon), яка запускається автоматично при старті Raspberry Pi та перезапускає HTTP-сервер трансляції у випадку збоїв. Для підвищення надійності запуску, щоб уникнути помилок доступу або зайнятості пристрою при старті системи, було використано розширену конфігурацію з примусовим додаванням груп доступу та затримкою старту.

Створення файлу конфігурації виконується командою:

sudo nano /etc/systemd/system/ustreamer.service

Далі було прописано вміст файлу.

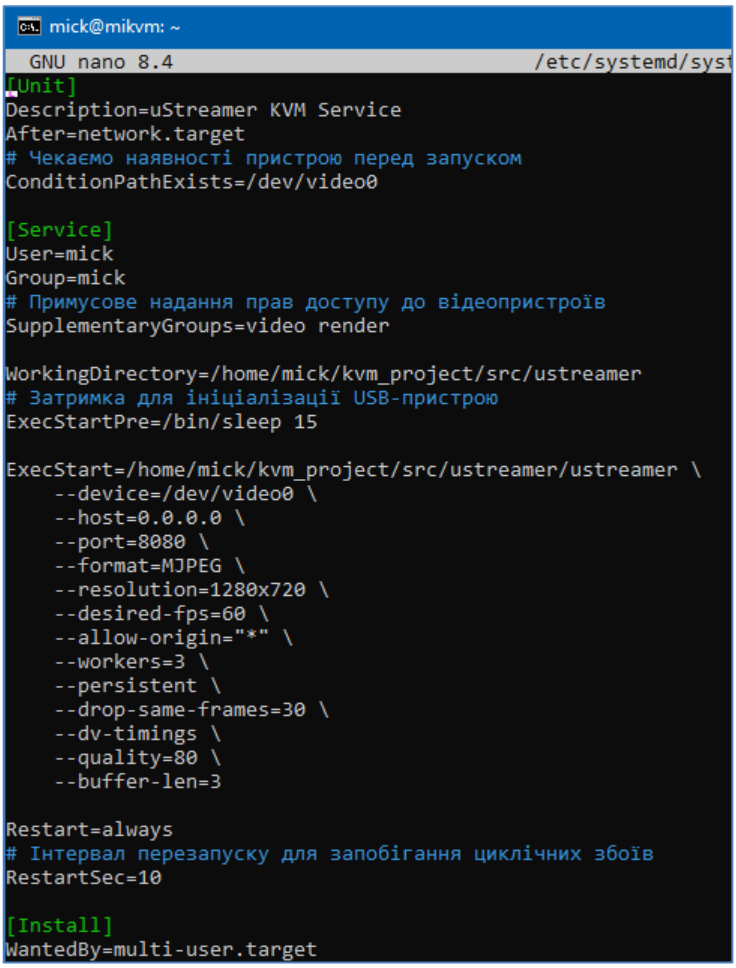


Рисунок 3.14 – Вміст файлу демона HTTP-сервера трансляції

Параметр --format=mjpeg примушує використовувати апаратний формат MJPEG, який видає карта AM8352. Це дозволяє уникнути програмного перекодування кольорів, що занадто вантажить процесор.

Параметр --resolution=1280x720 обмежує роздільну здатність.

Параметр --allow-origin="*" дозволяє вбудовувати відеоплеєр у будь-яку веб-сторінку.

Без параметру --persistent програма uStreamer завершить роботу, якщо вимкнеться комп'ютер або від'єднається HDMI кабель. З цим параметром вона показуватиме «заглушку» при відсутності сигналу.

Параметр --drop-same-frames=30 дозволяє пропускати дубльовані кадри, якщо зображення на екрані статичне, знижуючи навантаження на мережу.

Параметр --buffer-len=3 забезпечує короткий буфер кадрів для зменшення затримки відображення.

Активація служби командами:

sudo systemctl daemon-reload – Оновлює список доступних служб.

sudo systemctl enable ustreamer – Створює символічне посилання для запуску служби при старті ОС.

sudo systemctl restart ustreamer – Запускає службу негайно.

Відеопотік постійно доступний за адресою http://192.168.0.100:8080/stream.

Програмна реалізація системи віддаленого керування периферійними пристроями.

Метою даної розробки стало створення програмного комплексу KVM, що дозволить використовувати один комплект пристроїв введення для керування декількома обчислювальними машинами в одній локальній мережі.

Для реалізації було обрано клієнт-серверну архітектуру. Клієнт (Client/Master) це комп'ютер, до якого фізично підключені пристрої введення. Він перехоплює дії користувача та транслює їх у мережу. Сервер (Server/Slave) це цільовий комп'ютер, який приймає дані з мережі та емулює відповідні події на рівні операційної системи.

Для передачі даних обрано протокол UDP (User Datagram Protocol) замість TCP, бо керування мишею вимагає мінімальної затримки. Протокол TCP гарантує доставку пакетів, але витрачає час на встановлення з'єднання «рукостисканням» та підтвердження отримання кожного пакету. В той же час, UDP дозволяє відправляти пакети координат «пострілами». Якщо один пакет із координатами втрачено, це не є критичним, оскільки наступний пакет через 10 мс актуалізує положення курсора і забезпечить плавність руху.

Також важливо позбутися проблеми різниці фізичних меж екрана клієнта та сервера коли курсор досягає краю монітора на керуючому ПК, операційна система

перестає генерувати події руху, що унеможливлює подальше переміщення курсора на віддаленому екрані. Для цього розроблено та імплементовано алгоритм фіксації курсора. При активації режиму керування натисканням Right Ctrl курсор на клієнті примусово програмно переміщується в геометричний центр екрана.

Програмний засіб розроблено мовою Python 3 із використанням наступних бібліотек:

1) Pynput це кросплатформенна бібліотека для моніторингу та керування пристроями введення. Використовує системні хуки hooks для глобального перехоплення подій клавіатури та миші, навіть коли вікно консолі не активне.

2) Socket це стандартна бібліотека для мережевої взаємодії через інтерфейс сокетів. Виконує створення UDP-дейтаграм, прив'язка до портів, відправка та отримання байтових масивів.

3) Threading це реалізація багатопотоковості. Дозволяє розпаралелити процес прослуховування мережевого порту та опитування стану миші, запобігаючи блокуванню основного циклу програми Main Loop.

4) Ctypes забезпечує доступ до зовнішніх функцій DLL-бібліотек (Foreign Function Interface). Викликає функції GetSystemMetrics з системної бібліотеки Windows user32.dll для точного визначення роздільної здатності екрана, що необхідно для коректної роботи алгоритму центрування.

Підготовка середовища виконання.

Перед запуском програмного коду необхідно встановити інтерпретатор Python та зовнішні залежності. Команда інсталяції бібліотеки в cmd Windows на клієнті та сервері:

pip install pynput, де:

pip – виклик пакетного менеджера Python (Package Installer for Python).

install – команда, що ініціює процес завантаження та розпакування пакету.

pynput – ідентифікатор бібліотеки в репозиторії PyPI.

Після її виконання у систему завантажуються драйвери та скрипти, необхідні для програмного перехоплення введення.

Створення виконуваного файлу.

Оскільки система реалізована у вигляді скрипта, створення основного виконуваного файлу відбувається через текстовий редактор. Для цього створено текстовий файл kvm_daemon.py в робочій директорії. Слідом у файл було вставлено повний код, наведений у додатку [ДОДАТОК А].

Запуск серверної підсистеми.

Ця команда виконується на комп'ютері, яким необхідно керувати (Slave). Для подальшого запуску програми в фоні без виводу консолі на екран формат файлу було змінено на pyw. Команда запуску в cmd Windows:

python kvm_daemon.pyw server

Програма ініціалізує клас KVM Daemon, відкриває UDP-сокет на порту 6000 та входить у нескінченний цикл очікування вхідних пакетів.

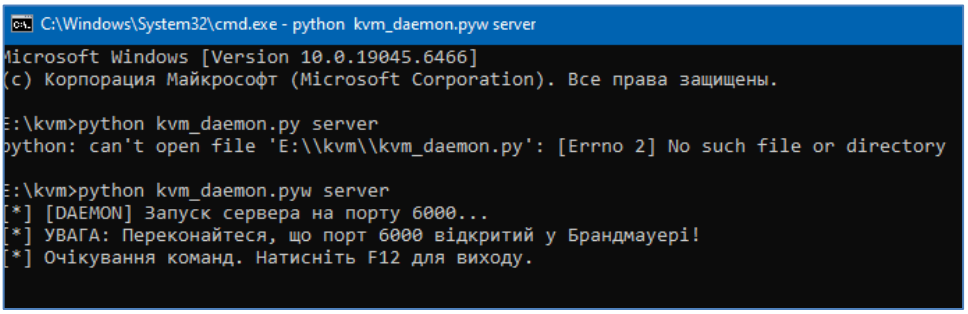


Рисунок 3.15 – Запуск сервера

Запуск клієнтської підсистеми.

Ця команда виконується на основному комп'ютері (Master), який буде керувати. Команда запуску в cmd Windows:

python kvm_daemon.py client 192.168.0.210

client – аргумент, що переводить програму в режим відправника. Активує listener (слухач) клавіатури та миші.

192.168.0.210 – IP-адреса цільового комп'ютера сервера у локальній мережі.

Скрипт встановлює цільову адресу для надсилання UDP-пакетів. Передача даних не починається автоматично (для запобігання втраті керування власним ПК), а очікує керуючого сигналу.

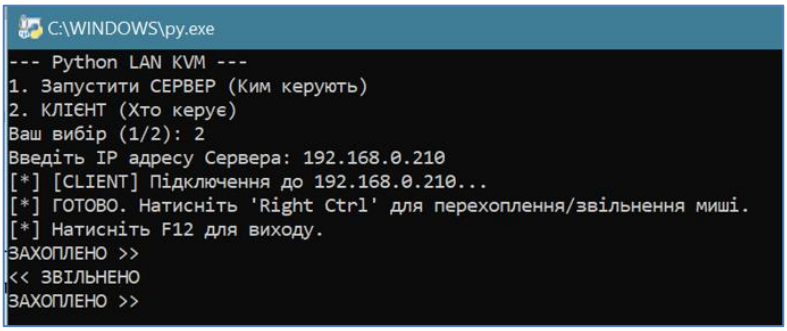


Рисунок 3.16 – Запуск клієнта

В результаті завдяки заздалегідь запущеній трансляції відеопотоку видно коректну роботу скриптів.

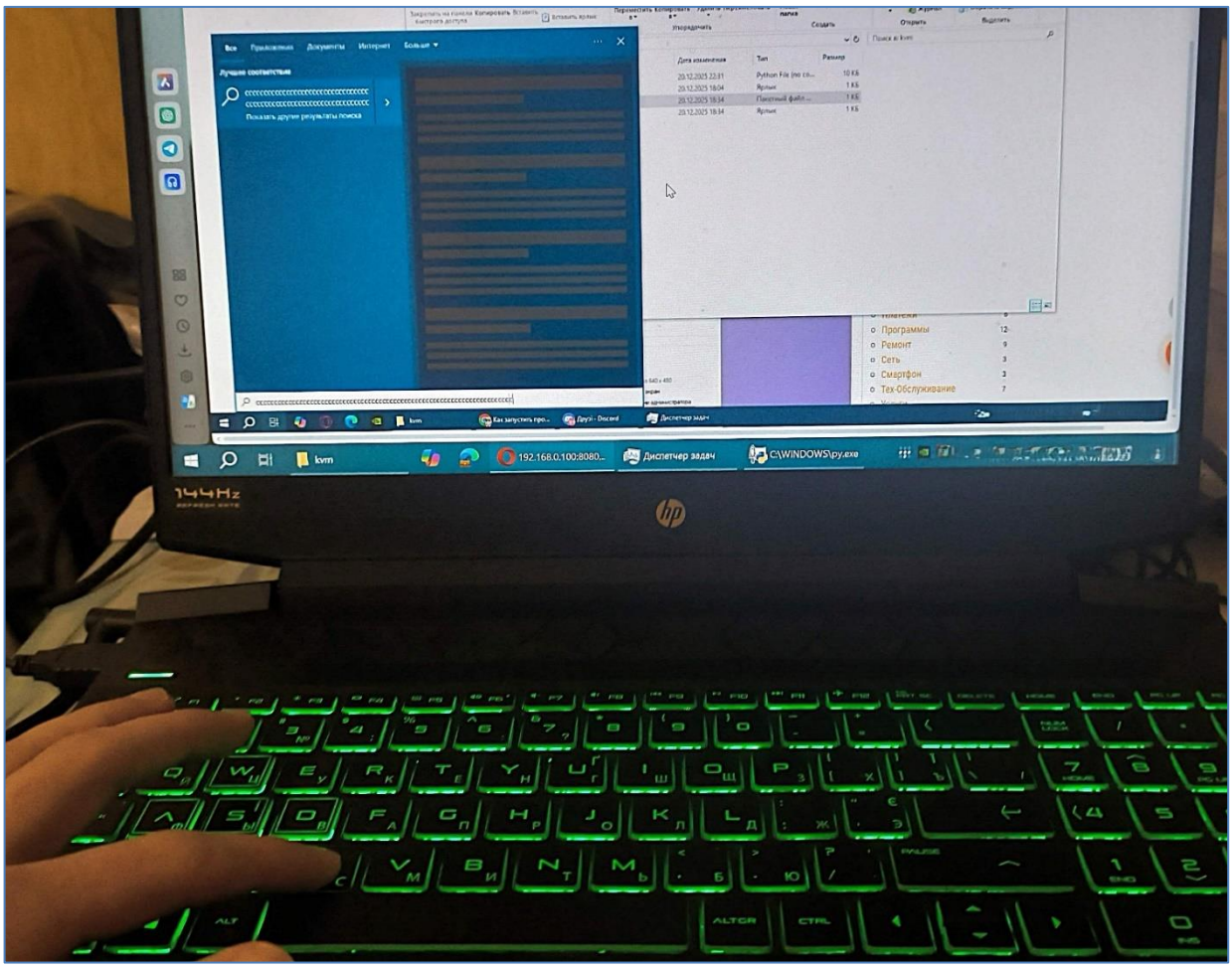


Рисунок 3.17 – Натискання клавіші C на клієнті та виконання дії на сервері

Для забезпечення автозапуску скрипта на сервері було створено файл start_service.bat із наступним змістом, а його ярлик переміщено в папку автозавантаження Windows:

start "" pythonw.exe "E:\kvm\kvm_daemon.pyw" server

start "" – команда Windows Shell для запуску програми в окремому потоці. Порожні лапки "" є обов'язковим параметром для заголовка вікна. Це дозволяє самому BAT-файлу завершити роботу одразу після запуску скрипта.

Реалізація механізмів керування.

Задля забезпечення ергономічності та безпеки використання системи мною було впроваджено спеціальні керуючі сигнали Hotkeys.

Клавіша Right Ctrl використовується як перемикач (toggle) захоплення рухів мишею з керування клієнтом на керування сервером, або навпаки. При натисканні вона змінює стан внутрішньої змінної is_capturing. Стан ON: координати миші фіксуються в центрі екрана клієнта, дельти зміщення транслюються на сервер. Стан OFF: хуки деактивуються, користувач керує власним ПК.

Аварійна зупинка «Kill Switch» клавішею F12 працює глобально на обох сторонах з'єднання. У випадку програмного збою (наприклад, втрата мережі при активному захопленні миші) дозволяє примусово перервати цикл виконання, закрити сокети та звільнити ресурси системи без необхідності перезавантаження ПК.

Керування віддаленим ПК через VPN Tailscale потребує дозволу в брандмауері для python та python.exe працювати через публічні мережі.

3.3 Створення веб-серверу для керування кнопками на передній панелі ПК

Далі в цьому підрозділі було описано практичний процес створення модуля для дистанційного керування живленням ПК. Як було вказано у попередньому

підрозділі, головним пріоритетом під час розробки була безпека материнської плати та Raspberry Pi, через що було реалізовано повну гальванічну ізоляцію між керуючим пристроєм та комп'ютером.

Налаштування програмного середовища.

Щоб дотримуватися сучасних стандартів розробки на Python та не засмічувати системні директорії, проєкт було розгорнуто в ізольованому віртуальному середовищі.

Спочатку було створено окрему папку для проєкту, щоб усі файли зберігалися в одному місці.

mkdir /home/mick/pc_remote

cd /home/mick/pc_remote

Для того, щоб бібліотеки проєкту не конфліктували з іншими програмами на Raspberry Pi, було створено віртуальне середовище Python.

python3 -m venv venv

Після цього в активоване віртуальне середовище було завантажено потрібні бібліотеки.

source venv/bin/activate

pip install fastapi uvicorn gpiozero

Для реалізації веб-серверу я обрав такий стек технологій:

- FastAPI – сучасний веб-фреймворк, який забезпечує високу продуктивність завдяки асинхронній обробці запитів та автоматично валідує вхідні дані [43].
- Uvicorn – легкий та швидкий ASGI-сервер, необхідний для запуску асинхронних Python-додатків.

- Gpiozero – високорівнева бібліотека для роботи з GPIO, яка надає зручний об'єктно-орієнтований інтерфейс і робить код значно читабельнішим у порівнянні зі застарілими аналогами, наприклад, RPi.GPIO.

Щоб вийти з віртуального середовища коли на початку рядка терміналу є напис «venv», треба ввести команду: *deactivate.*

Розробка основного керуючого ПЗ.

Програмна логіка реалізована в одному файлі main.py. Це асинхронний веб-сервер, який виконує дві задачі: надає веб-сторінку з кнопками та обробляє натискання цих кнопок, подаючи сигнали на GPIO.

У тій самій директорії було створено файл main.py. Далі в текстовому редакторі було вписано заздалегідь розроблений код серверу з вбудованим HTML-кодом інтерфейсу [ДОДАТОК Б].

nano main.py

Перебуваючи у директорії проєкту з активованим віртуальним середовищем, було виконано команду запуску:

python main.py

Після успішного старту у терміналі з'являється повідомлення про успішний запуск сервера Uvicorn:

INFO: Uvicorn running on http://0.0.0.0:8000.

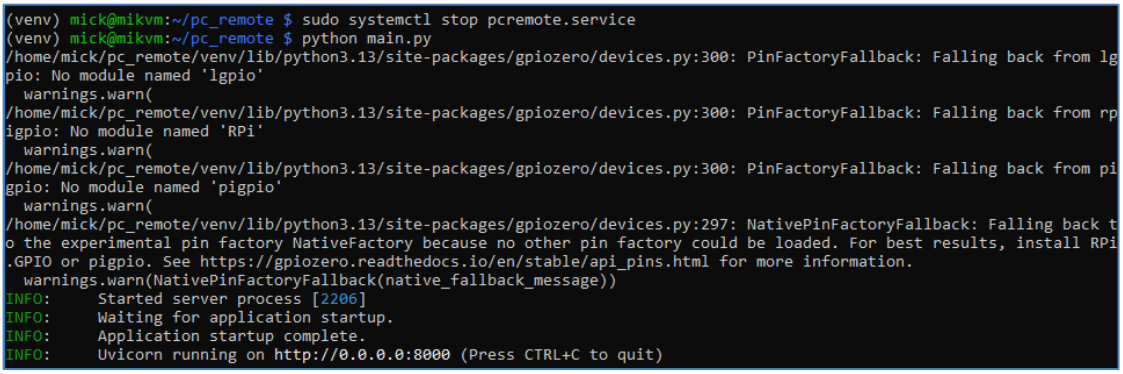


Рисунок 3.18 – Успішний запуск сервера

Перевірка через браузер.

Для перевірки доступності в браузері було прописано адресу Raspberry Pi в локальній мережі та вказано порт 8000:

http://192.168.0.100:8000.

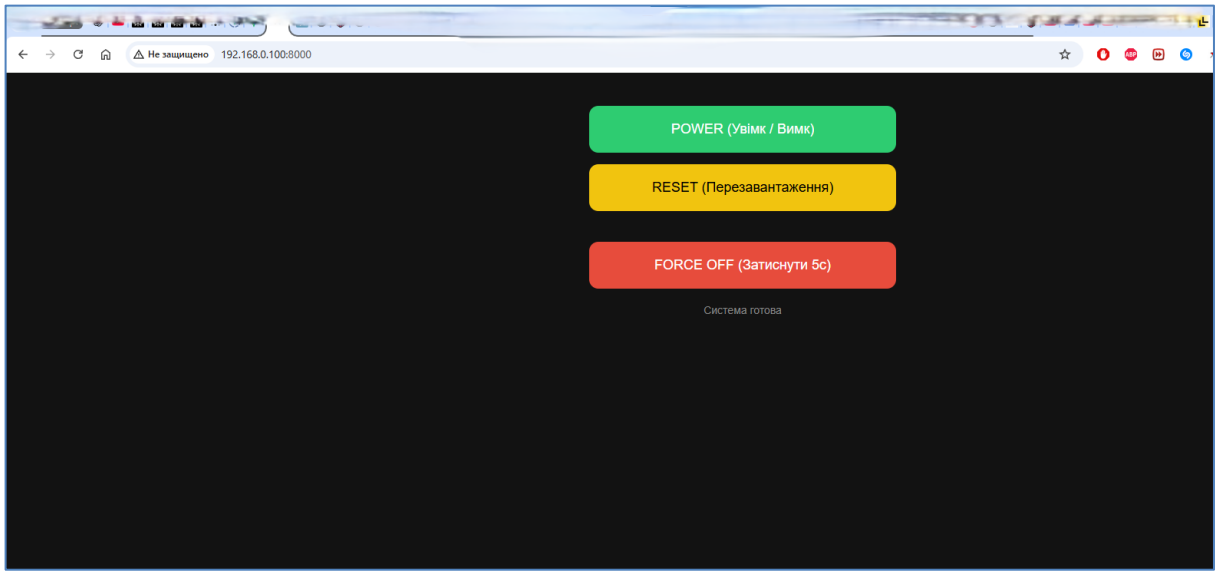


Рисунок 3.19 – Інтерфейс керування в десктопному браузері

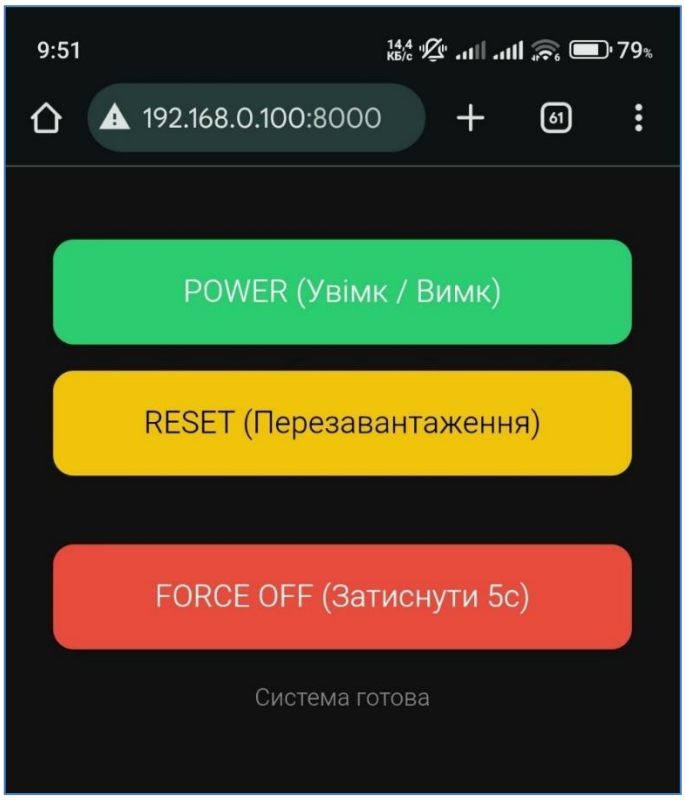


Рисунок 3.20 – Інтерфейс керування в мобільному браузері

Тепер необхідно провести перевірку функціоналу.

При натисканні кнопки «Power» на екрані, світлодіод на першому модулі PC817 загоряється на 0.5 секунди, а слідом починається запуск ПК.

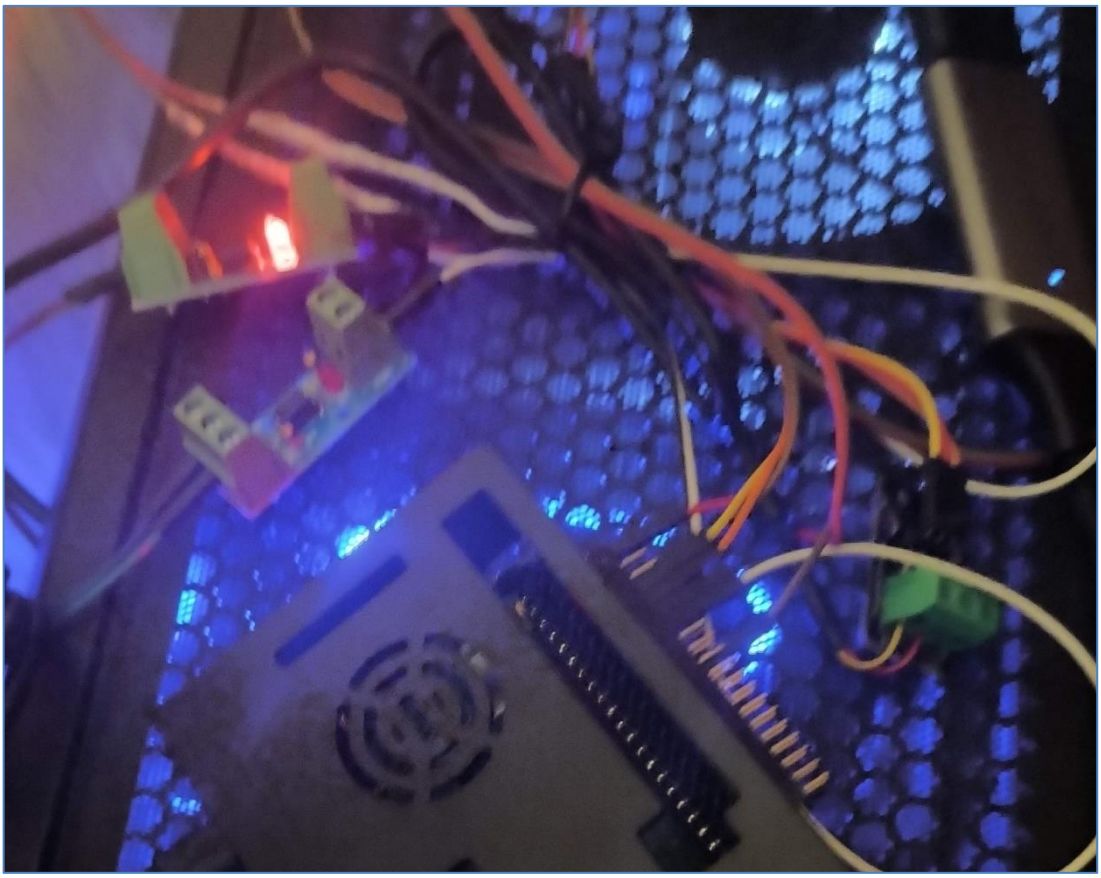


Рисунок 3.21 – Момент спрацювання першої оптопари

При натисканні кнопки «Reset», спрацьовує світлодіод на другому модулі, а ПК перезавантажується.

Кнопка «Force off» активує перший модуль на 5 секунд. При підтвердженні дії ПК вимкнеться.

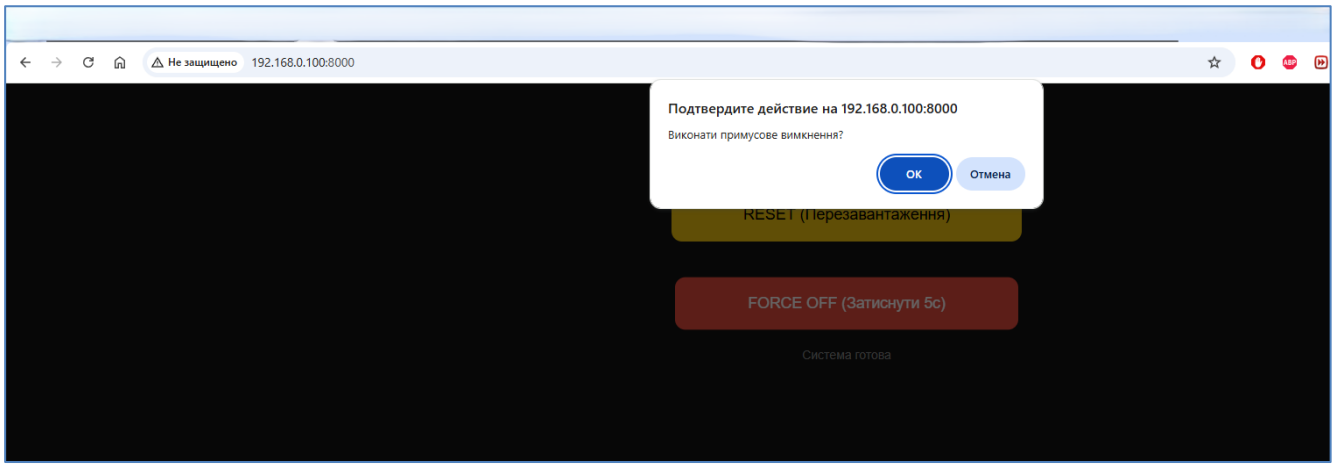


Рисунок 3.22 – Діалогове вікно для примусового вимкнення

Для зупинки сервера потрібно ввести комбінацію клавіш Ctrl+C у терміналі.

Налаштування автозапуску веб-сервера.

Оскільки Raspberry Pi може перезавантажуватись, наприклад, після зникнення світла, критично важливо, щоб веб-сервер відновлював роботу автоматично без участі користувача. Для цього було створено системну службу за допомогою systemd. В редакторі nano з правами адміністратора створено файл опису сервісу:

sudo nano /etc/systemd/system/pcremote.service

У файл було додано наступні параметри і вказано шлях до віртуального середовища та робочої папки:

```
mick@mikvm: ~/pc_remote
  GNU nano 8.4                                /etc/systemd/system/pcremote.service
[Unit]
Description=PC Remote Control Web Server
After=network.target

[Service]
# Вказано користувача mick, щоб уникнути проблем з доступом до файлів
User=mick
WorkingDirectory=/home/mick/pc_remote
# Запущено з віртуального середовища (venv)
ExecStart=/home/mick/pc_remote/venv/bin/python main.py
# Автоматичний перезапуск сервера
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Рисунок 3.23 – Зміст служби автозапуску

Після збереження файлу було виконано ряд команд, щоб активувати новий сервіс [45]:

sudo systemctl daemon-reload

Ця команда оновлює список сервісів, щоб система побачила новий файл.

sudo systemctl enable pcremote.service

Ця команда додає сервіс в автозавантаження. Тепер він буде стартувати разом із завантаженням Raspberry Pi.

sudo systemctl start pcremote.service

Ця команда запускає сервер прямо зараз, щоб можна було перевірити його роботу.

Після завершення всіх налаштувань можна перейти до фінального тестування. Ввівши в браузері адресу http://192.168.0.100:8000, можна отримати доступ до веб-інтерфейсу. Натискання кнопок на екрані успішно імітує фізичне натискання кнопок на корпусі ПК, дозволяючи вмикати та перезавантажувати комп'ютер дистанційно, перебуваючи в тій самій локальній мережі.

3.4 Організація захищеного віддаленого доступу до системи IP-KVM

Фінальним завданням при розробці IP-KVM є забезпечення доступу до пристрою з зовнішньої мережі. Оскільки Raspberry Pi знаходиться за NAT-маршрутизатором у локальній мережі, прямий доступ за IP-адресою ускладнений і небезпечний. Використання переадресації портів (Port Forwarding) створює вразливості в системі безпеки, відкриваючи інтерфейси керування для сканування зловмисниками.

Вирішенням цієї проблеми стала технологія Mesh VPN на базі Tailscale. Це рішення дозволяє створити захищену віртуальну приватну мережу поверх

публічного інтернету, об'єднуючи керуючий пристрій (ноутбук або смартфон) та виконавчий пристрій (Raspberry Pi) в єдиний зашифрований простір.

Основою безпеки Tailscale є використання сучасного протоколу WireGuard, який забезпечує високу швидкість передачі даних та криптографічну стійкість. Ключовим аспектом захисту є наскрізне шифрування (End-to-End Encryption), завдяки якому весь трафік шифрується на пристрої відправника і розшифровується виключно отримувачем, унеможливлюючи доступ серверів координації до ключів шифрування. Важливою перевагою є можливість роботи без відкритих портів завдяки технології NAT traversal, яка встановлює прямі P2P-з'єднання через складні мережеві екрани, залишаючи периметр закритим для зовнішніх сканувань. Крім того, система використовує автентифікацію через SSO (Single Sign-On) за допомогою надійних провайдерів ідентифікації Google, Microsoft, GitHub, що дозволяє інтегрувати багатофакторну автентифікацію MFA для захисту доступу до IP-KVM.

Інсталяція та налаштування VPN-клієнта на серверній частині.

Першим кроком реалізації мережевої підсистеми є встановлення клієнта Tailscale на Raspberry Pi 4B. Для цього використовується офіційний скрипт інсталяції. У терміналі Raspberry Pi було виконано команду для завантаження інсталяційного скрипта та його виконання:

curl -fsSL https://tailscale.com/install.sh | sh

Після завершення встановлення пакетів необхідно запустити службу демона Tailscale та авторизувати пристрій:

sudo tailscale up

Після виконання команди up у терміналі було згенеровано унікальне URL-посилання. Це посилання було скопійовано, відкрито у веб-браузері та проведена авторизація через обліковий запис провайдера ідентифікації Google, Microsoft або GitHub. Після успішного входу Raspberry Pi було додано до списку пристроїв віртуальної мережі.

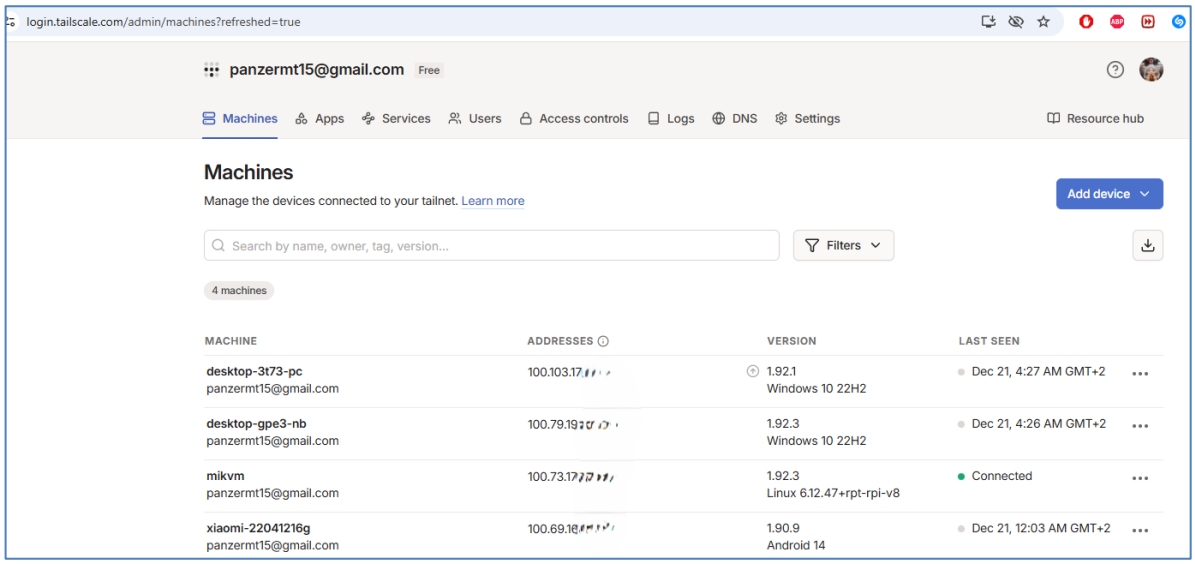


Рисунок 3.24 – Список пристроїв Tailscale

Оскільки IP-KVM є пристроєм, який повинен працювати автономно 24/7, критично важливо забезпечити автоматичний старт VPN-служби після можливих перезавантажень, наприклад, після зникнення живлення. Для цього використовується система ініціалізації systemd. Виконання команди, яка додає службу в автозавантаження та негайно активує її:

sudo systemctl enable --now tailscaled

Для коректної роботи у VPN-мережі серверні додатки заздалегідь були налаштовані на прослуховування всіх доступних мережевих інтерфейсів 0.0.0.0, а не лише локального 127.0.0.1 або localhost [46]. Це забезпечило доступність сервісів для авторизованих клієнтів через віртуальний інтерфейс tailscale0.

Отримання доступу з клієнтського пристрою.

Після налаштування серверної частини керування системою здійснюється наступним чином:

1) На клієнтський ноутбук, планшет або смартфон встановлюється офіційний додаток Tailscale.
2) Далі йде авторизація під тим самим обліковим записом, що і на Raspberry Pi.
3) У додатку Tailscale відображається список активних пристроїв та їх IP-адреси у віртуальній мережі.

Для доступу до інтерфейсів IP-KVM використовуються наступні URL-адреси:

- Відеотрансляція: http://100.x.y.z:8080/stream
- Панель керування живленням: http://100.x.y.z:8000

Завдяки службі MagicDNS, яка автоматично реєструє доменні імена для пристроїв у мережі, доступ до сервісів може здійснюватися за ім'ям пристрою замість його IP-адреси.

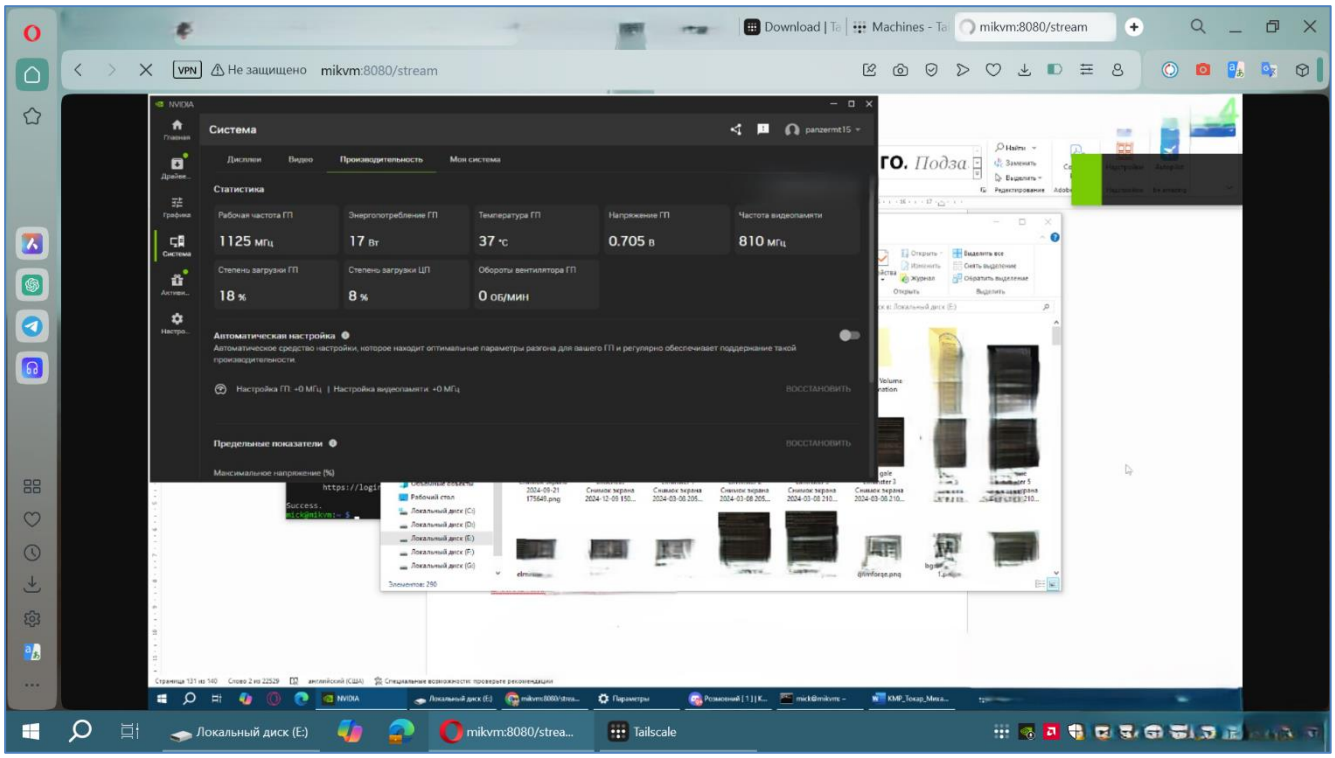


Рисунок 3.25 – Доступ до відеотрансляції через Tailscale

Використання Tailscale повністю шифрує трафік між клієнтом та сервером (end-to-end encryption), а сама система залишається невидимою для сканерів портів у публічному інтернеті.

3.5 Аналіз часових затримок системи

Ефективність роботи розробленого IP-KVM визначається сумарною затримкою між дією оператора та візуальною або апаратною реакцією системи. Для

комплексної оцінки швидкодії було проведено експериментальне дослідження за чотирма ключовими показниками, результати якого наведено нижче.

Для визначення наскрізної затримки відеосигналу «Glass-to-Glass» застосовувався метод синхронної фотофіксації екранів джерела та клієнта, на яких відображався мілісекундний таймер. Експериментально встановлено, що при роботі у локальній мережі середня затримка становить 100–200 мс. При використанні захищеного VPN-каналу Tailscale цей показник зростає до 150–250 мс через накладні витрати на шифрування трафіку. Отримані значення є цілком прийнятними для виконання типових адміністративних завдань, налаштування BIOS та роботи зі статичними інтерфейсами, проте затримка стає відчутною при передачі динамічного контенту. Результати ще раз доводять, що для комфортного виконання віддалених робіт за ПК необхідний апгрейд апаратної частини, а саме, заміна карти відеозахоплення на більш потужну й досконалу.

Аналіз мережевої складової проводився з використанням діагностичного інструментарію tailscale ping, що дозволило класифікувати тип з'єднання. У режимі прямого з'єднання (Direct/P2P) час проходження пакету (RTT) знаходиться в межах 5–50 мс, що є оптимальним показником для стабільної трансляції. У випадках, коли пряме з'єднання неможливе і трафік маршрутизується через сервери-ретранслятори (Relay/DERP), затримка зростає до 70–200 мс, а це сильно впливає на плавність відеопотоку. Таким чином, для коректної експлуатації системи необхідною умовою є забезпечення прямого з'єднання між клієнтом та сервером.

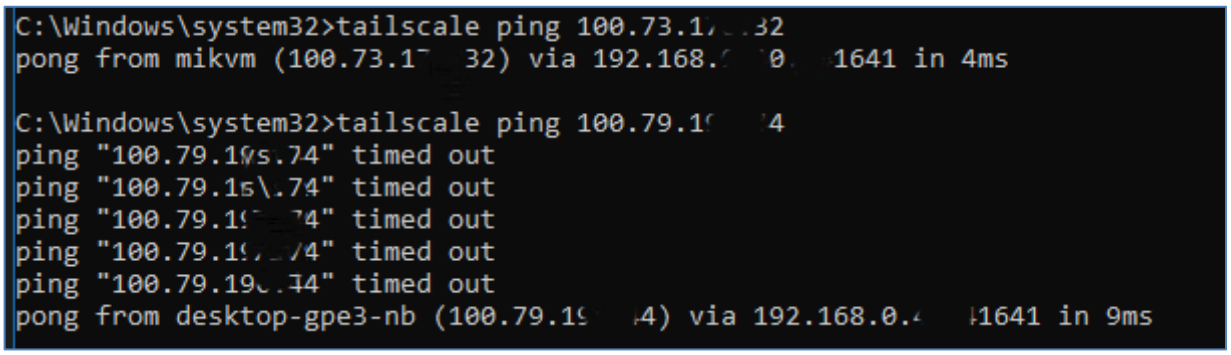


Рисунок 3.26 – Результат тестування затримки tailscale ping

Вимірювання часу реакції підсистеми керування живленням через GPIO показало, що інтервал між натисканням кнопки у веб-інтерфейсі та фізичним спрацюванням оптопари на платі Raspberry Pi становить від 30 до 60 мс. Враховуючи, що ця операція дискретна, така затримка фактично є непомітною і забезпечує миттєву реакцію системи на виконання команд увімкнення, вимкнення або перезавантаження цільового комп'ютера.

Дослідження підсистеми емуляції USB-пристроїв, реалізованої через Python-скрипти, виявило різний вплив затримок на роботу клавіатури та миші. Для клавіатури затримка введення становить 30–50 мс, що є достатнім для комфортного набору тексту в терміналі. У випадку з комп'ютерною мишею затримка коливається в межах 50–100 мс. Це створює ефект незначної інерційності курсора, який не перешкоджає роботі з графічним інтерфейсом операційної системи, проте унеможливлює використання IP-KVM для виконання високодинамічних операцій.

У висновку, сумарна затримка між дією та її результатом становить 150–200 мс. Основний вклад до неї вносить затримка відеопотоку, тоді як затримка інших дій майже непомітна. Я вважаю, що в майбутньому вкрай необхідно модернізувати систему трансляції картинки шляхом заміни карти відеозахоплення на екземпляр більш високого цінового сегменту.

РОЗДІЛ 4. ПЕРСПЕКТИВИ ПОДАЛЬШОГО РОЗВИТКУ

Не дивлячись на те, що розроблений у рамках дипломної роботи прототип IP-KVM на базі Raspberry Pi 4B успішно впорався з поставленими задачами, практичне тестування виявило низку нюансів, які варто врахувати в майбутньому. Аналіз роботи системи в реальних умовах показав, що для перетворення цього прототипу на повноцінний, професійний інструмент необхідно зосередитися не лише на програмних налаштуваннях, а й на вдосконаленні апаратної частини. У цьому розділі буде розглянуто конкретні кроки, які дозволять зробити систему надійнішою, швидшою та зручнішою.

Еволюція апаратної платформи та якості зображення.

Найбільш очевидним напрямком розвитку є оновлення ядра системи. Перехід на Raspberry Pi 5 виглядає цілком логічним кроком, і справа тут не лише у збільшенні обчислювальної потужності. Головна перевага п'ятої версії для мого проєкту – це архітектурні зміни, зокрема винесення контролера периферії в окремий чіп RP1. Це дозволить уникнути тих конфліктів переривань USB, з якими доводиться стикатися на четвертій моделі, та забезпечить стабільнішу роботу моєї системи під навантаженням.

Іншим вузьким місцем поточної реалізації є якість та затримка відеопотоку. Використання бюджетної карти захоплення на чіпі AM8352 хоч і є економічно виправданим для базових задач, але вносить помітну затримку через MJPEG-стиснення. Для комфортної віддаленої роботи, коли кожна мілісекунда затримки відчувається занадто сильно, варто розглянути альтернативні шляхи.

Модернізація USB-тракту на MacroSilicon MS2130 до USB 3.0 та формату YUV422 дозволяє передавати картинку майже без стиснення. Це дає можливість отримати плавні 60 кадрів на секунду при 1080p, що робить віддалену роботу візуально непомітною від локальної.

Перехід на інтерфейс CSI з чіпом Toshiba TC358743. Це рішення професійного рівня інтеграції, реалізоване на деяких комерційних системах. Використовуючи інтерфейс камери CSI-2 замість USB, можна подавати відеопотік безпосередньо в графічне ядро Raspberry Pi. Такий підхід дозволяє знизити затримку до набагато кращих показників – менше 100 мс, що є критично важливим для задач, які вимагають миттєвої реакції.

Апаратна емуляція KVM замість програмних скриптів.

На даному етапі реалізована програмна емуляція клавіатури та миші, покладаючись на скрипти у Windows 10. Це працює, поки працює сама операційна система. Але коли ОС зависла або система знаходиться в меню BIOS будь-які керуючі дії стають неможливими. Під час роботи над проєктом спроби підключити Raspberry Pi напряму як USB-гаджет викликали критичні помилки Windows BSOD. Це пов'язано не лише з конфліктами драйверів, а й з станом самої операційної системи: стара ОС на моєму ПК, перенасичена системними помилками та програмними конфліктами, стала значно чутливіша до підключення нестандартних пристроїв. Конкретно в моєму випадку ОС може видати синій екран при запуску утиліти для керування мишею, або при підключенні флешок до деяких роз’ємів.

Вирішенням цієї проблеми є реалізація True Hardware KVM через Linux USB Gadget API (ConfigFS). Налаштувавши Raspberry Pi так, щоб він прикидався звичайною апаратною клавіатурою та мишкою на рівні заліза, буде здобута низка переваг:

- Повний контроль над комп'ютером ще до завантаження ОС (в BIOS/UEFI).
- Відсутність необхідності ставити будь-який софт чи писати скрипти на цільовий ПК.
- Стабільність роботи – правильно налаштовані дескриптори пристрою (VID/PID) змусять Windows бачити стандартну периферію, що мінімізує ризик збоїв навіть на застарілих та перевантажених системах, усуваючи основну причину синіх екранів.

Clear CMOS як останній аргумент адміністратора.

Поточна система вміє натискати кнопки Power та Reset. Це необхідний мінімум, але будь-який системний адміністратор знає ситуацію, коли цього недостатньо, а комп'ютер циклічно перезавантажується і не пускає в BIOS.

Ідея полягає в тому, щоб інтегрувати в систему ще одну оптопару PC817, підключену безпосередньо до джампера Clear CMOS на материнській платі. Це дасть можливість фізично скинути налаштування BIOS до заводських віддалено, одним кліком у веб-інтерфейсі. Така проста модифікація перетворить KVM-систему на універсальний інструмент для діагностики та налаштування віддаленої робочої станції.

Інтеграція в екосистему Розумного дому.

Враховуючи цілодобовий режим роботи апаратної платформи Raspberry Pi 4B, я вважаю доцільною реалізацію системи розумного дому на її основі. Технічні характеристики мікрокомп'ютера дозволяють паралельно розгорнути повноцінний сервер розумного дому Home Assistant у середовищі контейнеризації Docker [47, 48].

Інтеграція підсистеми IP-KVM з екосистемою розумного дому може бути реалізована за допомогою протоколу MQTT. Це дозволить створити єдиний інформаційний простір, де події однієї системи ініціюють реакції іншої. Наприклад, при віддаленому вимкненні робочої станції через інтерфейс KVM, контролер передає відповідний сигнал до Home Assistant, який, згідно із заданим сценарієм, автоматично вимикає освітлення у приміщенні та переводить кліматичну техніку в енергоощадний режим. Такий підхід забезпечує централізоване керування та автоматизацію інфраструктури.

Також з метою покращення ергономічних показників робочого місця, пропонується імплементація системи адаптивного освітлення [48]. Шляхом підключення цифрового датчика освітленості, наприклад, BH1750 до інтерфейсу

I2C, стає можливим реалізація алгоритму динамічного керування яскравістю підключених до системи розумного дому джерел світла.

Система автоматично підтримуватиме нормативний рівень освітленості, наприклад, 500 люкс згідно з європейськими стандартами, компенсуючи зміни природного освітлення. Це рішення не лише розширює функціональність системи автоматизації та забезпечує дотримання вимог норм охорони праці, але й сприяє збереженню здоров'я користувача та підвищенню продуктивності праці.

Висновки до розділу.

Узагальнюючи вищевикладене, можна сказати, що реалізація запропонованих напрямків модернізації дозволить трансформувати проєкт з вузькоспеціалізованого засобу віддаленого доступу у комплексний центр керування інфраструктурою робочого місця. Впровадження сучасних інтерфейсів відеозахоплення, перехід до апаратної емуляції периферійних пристроїв, глибока інтеграція з екосистемою розумного дому та створення повноцінної системи безпеки суттєво підвищать надійність, функціональність та ергономіку розробленої системи [50].

ВИСНОВКИ

Під час виконання даної дипломної роботи було вирішено актуальне науково-прикладне завдання створення автономної платформи віддаленого керування робочим місцем на базі одноплатного комп'ютера Raspberry Pi. Розроблена система є відповіддю на сучасні виклики у сфері системного адміністрування, де критично важливим є забезпечення безперервності бізнес-процесів та мінімізація часу простою обладнання. Проведене дослідження, проєктування та практична реалізація прототипу дозволили сформулювати наступні розгорнуті висновки.

Детальне вивчення існуючих комерційних рішень класу IP-KVM, зокрема, продукції Lantronix та ATEN, показало, що хоча вони і забезпечують високий рівень надійності, їхня закрита архітектура суттєво обмежує можливості кастомізації та інтеграції в нестандартні інфраструктури. Висока вартість таких рішень робить їх масове впровадження для індивідуальних робочих місць або малого бізнесу економічно недоцільним. Натомість, розробка власної системи на базі відкритої апаратної платформи Raspberry Pi та Open Source програмного забезпечення дозволила знизити вартість кінцевого виробу у 5-7 разів, забезпечивши при цьому гнучкість налаштувань та можливість програмного розширення функціоналу.

В ході роботи було спроєктовано та зібрано діючий прототип на базі мікрокомп'ютера Raspberry Pi 4B. Вибір чіпа відеозахоплення AM8352 став оптимальним компромісом між ціною та якістю, забезпечивши передачу зображення з роздільною здатністю 720p, що є достатнім для виконання більшості адміністративних задач. Особливу увагу було приділено підсистемі керування живленням: реалізована схема через GPIO-інтерфейс із використанням оптопар забезпечує гальванічну розв'язку, захищаючи керуючий контролер від електричних завад цільового ПК. За допомогою неї було реалізовано функцію примусового перезавантаження Hard Reset, емулюючи фізичне замикання контактів Power та

Reset, яка єдиним дієвим методом відновлення роботи при зависанні операційної системи цільового ПК.

Порівняльний аналіз методів трансляції відеопотоку виявив суттєві відмінності між технологіями. Використання формату MJPEG через сервіс uStreamer продемонструвало найвищу стабільність та сумісність з будь-якими веб-браузерами, проте цей метод характеризується значною затримкою 200-300 мс та високим навантаженням на смугу пропускання через передачу повних кадрів. Для роботи з динамічним контентом більш ефективним визнано використання технології WebRTC, яка завдяки алгоритмам міжкадрового стиснення дозволяє зменшити затримку до рівня менше 100 мс, забезпечуючи комфортну взаємодію в реальному часі, хоча і потребує більших обчислювальних ресурсів з боку клієнта та сервера.

Майбутня реалізація захоплення HID-пристроїв через USB OTG інтерфейс Raspberry Pi це перспективний напрямок для отримання повного контролю над системою на рівні BIOS/UEFI, де програмні засоби віддаленого доступу RDP і VNC є недієздатними. Разом з тим, в ході тестування було виявлено проблеми сумісності зі старою Windows 10, які проявлялися у вигляді критичних системних помилок BSOD при прямому підключенні. Через це у поточній реалізації тимчасово застосовуються Python-скрипти на стороні цільової системи для трансляції команд введення. Стратегічним завданням подальшої модернізації визначено повну заміну цих скриптів на нативну апаратну емуляцію через тонке налаштування дескрипторів USB ConfigFS. Це забезпечить коректне розпізнавання пристрою як стандартної композитної HID-периферії, усуне залежність від програмних агентів в ОС та ліквідує конфлікти драйверів.

В ході досліджень було встановлено, що відкритий доступ до веб-інтерфейсу керування KVM несе критичні ризики, оскільки надає повний контроль над фізичним обладнанням. Для нівелювання загроз було впроваджено технологію Tailscale mesh-VPN на базі WireGuard замість прямого прокидання портів. Такий підхід не лише гарантує наскрізне шифрування та конфіденційність даних, але й

дозволяє обходити обмеження NAT без необхідності оренди публічної IP-адреси, забезпечуючи безпечний доступ до системи з будь-якої точки світу.

Глибокий аналіз результатів тестування дозволив сформувати дорожню карту подальшого вдосконалення системи. Перехід на Raspberry Pi 5 відкриє доступ до інтерфейсу PCIe та дозволить створення більш комплексних систем. Інтеграція функції Clear CMOS через додатковий канал керування реле/оптопарою надасть можливість віддаленого скидання налаштувань BIOS, яка перетворить систему на повноцінний інструмент для віддаленого керування.

Об'єднання розробленого рішення з платформою Home Assistant через протокол MQTT дозволить створити єдине середовище керування робочим місцем. Воно буде включати не лише контроль живлення ПК, а й реалізацію сценаріїв адаптивного освітлення та клімат-контролю, що відповідає сучасним тенденціям енергоефективності та концепції IoT.

Розроблена автономна платформа повністю відповідає поставленим вимогам щодо функціональності, надійності та економічної ефективності. Вона забезпечує незалежний від стану операційної системи доступ до цільового комп'ютера, дозволяє проводити повний цикл обслуговування – від налаштування BIOS до перевстановлення ОС. Отримані результати підтверджують перспективність використання одноплатних комп'ютерів як основи для побудови професійних інструментів системного адміністрування, що за своїми характеристиками наближаються до промислових стандартів.

ПЕРЕЛІК ПОСИЛАНЬ:

1. GeeksforGeeks. Introduction to Internet of Things - IOT - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/computer-networks/introduction-to-internet-of-things-iot-set-1/ (date of access: 25.12.2025).

2. Internet of Things. *ITU*. URL: https://www.itu.int/en/ITU-T/techwatch/Pages/internetofthings.aspx (date of access: 25.12.2025).

3. Atzori L., Iera A., Morabito G. The Internet of Things: A survey. *Computer Networks*. 2010. Vol. 54, no. 15. P. 2787–2805. URL: https://doi.org/10.1016/j.comnet.2010.05.010 (date of access: 25.12.2025).

4. Özdemir V., Hekim N. Birth of Industry 5.0: Making Sense of Big Data with Artificial Intelligence, "The Internet of Things" and Next-Generation Technology Policy. *OMICS: A Journal of Integrative Biology*. 2018. Vol. 22, no. 1. P. 65–76. URL: https://doi.org/10.1089/omi.2017.0194 (date of access: 25.12.2025).

5. IoT connections worldwide 2034| Statista. *Statista*. URL: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (date of access: 25.12.2025).

6. James Howell. An Introduction To IoT Architecture. *101 Blockchains*. URL: https://101blockchains.com/iot-architecture/ (date of access: 25.12.2025).

7. B C B., C T V. G., B H R. A Survey: Internet of Things (IOT) Technologies, Applications. *International Journal for Research in Applied Science and Engineering Technology*. 2022. Vol. 10, no. 8. P. 640–644. URL: https://doi.org/10.22214/ijraset.2022.42595 (date of access: 25.12.2025).

8. What are the benefits of remote control technology? - Gds Lighting. *Gds Lighting*. URL: https://www.gdslighting.com/en/what-are-the-benefits-of-remote-control-technology/ (date of access: 25.12.2025).

9. Industry 4.0. *IT-Enterprise – цифрова трансформація бізнес-процесів, ERP*| *it.ua*. URL: https://www.it.ua/knowledge-base/technology-innovation/industry-4 (date of access: 25.12.2025).

10. A Review of the Industry 4.0 to 5.0 Transition: Exploring the Intersection, Challenges, and Opportunities of Technology and Human–Machine Collaboration / M. T. Islam et al. *Machines*. 2025. Vol. 13, no. 4. P. 267. URL: https://doi.org/10.3390/machines13040267 (date of access: 25.12.2025).

11. GeeksforGeeks. What is a Programmable Logic Controllers (PLC) | Working and Structure - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/electrical-engineering/what-is-a-programmable-logic-controllers-plc/ (date of access: 25.12.2025).

12. What is a DCS?. *Schneider Electric Blog*. URL: https://blog.se.com/industry/2024/01/30/what-is-a-dcs/ (date of access: 25.12.2025).

13. YEVSEIENKO O., KACHANOV P. SHOPPING MALL PREMISES SCADA-MICROCLIMATE CONTROL SYSTEM DEVELOPMENT. *Herald of Khmelnytskyi National University. Technical sciences*. 2022. Vol. 309, no. 3. P. 168–176. URL: https://doi.org/10.31891/2307-5732-2022-309-3-168-176 (date of access: 25.12.2025).

14. Mishra S. Our Best Picks for Remote Desktop Tools (2026). *Global IT Hardware Management | Workwize*. URL: https://www.goworkwize.com/blog/remote-desktop-tools (date of access: 25.12.2025).

15. Підключення до Raspberry Pi через VNC. IT Master. URL: https://itmaster.biz.ua/electronics/raspberry-pi/raspberrypi-vnc.html (дата звернення: 16.10.2025).

16. Nianrong Wang. Research on Network Security Policies for KVM-based Virtualization Platforms. *CIBDA '24: Proceedings of the 5th International Conference on Computer Information and Big Data Applications*. 2024. No. 5. URL: https://doi.org/10.1145/3671151.3671170 (date of access: 20.12.2025).

17. Raspberry Pi OS. *Raspberry Pi*. URL: https://www.raspberrypi.com/documentation/computers/os.html#introduction (date of access: 20.12.2025).

18. Налаштування Raspberry Pi для віддаленого доступу. Edovia Support. URL: https://help.edovia.com/uk/screens-4/getting-started/configure-raspberry-pi (дата звернення: 15.10.2025).

19. Олександр Тартачний. 8 пристроїв на основі Raspberry Pi. robot dreams. URL: https://robotdreams.cc/uk/blog/52-8-ustroystv-na-osnove-raspberry-pi (дата звернення: 16.10.2025).

20. NEWS P. Raspberry Pi отримує вбудований інструмент віддаленого доступу – Raspberry Pi Connect. ProIT: медіа для профі в IT. URL: https://proit.ua/raspberry-pi-otrimuie-vbudovanii-instrumient-viddalienogho-dostupu-raspberry-pi-connect-2/ (дата звернення: 16.10.2025).

21. ATEN. What is a KVM switch?. *KVM, Professional AV, Power Distribution Unit, Control System by ATEN | ATEN Belgium - Deutsch*. URL: https://www.aten.com/global/en/resources/feature-articles/what-is-a-kvm-switch/ (date of access: 25.12.2025).

22. Implementation of an Embedded System into the Internet of Robotic Things / J. Krejčí та ін. Micromachines. 2022. Т. 14, № 1. С. 113. URL: https://doi.org/10.3390/mi14010113 (дата звернення: 20.10.2025).

23. Domski W. Remote Laboratory Offered as Hardware-as-a-Service Infrastructure. Electronics. 2022. Т. 11, № 10. С. 1568. URL: https://doi.org/10.3390/electronics11101568 (дата звернення: 20.10.2025).

24. Development and implementation of a raspberry Pi-based IoT system for real-time performance monitoring of an instrumented tractor / V. Mahore та ін. Smart Agricultural Technology. 2024. Т. 9. С. 100530. URL: https://doi.org/10.1016/j.atech.2024.100530 (дата звернення: 20.10.2025).

25. A comprehensive review on applications of Raspberry Pi / S. E. Mathe та ін. Computer Science Review. 2024. Т. 52. С. 100636. URL: https://doi.org/10.1016/j.cosrev.2024.100636 (дата звернення: 20.10.2025).

26. Remote Machine Management via Raspberry Pi and the Internet of Things. International Research Journal of Modernization in Engineering Technology and Science. 2024. URL: https://doi.org/10.56726/irjmets53141 (дата звернення: 22.10.2025).

27. Raspberry Pi 4 Tech Specs. *Raspberry Pi*. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (date of access: 20.12.2025).

28. Raspberry Pi DS18B20 Temperature Sensor Tutorial. *Circuit Basics*. URL: https://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/ (date of access: 25.12.2025).

29. GitHub - steve-m/hsdaoh: High Speed Data Acquisition over HDMI - Userspace library. *GitHub*. URL: https://github.com/steve-m/hsdaoh (date of access: 25.12.2025).

30. PC817 IC Optocoupler Pinout, Circuit, Datasheet, and Uses - Jotrin Electronics. *Jotrin Electronics - Global Electronic Components Distributor*. URL: https://www.jotrin.com/technology/details/pc817-ic-optocoupler-pinout-circuit-datasheet-uses (date of access: 25.12.2025).

31. Domski W. Remote Laboratory Offered as Hardware-as-a-Service Infrastructure. Electronics. 2022. T. 11, № 10. C. 1568. URL: https://doi.org/10.3390/electronics11101568 (дата звернення: 22.10.2025).

32. Hobbs T., Ali A. Smart Home Control Using Real-Time Hand Gesture Recognition and Artificial Intelligence on Raspberry Pi 5. Electronics. 2025. T. 14, № 20. C. 3976. URL: https://doi.org/10.3390/electronics14203976 (дата звернення: 25.10.2025).

33. Lukito C., Lukito R. B., Ernawati E. Implementation of IoT Edge Computing for Control and Monitoring System of Hydroponic Plant Water Quality Using Raspberry Pi. International Journal of Computer Science and Humanitarian AI. 2024. T. 1, № 1. C. xx. URL: https://doi.org/10.21512/ijcshai.v1i1.12153 (дата звернення: 25.10.2025).

34. Yaldaie A., Porras J., Drögehorn O. Innovative Home Automation with Raspberry Pi: A Comprehensive Approach to Managing Smart Devices. Asian Journal of Computer Science and Technology. 2024. T. 13, № 1. C. 27–40. URL: https://doi.org/10.70112/ajcst-2024.13.1.4260 (дата звернення: 25.10.2025).

35. Comprehensive Security for IoT Devices with Kubernetes and Raspberry Pi Cluster / I.-C. Donca та ін. Electronics. 2024. Т. 13, № 9. С. 1613. URL: https://doi.org/10.3390/electronics13091613 (дата звернення: 25.10.2025).

36. Basic concepts and taxonomy of dependable and secure computing / A. Avizienis та ін. IEEE Transactions on Dependable and Secure Computing. 2004. Т. 1, № 1. С. 11–33. URL: https://doi.org/10.1109/tdsc.2004.2 (дата звернення: 25.10.2025).

37. Security, privacy and trust in Internet of Things: The road ahead / S. Sicari та ін. Computer Networks. 2015. Т. 76. С. 146–164. URL: https://doi.org/10.1016/j.comnet.2014.11.008 (дата звернення: 26.10.2025).

38. Tailscale VPN - Documentation. *Home - Documentation*. URL: https://docs.rockylinux.org/10/uk/guides/security/tailscale_vpn/ (date of access: 20.12.2025).

39. Access a remote terminal with SSH. *Raspberry Pi*. URL: https://www.raspberrypi.com/documentation/computers/remote-access.html#ssh (date of access: 16.12.2025).

40. What is config.txt?. *Raspberry Pi*. URL: https://www.raspberrypi.com/documentation/computers/config_txt.html#what-is-config-txt (date of access: 18.12.2025).

41. Watchdog on Raspberry Pi (Or other autoreboot methods?). *Reddit*. URL: https://www.reddit.com/r/raspberry_pi/comments/52szl9/watchdog_on_raspberry_pi_or_other_autoreboot/?tl=ru (date of access: 18.12.2025).

42. GitHub - pikvm/ustreamer: µStreamer - Lightweight and fast MJPEG-HTTP streamer. *GitHub*. URL: https://github.com/pikvm/ustreamer (date of access: 19.12.2025).

43. Writing my own KVM client in python – Made by Mikal. *Made by Mikal*. URL: https://www.madebymikal.com/writing-my-own-kvm-client-in-python/ (date of access: 20.12.2025).

44. FastAPI та Python: створення ефективного RESTful API з нуля. *Epam*. URL: https://careers.epam.ua/blog/efficient-restful-api-from-scratch (дата звернення: 15.12.2025).

45. Ellingwood J. Manage Systemd Services with systemctl on Linux. *DigitalOcean: AI-Powered Unified Agentic Cloud Infrastructure*. URL: https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units (date of access: 20.12.2025).

46. GeeksforGeeks. What is Local Host? - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/computer-networks/what-is-local-host/ (date of access: 20.12.2025).

47. Posashkov O., Tsymbal O. Development of the architecture of the remote access system to educational laboratory equipment using automated solutions. INNOVATIVE TECHNOLOGIES AND SCIENTIFIC SOLUTIONS FOR INDUSTRIES. 2024. № 3 (29). С. 64–75. URL: https://doi.org/10.30837/2522-9818.2024.3.064 (дата звернення: 25.10.2025).

48. Real time optimal schedule controller for home energy management system using new binary backtracking search algorithm / M. S. Ahmed та ін. Energy and Buildings. 2017. Т. 138. С. 215–227. URL: https://doi.org/10.1016/j.enbuild.2016.12.052 (дата звернення: 25.10.2025).

49. Ben A., Adly E., Hasannah D. Internet Of Things (Iot) Light Control System Using a Mobile-Based Raspberry Pi. IAIC Transactions on Sustainable Digital Innovation (ITSDI). 2020. Т. 2, № 1. С. 46–53. URL: https://doi.org/10.34306/itsdi.v2i1.351 (дата звернення: 22.10.2025).

50. Roman R., Zhou J., Lopez J. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*. 2013. Т. 57, № 10. С. 2266–2279. URL: https://doi.org/10.1016/j.comnet.2012.12.018 (дата звернення: 25.10.2025).

ДОДАТОК А

Повний код python-скрипту для захоплення і трансляції сигналів миші та клавіатури

```
import socket # мережева взаємодія, відпрака пакетів
import json
import threading # багатопоточність
import sys
import time
import ctypes # отримання розмірів екрана
from pynput import mouse, keyboard

# --- НАЛАШТУВАННЯ ---
SERVER_PORT = 6000
BUFFER_SIZE = 4096
MOUSE_SENSITIVITY = 1.5

# Клавіша для перемикання керування тільки для Клієнта
TOGGLE_KEY = keyboard.Key.ctrl_r
# Клавіша для аварійної зупинки на обох ПК
EXIT_KEY = keyboard.Key.f12

class KVMDaemon:
    def __init__(self):
        self.running = True
        self.is_capturing = False
        self.sock = None
        self.mouse_ctrl = mouse.Controller()
        self.keyboard_ctrl = keyboard.Controller()

        # Отримуємо розміри екрана для центрування
        try:
            user32 = ctypes.windll.user32
            self.center_x = user32.GetSystemMetrics(0) // 2
            self.center_y = user32.GetSystemMetrics(1) // 2
        except:
            # Fallback якщо не Windows або помилка
            self.center_x = 960
            self.center_y = 540

    def serialize_key(self, key):
        """Перетворює об'єкт клавіші у рядок для відправки"""
        try:
            if hasattr(key, 'char') and key.char:
                return {"type": "char", "value": key.char}
            else:
                return {"type": "special", "value": str(key)}
        except Exception:
            return {"type": "special", "value": str(key)}

    def deserialize_key(self, key_data):
        """Відновлює об'єкт клавіші з отриманих даних"""
        try:
            if key_data["type"] == "char":
```

```
    def deserialize_key(self, key_data):
        """Відновлює об'єкт клавіші з отриманих даних"""
        try:
            if key_data["type"] == "char":
                return key_data["value"]
            elif key_data["type"] == "special":
                key_str = key_data["value"]
                clean_name = key_str.replace("Key.", "")
                return getattr(keyboard.Key, clean_name, None)
        except:
            return None
        return None

    # --- СЕРВЕРНА ЧАСТИНА ---
    def start_server(self):
        print(f"[*] [DAEMON] Запуск сервера на порту {SERVER_PORT}...")
        print(f"[*] УВАГА: Переконайтеся, що порт {SERVER_PORT} відкритий у Брандмауері!")

        try:
            self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            self.sock.bind(("0.0.0.0", SERVER_PORT))
            self.sock.settimeout(1.0)
        except Exception as e:
            print(f"[!] Помилка запуску сервера: {e}")
            return

        print(f"[*] Очікування команд. Натисніть F12 для виходу.")

        def check_exit(key):
            if key == EXIT_KEY:
                print("Аварійна зупинка.")
                self.running = False
                return False

        listener = keyboard.Listener(on_release=check_exit)
        listener.start()

        while self.running:
            try:
                data, _ = self.sock.recvfrom(BUFFER_SIZE)
                message = json.loads(data.decode('utf-8'))
                self.execute_command(message)
            except socket.timeout:
                continue
            except json.JSONDecodeError:
                continue
            except Exception as e:
                pass
```

```
        try:
            ctype = cmd["t"]

            if ctype == "mm": # Mouse Move
                x = int(cmd["x"] * MOUSE_SENSITIVITY)
                y = int(cmd["y"] * MOUSE_SENSITIVITY)
                self.mouse_ctrl.move(x, y)

            elif ctype == "mc": # Mouse Click
                btn_name = cmd["b"].split('.')[-1]
                btn = getattr(mouse.Button, btn_name, mouse.Button.left)
                if cmd["p"]:
                    self.mouse_ctrl.press(btn)
                else:
                    self.mouse_ctrl.release(btn)

            elif ctype == "ms": # Mouse Scroll
                self.mouse_ctrl.scroll(cmd["dx"], cmd["dy"])

            elif ctype == "kp": # Key Press
                key = self.deserialize_key(cmd["k"])
                if key: self.keyboard_ctrl.press(key)

            elif ctype == "kr": # Key Release
                key = self.deserialize_key(cmd["k"])
                if key: self.keyboard_ctrl.release(key)

        except Exception:
            pass

    # --- КЛІЄНТСЬКА ЧАСТИНА ---
    def start_client(self, target_ip):
        print(f"[*] [CLIENT] Підключення до {target_ip}...")
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.target_addr = (target_ip, SERVER_PORT)

        print(f"[*] ГОТОВО. Натисніть 'Right Ctrl' для перехоплення/звільнення миші.")
        print(f"[*] При ЗАХОПЛЕННІ курсор буде зафіксовано в центрі (режим Infinite Mouse).")
        print(f"[*] Натисніть F12 для виходу.")

        # Потік опитування миші
        poll_thread = threading.Thread(target=self.mouse_poller, daemon=True)
        poll_thread.start()

        # Слухачі
        with mouse.Listener(on_click=self.on_click, on_scroll=self.on_scroll) as m_listen:
            with keyboard.Listener(on_press=self.on_press, on_release=self.on_release) as k_listen:
                k_listen.join()
                m_listen.join()
```

```
    def send_data(self, data):
        try:
            msg = json.dumps(data).encode('utf-8')
            self.sock.sendto(msg, self.target_addr)
        except:
            pass

    def mouse_poller(self):
        """Оновлена логіка з фіксацією курсора в центрі"""
        # Ставимо курсор в центр
        self.mouse_ctrl.position = (self.center_x, self.center_y)
        last_pos = (self.center_x, self.center_y)

        while self.running:
            try:
                if self.is_capturing:
                    current_pos = self.mouse_ctrl.position

                    # Рахуємо різницю від центру
                    dx = current_pos[0] - self.center_x
                    dy = current_pos[1] - self.center_y

                    if dx != 0 or dy != 0:
                        # Відправляємо зміщення
                        self.send_data({"t": "mm", "x": dx, "y": dy})

                        # Миттєво повертаємо курсор в центр. Це дозволяє рухати мишу нескінченно
                        self.mouse_ctrl.position = (self.center_x, self.center_y)

                time.sleep(0.01) # 100Hz
            except Exception:
                pass

    def on_click(self, x, y, button, pressed):
        if not self.is_capturing: return
        self.send_data({"t": "mc", "b": str(button), "p": pressed})

    def on_scroll(self, x, y, dx, dy):
        if not self.is_capturing: return
        self.send_data({"t": "ms", "dx": dx, "dy": dy})

    def on_press(self, key):
        if key == EXIT_KEY:
            self.running = False
            return False

        if key == TOGGLE_KEY:
            self.is_capturing = not self.is_capturing
```

```
        if key == TOGGLE_KEY:
            self.is_capturing = not self.is_capturing
            if self.is_capturing:
                # При включенні курсор одразу стрибає в центр, щоб не було ривків
                self.mouse_ctrl.position = (self.center_x, self.center_y)
                print("ЗАХОПЛЕНО (Курсор зафіксовано) >>")
            else:
                print("<< ЗВІЛЬНЕНО")
            return

        if not self.is_capturing: return
        self.send_data({"t": "kp", "k": self.serialize_key(key)})

    def on_release(self, key):
        if key == TOGGLE_KEY or key == EXIT_KEY: return
        if not self.is_capturing: return
        self.send_data({"t": "kr", "k": self.serialize_key(key)})

if __name__ == "__main__":
    daemon = KVMDaemon()

    if len(sys.argv) > 1:
        mode = sys.argv[1].lower()
        if mode == "server":
            daemon.start_server()
        elif mode == "client":
            if len(sys.argv) < 3:
                print("Вкажіть IP сервера!")
            else:
                daemon.start_client(sys.argv[2])
    else:
        print("--- Python LAN KVM (Fixed Mouse) ---")
        print("1. СЕРВЕР")
        print("2. КЛІЄНТ")
        choice = input("Вибір (1/2): ")
        if choice == "1":
            daemon.start_server()
        elif choice == "2":
            ip = input("Введіть IP сервера: ")
            daemon.start_client(ip)
```

ДОДАТОК Б

Повний код веб-серверу на базі FastAPI для керування живленням ПК

mick@mikvm: ~/pc_remote

GNU nano 8.4 main.py

```
import uvicorn
from fastapi import FastAPI
from fastapi.responses import HTMLResponse
from gpiozero import OutputDevice
import asyncio

# --- НАЛАШТУВАННЯ СЕРВЕРУ ---
# Призначено порти GPIO, до яких підключені оптопари через клеми "+" модулів
PIN_POWER = 17
PIN_RESET = 27

# Задано тривалість імпульсів
SHORT_PRESS = 0.5  # Звичайне натискання
LONG_PRESS = 5.0   # Довге натискання для примусового вимкнення

app = FastAPI(title="PC Controller")

# Ініціалізація GPIO
# Параметр active_high=True означає, що команда .on() подасть 3.3В і відкриє оптопару, off() >
try:
    power_sw = OutputDevice(PIN_POWER, active_high=True, initial_value=False)
    reset_sw = OutputDevice(PIN_RESET, active_high=True, initial_value=False)
    GPIO_AVAILABLE = True
except Exception:
# Обробка помилки, якщо код запущено не на Raspberry Pi
    GPIO_AVAILABLE = False
    printf"Помилка ініціалізації GPIO: {e}")

# --- HTML ІНТЕРФЕЙС ---
html_ui = """
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
    <title>PC Remote</title>
    <style>
        body { background: #121212; color: #fff; font-family: sans-serif; text-align: center;>
        .btn {
            display: block; width: 100%; max-width: 400px; margin: 15px auto; padding: 20px;
            font-size: 18px; border: none; border-radius: 12px; cursor: pointer; color: white;
            transition: opacity 0.2s;
        }
        .btn:active { transform: scale(0.98); }
        .btn-pwr { background: #2ecc71; }
        .btn-rst { background: #f1c40f; color: #000; }
        .btn-kill { background: #e74c3c; margin-top: 40px; }
        .status { margin-top: 20px; color: #888; font-size: 14px; min-height: 20px;}
    </style>
</head>
<body>
    <h2Керування ПК</h2>

    <button class="btn btn-pwr" onclick="doAction('/power')">
        POWER (Увімк / Вимк)
    </button>
```

^G Довідка ^O Виписати ^F Пошук ^K Вирізати ^T Виконати ^C Місце

mick@mikvm: ~/pc_remote

GNU nano 8.4 main.py

```
<body>
    <h2Керування ПК</h2>

    <button class="btn btn-pwr" onclick="doAction('/power')">
        POWER (Увімк / Вимк)
    </button>

    <button class="btn btn-rst" onclick="doAction('/reset')">
        RESET (Перезавантаження)
    </button>

    <button class="btn btn-kill" onclick="if(confirm('Виконати примусове вимкнення?')) doAction('/force')">
        ORCE OFF (Затиснути 5с)
    </button>

    <div id="log" class="status">Система готова</div>

    <script>
        function doAction(url) {
            const log = document.getElementById('log');
            log.innerText = "Надсилання сигналу...";
            log.style.color = "#ccc";

            fetch(url, {method: 'POST'})
                .then(r => r.json())
                .then(data => {
                    log.innerText = "✅ " + data.msg;
                    log.style.color = "#2ecc71";
                    setTimeout(() => log.innerText = "Готовий", 3000);
                })
                .catch(e => {
                    log.innerText = "❌ Помилка з'єднання";
                    log.style.color = "#e74c3c";
                });
        }
    </script>
</body>
</html>
"""

@app.get("/", response_class=HTMLResponse)
async def index():
    return html_ui

@app.post("/power")
async def action_power():
    if GPIO_AVAILABLE:
        power_sw.on()
        await asyncio.sleep(SHORT_PRESS)
        power_sw.off()
    return {"msg": "Кнопку POWER натиснуто"}

@app.post("/reset")
async def action_reset():
    if GPIO_AVAILABLE:
        reset_sw.on()
        await asyncio.sleep(SHORT_PRESS)
        reset_sw.off()
    return {"msg": "Кнопку RESET натиснуто"}
```



```
@app.get("/", response_class=HTMLResponse)
async def index():
    return html_ui

@app.post("/power")
async def action_power():
    if GPIO_AVAILABLE:
        power_sw.on()
        await asyncio.sleep(SHORT_PRESS)
        power_sw.off()
    return {"msg": "Кнопку POWER натиснуто"}

@app.post("/reset")
async def action_reset():
    if GPIO_AVAILABLE:
        reset_sw.on()
        await asyncio.sleep(SHORT_PRESS)
        reset_sw.off()
    return {"msg": "Кнопку RESET натиснуто"}

@app.post("/force")
async def action_force():
    if GPIO_AVAILABLE:
        power_sw.on()
        await asyncio.sleep(LONG_PRESS)
        power_sw.off()
    return {"msg": "Примусове вимкнення виконано"}

if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)
▬

^G Довідка      ^O Виписати     ^F Пошук        ^K Вирізат
^X Вихід        ^R Чит. файл    ^\ Заміна       ^U Вставит
```

ДОДАТОК В

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

на тему: «Автономна платформа віддаленого керування робочим місцем на базі Raspberry Pi»

Виконав студент групи КСДМ-62
Токар Михайло Віталійович

Керівник кваліфікаційної роботи :
Волохін Віталій Васильович ,
доцент кафедри, к.т.н., доцент

Київ – 2025

Мета роботи – розробка та реалізація системи віддаленого управління та моніторингу комп'ютерів на базі одноплатного комп'ютера Raspberry Pi 4B.

Об’єкт дослідження – процес автоматизації віддаленого доступу до робочої станції з використанням одноплатного комп'ютера Raspberry Pi 4B.

Предмет дослідження – методи та підходи віддаленого керування для забезпечення одночасно низької затримки обробки сигналів, економічної ефективності та надійності функціонування 24/7.

Актуальність обраної теми

За умов підвищення попиту гібридного роботи та розповсюдження розподілених серверних центрів мережеві спеціалісти потребують надійних рішень для віддаленого керування обладнанням без фізичного доступу. Промислові KVM вимагають дорогої спеціалізованої інфраструктури та складної кабельної розводки, тоді як KVM over IP дозволяє використовувати існуючу мережеву інфраструктуру. Під час аналізу ринку бюджетних IP-KVM було виявлено, що системи на базі Raspberry Pi пропонують економічну альтернативу комерційним рішенням за ціною $100-150 замість $500-5000.

Raspberry Pi виступає ідеальною платформою для edge-node завдяки компактності та низькому енергоспоживанню. Ускладнень в інтеграції з сучасними платформами автоматизації, наприклад, Home Assistant, додає закритість комерційних KVM систем. Open-source рішення на базі Linux та Python дозволяють повну кастомізацію та інтеграцію з будь-якою екосистемою.

Усі ці фактори наводять на очевидне рішення: створення власної системи віддаленого доступу до робочої станції.

3

Використання Raspberry Pi 4B

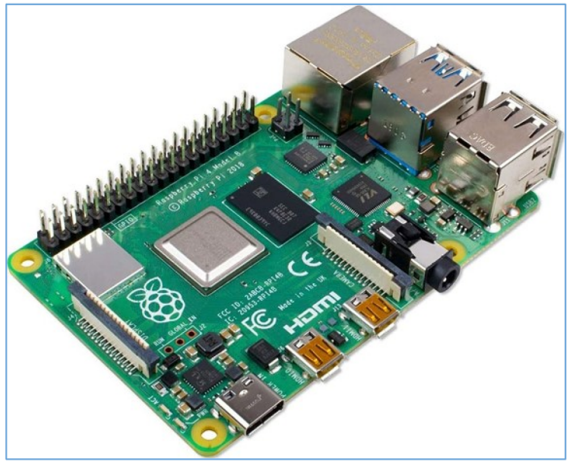


Рис. 4.1 - Друкована плата Raspberry Pi 4B

Raspberry Pi 4B є оптимальною платформою для IP-KVM та розумного дому завдяки поєднанню низької вартості в $60, достатньої продуктивності (4-ядерний процесор, 4 ГБ RAM), підтримки множинних з'єднань (Gigabit Ethernet, Wi-Fi 5, USB 3.0), а також великої екосистеми GPIO та датчиків. Розвинена спільнота розробників й ентузіастів вже створила багато цікавих рішень та вирішила величезну кількість проблем.

Компонентна база проєкту

Компонентна база проєкту включає USB HDMI-грабер на чіпі Actions AM8352 для захоплення HDMI-сигналу з роздільністю до 1080p@30fps у форматі MJPEG, дві оптопари PC817 для гальванічно розв'язаного керування кнопками Power та Reset, цифровий температурний датчик DS18B20 з інтерфейсом 1-Wire.



Рис. 5.1 - Датчик DS18B20, оптопара PC817 та USB HDMI-грабер

Програмна частина системи

Написання усіх програмних компонентів було виконано мовою Python для дистрибутиву Debian Linux, встановленому на Raspberry Pi 4B – Raspberry Pi OS Desktop. Python глибоко інтегровано в ОС, а тому працювати з ним дуже зручно.

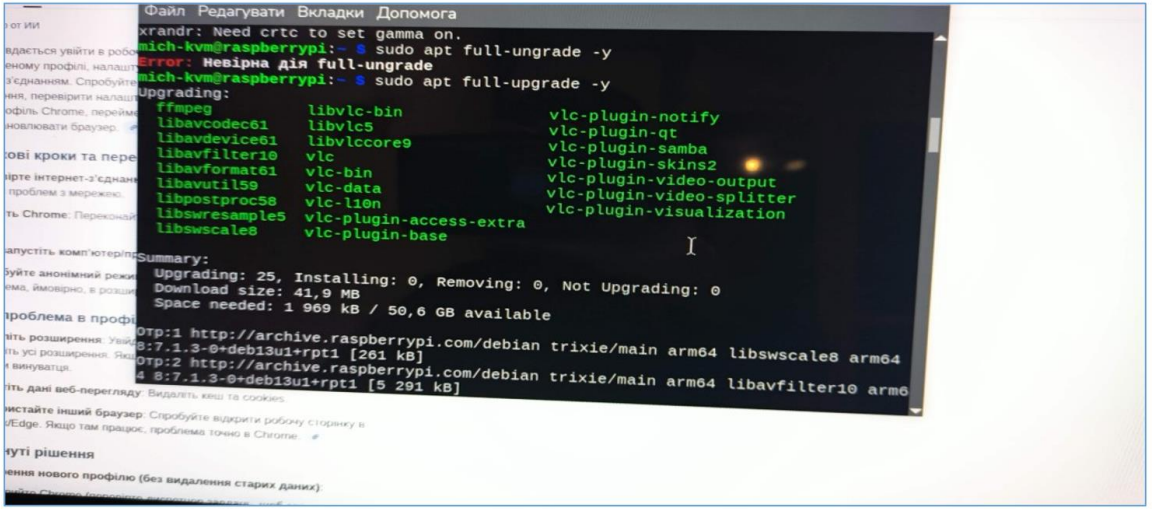


Рис. 6.1 – Програмне середовище системи

Фізичне виконання проєкту

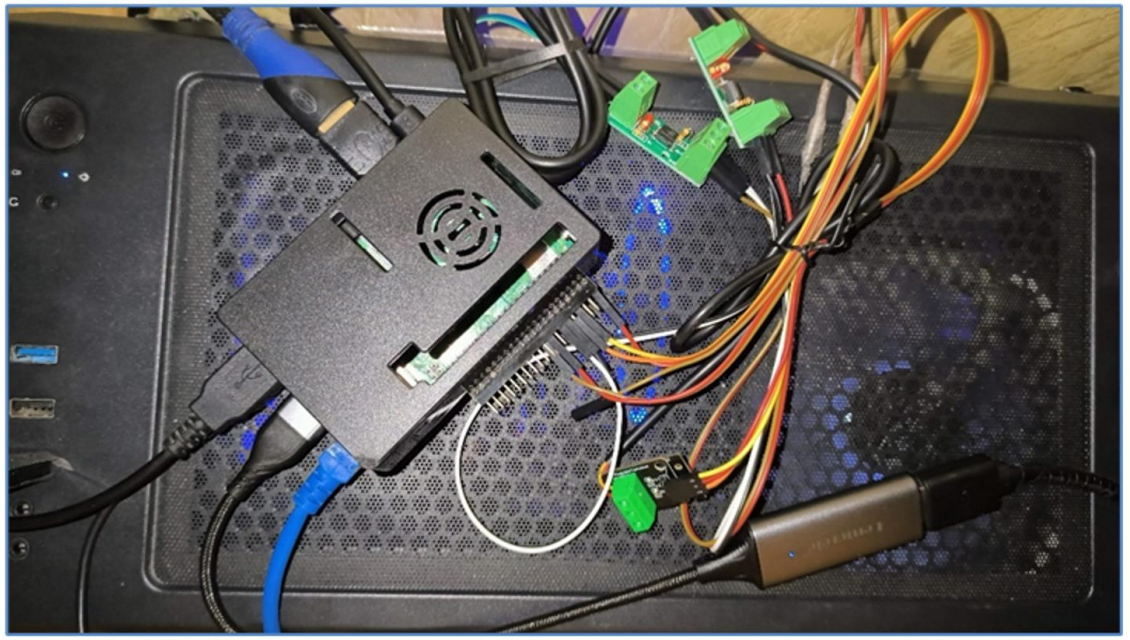


Рис. 7.1 – Реальне виконання проєкту включає сам мікрокомп’ютер, пластиковий захисний корпус, сенсорний 3.5’’ екран для локального керування Raspberry Pi 4B, підключений через спліттер пінів GPIO, що не блокує GPIO для підключення датчиків.

Сценарії використання проєкту

Основними і найбільш вірогідними сценаріями використання системи будуть:

- Сценарій 1. Нормальна робота системи при стабільній роботі всіх компонентів.
- Сценарій 2. Перевстановлення ОС Windows з Raspberry Pi як USB-накопичувача.
- Сценарій 3. Віддалене завантаження та налаштування BIOS/UEFI.
- Сценарій 4. Діагностика та відновлення після критичного збою.

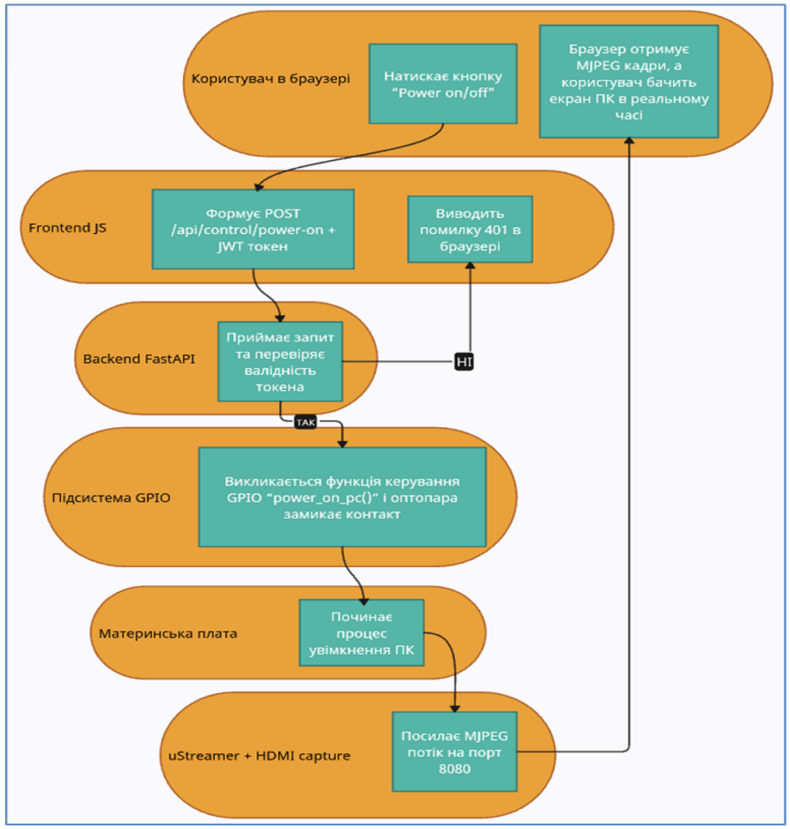


Рис. 8.1 – Процес увімкнення ПК

Підсистема трансляції відеосигналу

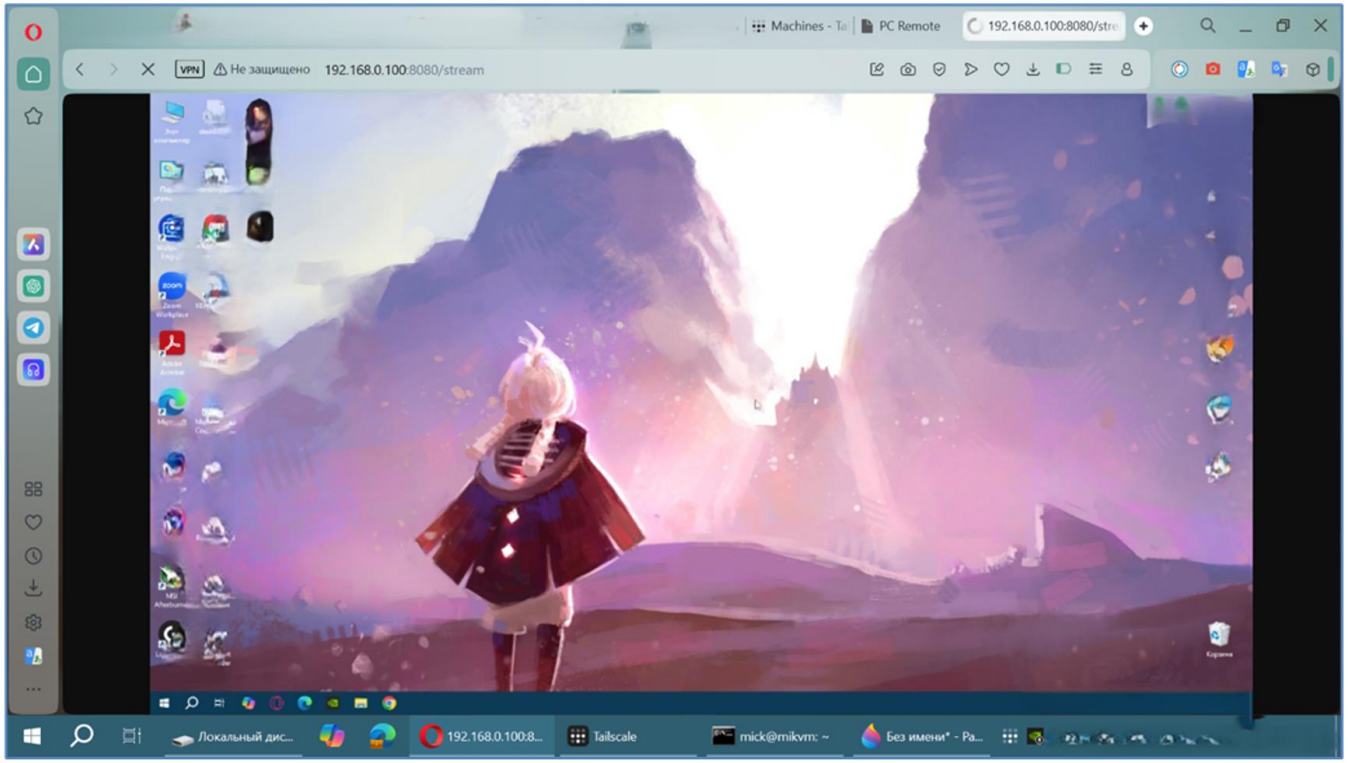


Рис. 9.1 – Вигляд трансляції в браузері на іншому ПК

Підсистема керування HID

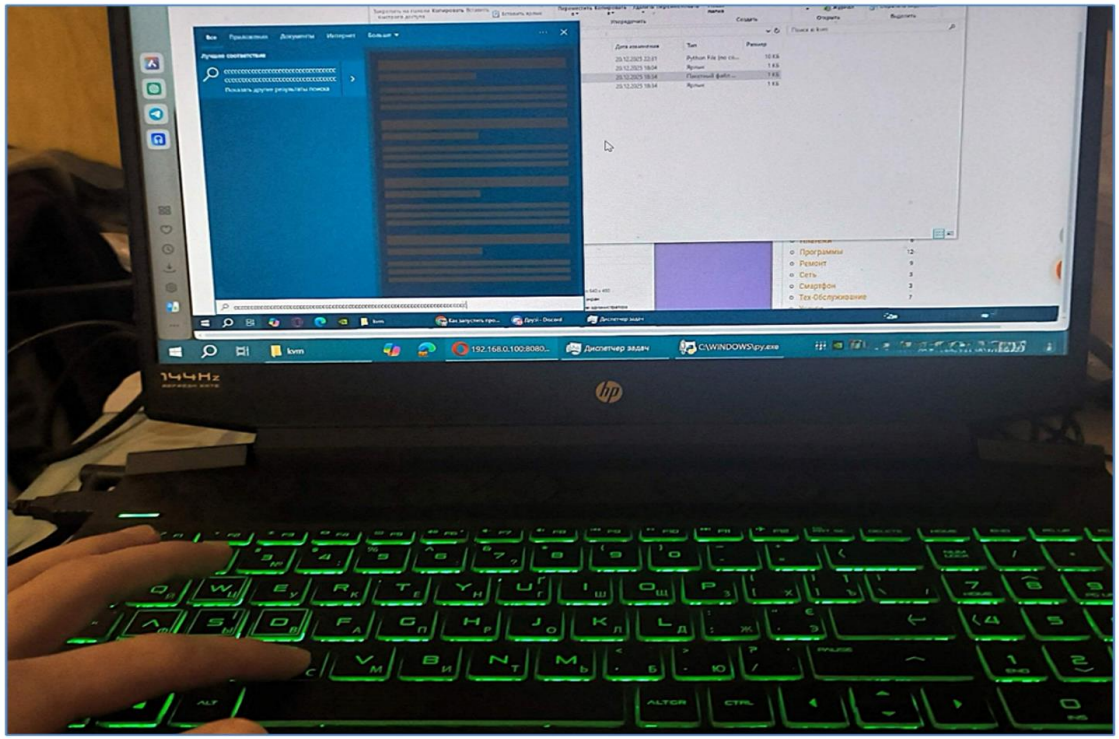


Рис. 10.1 – Натискання клавіші на клієнті та виконання дії на сервері

Підсистема керування живленням ПК

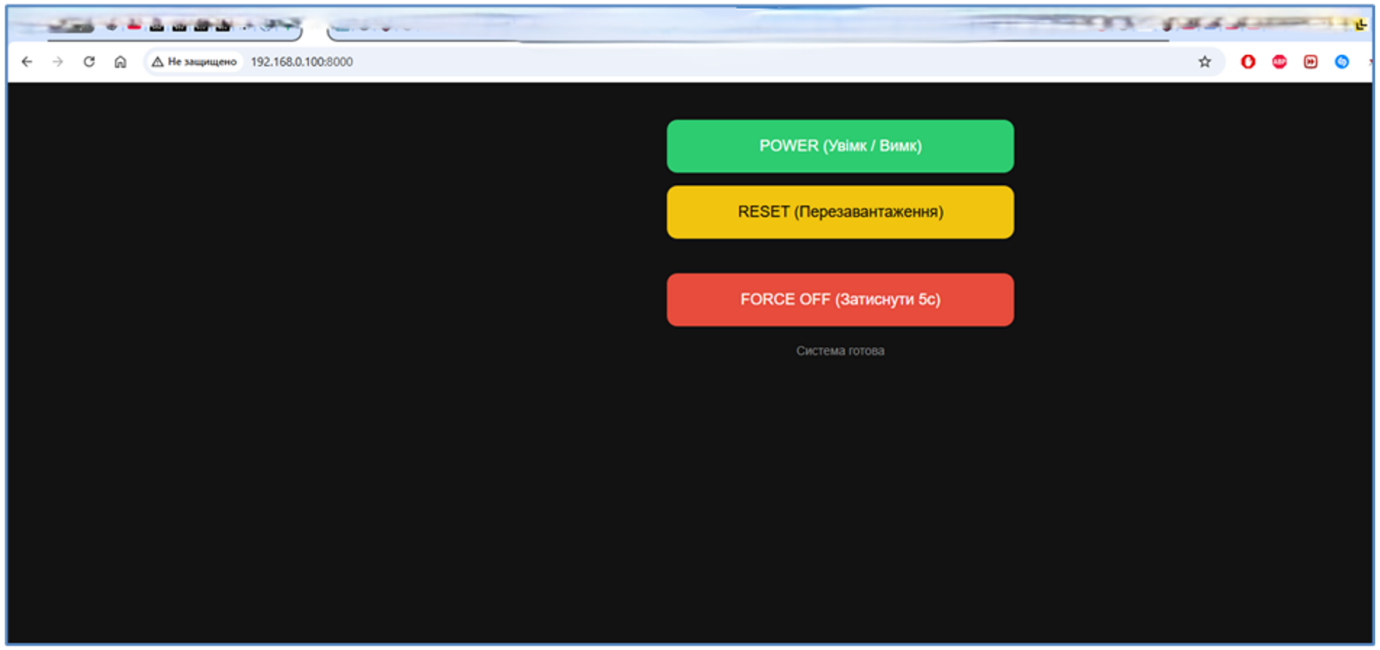


Рис. 11.1 – Інтерфейс керування в десктопному браузері

Відеотрансляція через VPN

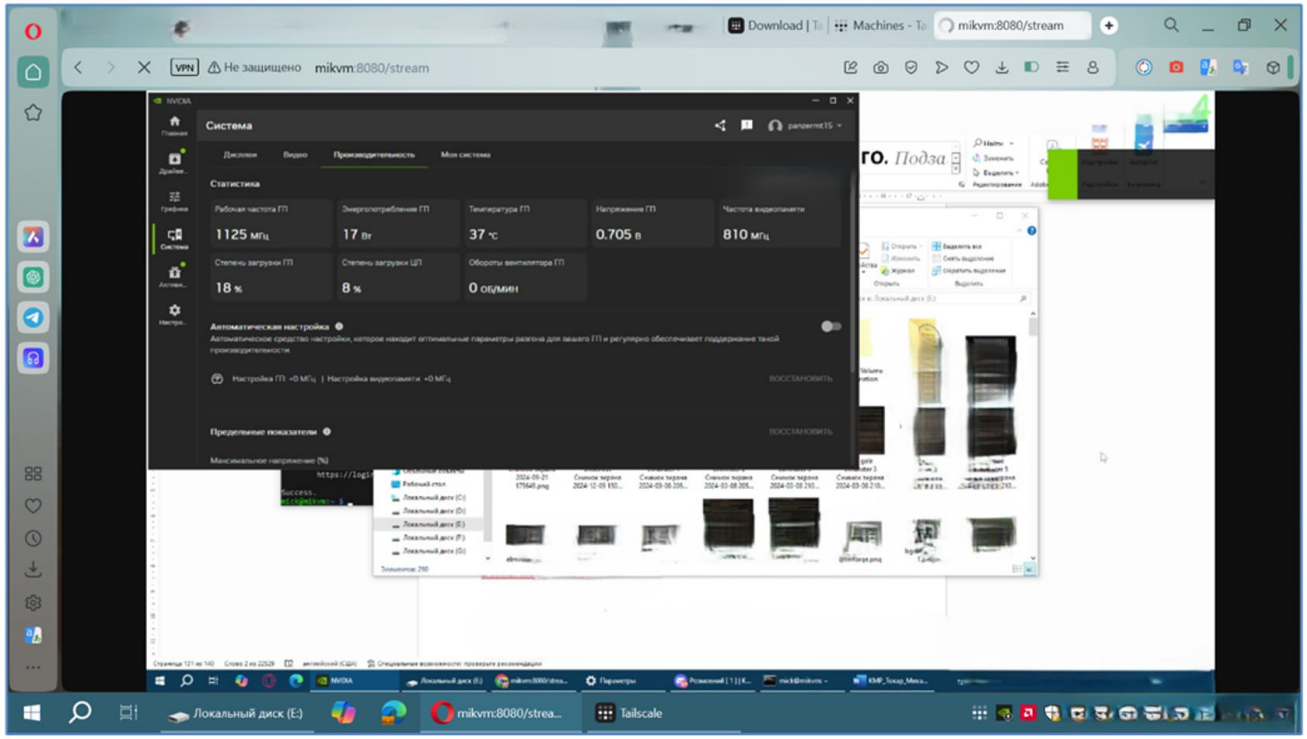


Рис. 12.1 – Доступ до відеотрансляції через Tailscale

Майбутній розвиток проєкту

Найбільш перспективними напрямками модернізації проєкту я вважаю поглиблену інтеграцію наявної IP-KVM системи в екосистему «Розумного будинку»; підвищення якості передаваного зображення до 1080p@60fps шляхом покупки нового грабера на чіпі MS2130; збільшення потужності обчислювальної системи шляхом покупки RPi5; створення адаптивної системи освітлення, що може адаптуватися до умов природнього освітлення або забезпечувати постійну освітленість на заданому рівні; створення просунутої системи безпеки будинку з використанням камер з комп’ютерним зором та сигналізації.

13

ВИСНОВКИ

Під час виконання даної магістерської роботи була успішно розроблена та реалізована система віддаленого управління та моніторингу комп'ютерів на базі Raspberry Pi 4B, що більш ніж доводить життєздатність використання одноплатних комп'ютерів як платформи для IP-KVM та edge computing застосувань. Система наочно демонструє, що економічні альтернативи комерційним KVM решенням можуть забезпечити порівнянну функціональність при значних енергетичних та вартісних заощадженнях до 10 разів.

Реалізація системи на практиці підтвердила переваги фізичної та програмної модульної архітектури. Використання чіпу Actions AM8352 дозволило організувати стабільну трансляцію відеопотоку 720p30. Налаштування режиму OTG на RPi 4B забезпечило стабільну трансляцію сигналів HID крізь мережу. При цьому сумарна затримка склала 100-200 мс.

ПРОДОВЖЕННЯ ВИСНОВКІВ

Написаний за допомогою фреймворку FastAPI backend компонент системи успішно реалізує функції KVM включаючи увімкнення й перезавантаження робочої станції методом прямого управляння через інтерфейс GPIO, реалізацію з'єднання для передачі подій у реальному часі та інтеграцію компонентів розумного дому.

Однією з основних цілей кваліфікаційної роботи було створення KVM системи, яка орієнтована на мінімальну завантаженість заліза RPi 4B. Після усіх налаштувань вона склала близько 30% завантаженості процесора, а це означає, що перспективи подальшої модернізації доволі великі.

15

Апробація

1. Токар М. В., Волохін В. В., СУЧАСНІ МЕТОДИ АВТЕНТИФІКАЦІЇ ТА РОЗМЕЖУВАННЯ ДОСТУПУ В ФІЗИЧНИХ СИСТЕМАХ ВІДДАЛЕНОГО АДМІНІСТРУВАННЯ. *Науковий простір: аналіз, сучасний стан, тренди та перспективи* : IX Міжнар. студент. наук. конф., м. Вінниця, 26 груд. 2025 р. URL: https://doi.org/10.62732/liga-inter-26.12.2025 (дата звернення: 24.12.2025).
2. Токар М. В., Волохін В. В. АПАРАТНА РЕАЛІЗАЦІЯ ДИСТАНЦІЙНОГО КЕРУВАННЯ ЖИВЛЕННЯМ ПК З ВИКОРИСТАННЯМ ГАЛЬВАНІЧНОЇ РОЗВ'ЯЗКИ. *Традиційні та інноваційні підходи до наукових досліджень* : X Міжнар. наук. конф., м. Луцьк, 26 груд. 2025 р. С. 313-316. URL: https://doi.org/10.62731/mcnd-26.12.2025 (дата звернення: 24.12.2025).

16