Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету(відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу»

Виконав: студент 2-го курсу групи 2ПІ-20м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Лесик О.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри ПЗ
Катєльніков Д.І.
(прізвище та ініціали)

« » 2021 р.

Опонент: к.т.н., доцент кафедри КН
Арсенюк І.Р.
(прізвище та ініціали)
« » 2021 р.

Допущено до захисту
Завідувач кафедри ПІ

д.т.н., проф Романюк О. Н.
(прізвище та ініціали)

«____»_____2021 р.

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Ступінь вищої освіти – магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"01" жовтня 2021 року

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

Лесику Олександру Валентиновичу

1. Тема роботи Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу. Керівник роботи: к.т.н., доцент кафедри ПЗ Катєльніков Д.І. затверджені наказом вищого навчального закладу від "25" вересня 2021 року № 214

2. Строк подання студентом роботи 01.12.2021 р.

3. Вихідні дані до роботи : мова програмування – Java; фреймворк (бібліотека) – Android Native; тип обробки даних – текстовий, графічний; формат зберігання конфігурації – xml; Тип додатку – Офлайн-конфігуратор; розподілена система управління версіями – Git; середовище розробки – Android Studio.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; розробка структур та алгоритмів додатку; розробка методу візуалізації; розробка додатку; тестування додатку; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): тема роботи, мета і задачі роботи; порівняльний аналіз додатків-конфігураторів; функціонал конфігураторів-аналогів; діаграма потоків даних додатку для конфігурування; інтерфейс програми; інтерфейс інструментів розробки.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Катєльніков Д.І. к.т.н., доц. каф. ПЗ	28.09.2021 р.	01.12.2021 р.
5	Бурєннікова Н.В. д.е.н., проф. каф. ЕПВМ	28.10.2021 р.	28.11.2021 р.

7. Дата видачі завдання 28.09.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	28.10.2021 р. - 01.10.2021 р.	Вик.
2	Розробка загальної моделі та структури системи	02.10.2021 р. – 10.10.2021 р.	Вик.
3	Підбір та інтеграція програмних засобів для реалізації розроблених модулі системи	11.10.2021 р. - 20.10.2021 р.	Вик.
4	Розробка модулю взаємодії між Android-HID	21.10.2021 р. - 05.11.2021 р.	Вик.
5	Розробка модулів побудови та збереження конфігурації системи	06.11.2021 р. - 15.11.2021 р.	Вик.
6	Тестування роботи модулів та інтерфейсу додатку для конфігурування	16.11.2021 р. - 20.11.2021 р.	Вик.
7	Економічна частина	21.11.2021 р. - 28.11.2021 р.	Вик.
8	Оформлення матеріалів до захисту МКР	29.11.2021 р. - 01.12.2021 р.	Вик.

Студент _____ **Лесик О.В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Катєльніков Д.І.**
(підпис) (прізвище та ініціали)

Опонент магістерської кваліфікаційної роботи _____ **Арсенюк І.Р**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.415

Лесик О.В. Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 126 с.

На укр. мові. Бібліогр.: 46 назв; рис.: 29; табл. 7.

У магістерській кваліфікаційній роботі розроблено мобільний додаток для конфігурування приймально-контрольного пристрою. Він дозволяє швидко та якісно налаштувати приймально-контрольний пристрій для забезпечення його правильної роботи з охорони об’єкту. У загальній частині роботи розглянуто особливості роботи з HID-пристроями та проведено аналіз наявних програмних аналогів. У розрахунково-конструкторській частині виконана розробка структурна схема додатку та проектування окремих його модулів . У технологічній частині виконано програмну реалізацію мобільного додатку для конфігурування приймально-контрольного пожежного пристрою, було описано класи, структури, графічний інтерфейсу програмного додатку.

Графічна частина складається з 15 плакатів із представленням розробки.

При створенні програмного додатку було використано мову програмування Java, а саме версії 8 та Kotlin. Додаток працює під операційною системою Android, версії 5.0 Lolllipop та вище.

Розробка додатку була виконана в інтегрованому середовищі програмування Android Studio.

Ключові слова: Android, HID-пристрій, протокол UDP, Android-розробка, інтерфейс спілкування HID Custom.

ABSTRACT

Lesyk O.V. Development of a method and software for a mobile application for configuring a fire control receiver and control device. Master's thesis in specialty 121 - Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2021. 126 p.

In Ukrainian language. Bibliogr .: 46 titles; fig .: 27; table 7.

In the master's qualification work, a mobile application was developed to configure the receiver-control device. It allows you to quickly and efficiently configure the receiver-control device to ensure its proper operation to protect the object. In the general part of the work the peculiarities of work with HID-devices are considered and the analysis of available software analogues is carried out. In the calculation and design part, the structural scheme of the application and the design of its individual modules were developed. In the technological part the software implementation of the mobile application for the configuration of the receiving and control fire device was performed, the classes, structures, graphical interface of the software application were described.

The graphic part consists of 15 posters presenting the development.

The Java programming language, version 8 and Kotlin, was used to create the software application. The application runs on Android, version 5.0 Lolllipop and higher. The application was developed in the integrated programming environment Android Studio.

Keywords: Android, HID-device, UDP protocol, Android-development, HID Custom communication interface.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. USB HID (human interface device) class - клас пристроїв USB для взаємодії з людиною. Цей клас включає в себе такі пристрої як клавіатура, миша, ігровий контролер і т.д.

Крім детальних специфікацій класичних пристроїв введення (типу клавіатур і мишок) стандарт HID визначає особливий клас пристроїв без детальних специфікацій. Цей клас іменується USB HID Consumer Control і представляє по суті нерегламентований канал зв'язку з пристроєм. При цьому пристрій користується тими ж стандартними для операційної системи драйверами що і мишка з клавіатурою. Таким чином можна створити USB пристрій який не вимагає створення та інсталяції спеціальних драйверів в більшості поширених комп'ютерних операційних систем.

Інтерфейс пристроїв також забезпечений особливим дескриптором, який визначає, чи є пристрій завантажувальним. Завантажувальний пристрій, строго відповідне мінімальним вимогам протоколу, буде розпізнано і завантажено BIOS. Кожен USB HID інтерфейс зв'язується з хостом з використанням функції управління або функції переривання[1].

Цей інструмент дозволяє створювати, редагувати і перевіряти приховані дескриптори звітів. Інструмент також підтримує різні формати виводу.txt, .inc, .h і т.д.). DT використовує таблиці використання на основі ASCII і також підтримує сторінки, визначені постачальником. Включені файли таблиці використання для документа таблиці, використання HID 1.0 Release Candidate 1, Клас монітора 1.0 Release Candidate 2 і Специфікація класу потужності.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та завдання дослідження. Метою роботи є розширення можливостей функціоналу приймально-контрольних пожежних приладів за рахунок їх конфігурування з зовнішнього комп’ютера або мобільного пристрою.

Основними задачами дослідження є:

- провести аналіз методів керування периферійними пристроями за протоколом USB Custom HID з Android пристроїв;
- розробити бітовий протокол обміну даними з приймально-контрольним пожежним приладом.
- розробити програмні засоби та систему конвертації інформації на основі запропонованих методів;
- провести тестування розроблених програмних засобів керування периферійними пристроями.

Об’єктом дослідження є процес керування периферійними пристроями за протоколом USB Custom HID.

Предмет дослідження – методи та алгоритми взаємодії Android пристроїв з протоколом USB Custom HID.

Наукова новизна отриманих результатів:

1. Подальшого розвитку отримав метод сполучення приймально-контрольних пожежних приладів та комп’ютера або мобільного пристрою, який на відміну від відомих, замість звичайного інтерфейсу USB-HID використовує комбінацію інтерфейсу USB-OTG та протоколу передачі даних UDP, що дозволяє покращити надійність з’єднання та потоку передачі даних між Android-пристроями та приймально-контрольними пожежними приладами.
2. Подальшого розвитку отримав метод серіалізації об’єктів у вихідний потік, який, на відміну від існуючих методів XML-серіалізації, використовує додаткові класи-нащадки, які мають власну реалізацію перевантажених методів класів XMLEncoder і XMLDecoder, що дозволяє підвищити гнучкість низькорівневого вихідного потоку даних.

Практичне значення отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для

керування периферійними пристроями за протоколом USB Custom HID.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованій праці [1], опублікованих у співавторстві, автору належать такі результати:

- аналіз методів і засобів реалізації програмного продукту;
- реалізація програмного продукту.

Апробація матеріалів магістерській кваліфікаційній роботі. Основні положення магістерської кваліфікаційної роботи обговорювалися на Міжнародній науково-практичній інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» 9-10 листопада 2021.

Структура та обсяг роботи. Робота складається з п’яти розділів. У першому розділі виконано аналіз стану питання HID пристроїв, зроблено порівняння додатків конфігураторів, виконано аналіз методів і засобів реалізації програмного продукту, а також сформульовано задачі магістерської кваліфікаційної роботи.

У другом розділі проаналізовано інформаційне забезпечення, розроблено загальну модель системи, алгоритм взаємодії Android з HID-пристроями, загальний алгоритм роботи додатку, модуль формування даних.

У третьому розділі виконано варіантний аналіз, розроблено модуль ідентифікування та з’єднання з HID-пристроями, модуль обміну даними, модуль побудови конфігурації та модуль збереження/завантаження налаштувань і обґрунтування вибору засобів реалізації програмного продукту модуля.

У четвертому розділі виконано загальне тестування програмного продукту.

У п’ятому розділі розраховано кошторис витрат, економічний ефект та термін окупності програмного додатку.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану USB HID пристроїв

До появи концепції HID існувало кілька конкретних протоколів, використовуваних для кожного типу пристроїв введення. Це означає, що якщо ви хочете підключити мишу до комп'ютера, вам знадобиться окремий протокол. Аналогічно, для однієї і тієї ж клавіатури потрібен окремий протокол і т.д. Пристрої, які хочуть підключитися до комп'ютера, повинні використовувати існуючі протоколи, або створювати власні драйвери. Це робить налаштування і налаштування пристрою морочливою і трудомісткою.

Інформація про USB-пристрій зберігається у сегментах його ПЗУ (лише для читання)пам'ять). Ці сегменти називаються дескрипторами. Дескриптор інтерфейсу може визначити пристрій, що належить до одного з кінцевої кількості класів. Клас HID є основним напрямком цього документа.

Пристрій класу USB/HID використовує відповідний драйвер класу HID для отримання та маршрутизувати всі дані. Маршрутизація та пошук даних здійснюється шляхом вивчення дескрипторів пристрою та наданих ним даних.

Протокол HID значно полегшує виробникам аксесуарів для ПК створення пристроїв з широкої сумісністю. Всі сучасні операційні системи підтримують протокол HID. Ви можете підключити USB-клавіатуру до ПК з Windows, Mac, Chromebook або навіть планшета Android, і вона повинна працювати в найкоротші терміни. Все завдяки HID[2].

Іншими словами, більшість операційних систем (ОС) розпізнають базові приховані пристрої, такі як миша і клавіатура, без необхідності в певному драйвері. Це також є необхідною умовою для plug and play (PnP) - корисної властивості USB-пристроїв.

Під час ранньої розробки специфікації HID передбачалися підкласи використовувати для ідентифікації конкретних протоколів різних типів класу HID пристроїв. Хоча це відображає модель, яка зараз використовується промисловістю

(усі пристрої використання протоколів, визначених подібними популярними пристроями), швидко стало очевидним, що цей підхід був надто обмежувальним. Тобто пристрої повинні вузько вписуватися визначених підкласів і не зможе забезпечити жодних функцій крім цього підтримується підкласом.

Використання є частиною дескриптора звіту і надає розробнику програми інформація про те, що насправді вимірює елемент керування. Крім того, тег Usage вказує на запропоноване постачальником використання для певного елемента керування або групи елементів керування.

Хоча дескриптори звіту описують формат даних- наприклад, три 8-бітові поля - тег Usage визначає, що слід робити з даними - наприклад, введення x, y та z. Ця функція дозволяє постачальнику гарантувати, що користувач бачить послідовне призначення функцій елементам керування у всіх додатках.

Клас HID складається переважно з пристроїв, які використовуються людьми для керування функціонування комп'ютерних систем. Типові приклади пристроїв класу HID включати:

- Клавіатури та вказівні пристрої, наприклад, стандартні миші, трекболи та джойстики.
- Елементи керування на передній панелі, наприклад: ручки, перемикачі, кнопки та повзунки.
- Елементи керування, які можна знайти на таких пристроях, як телефони, пульт від VCR елементи керування, ігри чи симулятори, наприклад: рукавички для даних, дросельні заслінки, рульові колеса та педалі керма.10 Визначення класу пристроїв для пристроїв інтерфейсу людини (HID), версія 1.11 27.06.00:

Пристрої, які можуть не вимагати взаємодії з людиною, але надають дані подібним чином форматувати на пристрої класу HID, наприклад, зчитувачі штрих-кодів, термометри або вольтметри.

Багато типових пристроїв класу HID включають індикатори, спеціалізовані дисплеї, аудіо зворотний зв'язок, а також силовий або тактильний зворотний зв'язок. Отже, визначення класу HID включає підтримку різних типів виводу,

спрямованого на кінцевого користувача.

Клас HID не обов'язково є людським інтерфейсом. Але пристрій, що використовує клас HID, повинен працювати в межах класу HID. Ця реалізація класу HID компонента USB має такі особливості:

- Усі дані обмінюються у звітах. Це структури фіксованої довжини, які надсилаються або запитуються хостом USB під час передачі керування або переривання. Звіти мають гнучкий формат і можуть містити будь-який тип даних. Кожен пристрій HID повинен мати один вхідний звіт у своєму дескрипторе звіту. Звіти про вихідні дані та функції є необов’язковими.
- Кінцева точка IN переривання потрібна для надсилання вхідних звітів на USB-хост.
- Максимальна кількість кінцевих точок IN та OUT переривань обмежена 1.
- Кінцева точка переривання OUT необов'язкова.
- Оскільки пристрій HID може надсилати дані в будь-який момент часу, використовуючи кінцеву точку переривання IN, драйвер хосту USB повинен переконатися, що дані періодично опитуються.
- Компонент USB підтримує клас HID для USB-пристроїв і USB-хостів (тільки для програм MDK-Professional).

HID Mobile Access® дозволяє співробітникам використовувати свої смартфони, планшети та електроніку, що носиться для отримання доступу до дверей, воріт, мереж, служб і т. д. Це нове рішення для контролю доступу спрощує роботу в сучасному світі мобільних технологій і надає вашій організації сучасний і професійний вигляд[3].

Рішення HID Mobile Access, що використовує можливості Seos® та відповідних технологій, пропонує наступні переваги:

- Зручність використання: для доступу до об'єктів користувачів більше не потрібно носити з собою картки, натомість вони можуть

використовувати свої мобільні пристрої. Надзвичайно зручні функції жестового управління дотиком або поворотом смартфона забезпечують простоту та ефективність контролю доступу для співробітників підприємств.

- Операційна ефективність та економічність: використання стабільного онлайн-порталу управління HID Global дозволяє адміністраторам створювати засоби ідентифікації, керувати ними, видавати та відкликати їх за допомогою хмарних служб. Оплата підписки дає можливість користувачам планувати витрати, замовляти нові та відкликати існуючі ліцензії. Завдяки цьому організації можуть масштабувати передплату залежно від зміни умов роботи.
- Високий рівень безпеки: HID Mobile Access працює через високозахищену надійну хмарну платформу, договір обслуговування до якої гарантує доступність на рівні 99,5 % відповідно до зобов'язань HID Global щодо якості та часу роботи сервісів. Крім того, рішення створено на основі технології ідентифікації особистості Seos та використовує найефективніші методи забезпечення цілісності даних.

Примітно, що USB HID може бути використаний як для опису роботи самого пристрою, так і для опису інтерфейсу пристрою. Наприклад, цілком допустимим буде використання USB пристрою, що має два різні USB інтерфейси одночасно (наприклад, USB-телефон може використовувати клавіатуру HID і USB аудіо пристрій для мікрофона).

Інтерфейс пристроїв також має спеціальний дескриптор, який визначає, чи є пристрій завантажувальним. Завантажувальний пристрій, який суворо відповідає мінімальним вимогам протоколу, буде розпізнаний і завантажений BIOS. Кожен USB HID інтерфейс зв'язується з хостом за допомогою функції управління або функції переривання.

1.2 Порівняльний аналіз аналогів

Наразі існує багато різних десктопних додатків для роботи з HID пристроями, а саме прилади пожежної безпеки. Але не вистачає мобільних конфігураторів. Розглянемо декілька з них.

tLoader – портативний десктопний додаток для конфігурування пожежних приймально-контрольних приладів окремої серії не адресних пристроїв Tiras П. Перед створенням конфігурації потрібно вибрати зі списку прилад для якого вона буде створюватись і потім проводити налаштування. Є можливість вичитування налаштованої раніше конфігурації, відсутня авторизація рівня доступу. Створена конфігурація записується у файл розширення XML та відсилається на прилад[5].

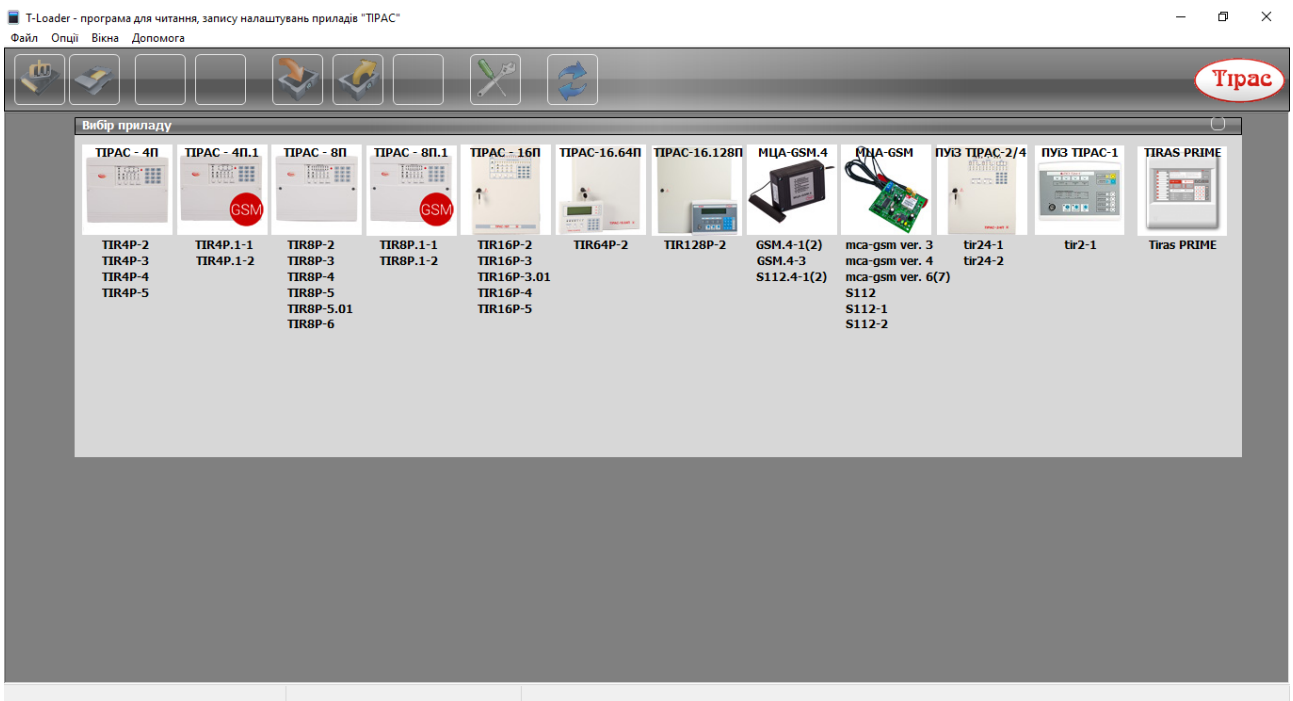


Рисунок 1.1 – Конфігуратор tLoader

tLoader Prime – портативний десктопний додаток для конфігурування пожежних приймально-контрольних приладів окремої серії неадресних Tiras Prime, аналог попередньо розглянутого додатку. Перед створенням конфігурації потрібно вибрати зі списку прилад для якого вона буде створюватись і потім проводити налаштування. Є можливість вичитування налаштованої раніше конфігурації,

відсутня авторизація рівня доступу. Вбудоване оновлення ПЗ за бажанням користувача. Створена конфігурація записується у файл розширення XML та відсилається на прилад[6].

Запис конфігурації в на приймально-контрольний пожежний пристрій реалізований простим поділенням файлу на пакети та послідовною передачею пакетів даних. Таке налаштування можливе лише за умови повного знеструмлення пристрою та під’єднання його до ПК за допомогою USB кабелю. Вимкнення пристрою для внесення змін в конфігурацію системи не є безпечним для підконтрольного об’єкту.

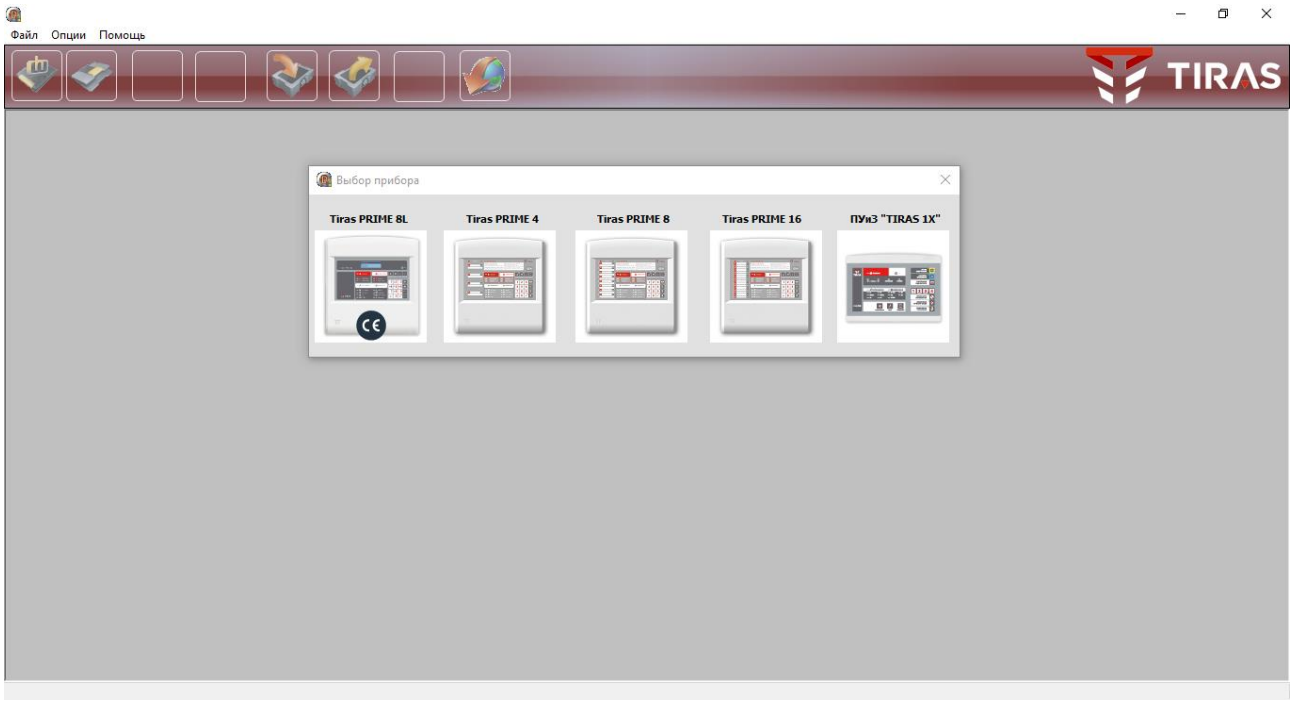


Рисунок 1.2 – Конфігуратор tLoader Prime

oLoaderII – десктопний додаток для конфігурування охоронних приймально-контрольних приладів. В додатку реалізоване автоматичне оновлення ПЗ та оновлення вбудованого ПЗ приладу за бажанням користувача. Присутня авторизація рівнів доступу та хмарна авторизація користувача. Можливе вичитування та запис конфігурації, працює з файлами розширення XML[7].

Запис конфігурації в на приймально-контрольний пожежний пристрій реалізований простим поділенням файлу на пакети та послідовною передачею

пакетів даних. Таке налаштування можливе лише за умови повного знеструмлення пристрою та під'єднання його до ПК за допомогою USB кабелю. Вимкнення пристрою для внесення змін в конфігурацію системи не є безпечним для підконтрольного об'єкту.

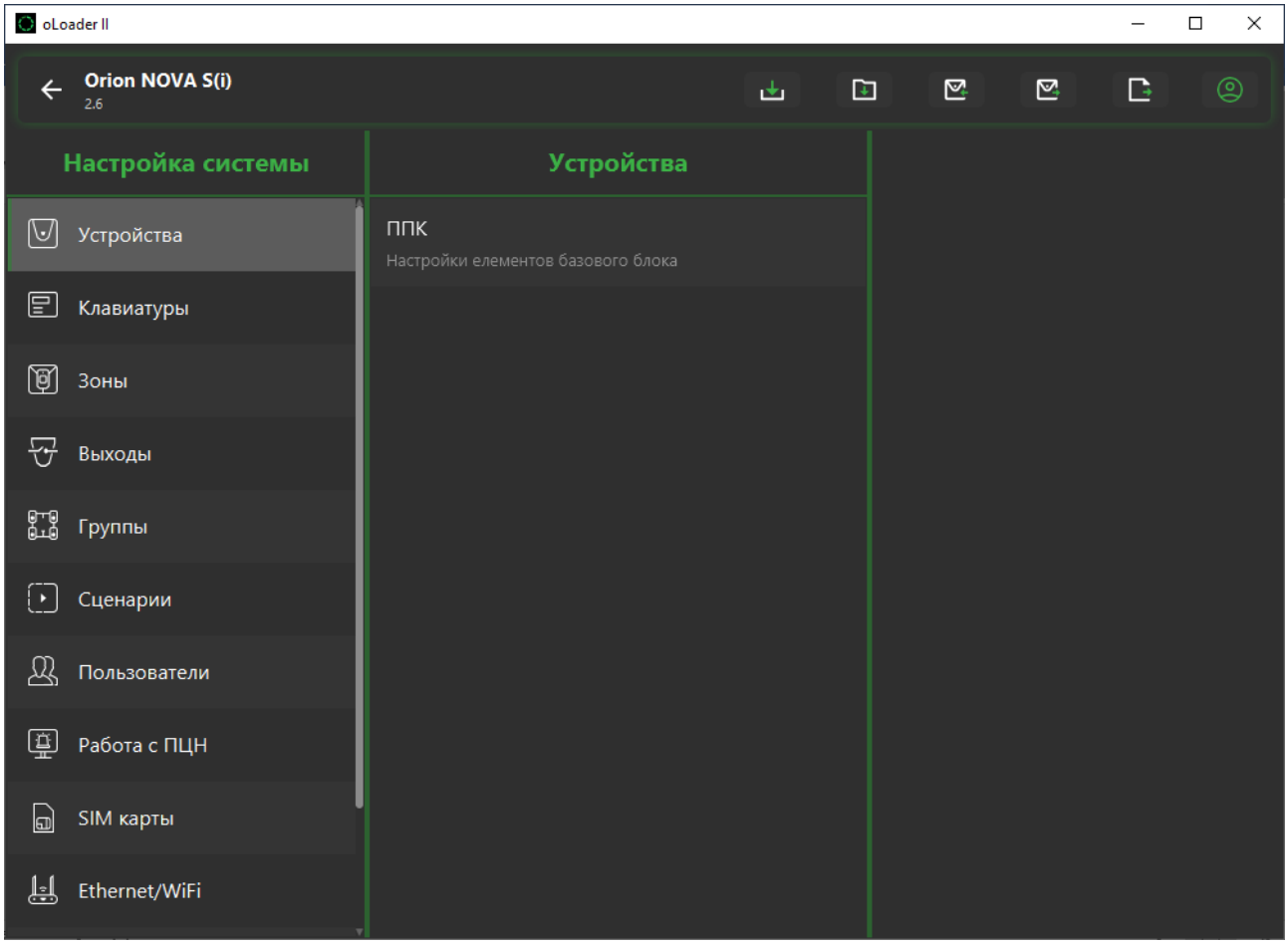


Рисунок 1.3 – Конфігуратор oLoaderII

aLoader - портативний десктопний додаток для конфігурування пожежних приймально-контрольних приладів окремої серії неадресних Tiras PrimeA. Перед створенням конфігурації потрібно вибрати зі списку прилад для якого вона буде створюватись і потім проводити налаштування. Є можливість вичитування налаштованої раніше конфігурації, відсутня авторизація рівня доступу. Вбудоване автоматичне оновлення ПЗ. Створена конфігурація записується у файл розширення XML та відсилається на прилад за особливим протоколом обміну TirasTransporter[8].

Запис конфігурації в на приймально-контрольний охоронний пристрій реалізований простим поділенням файлу на пакети та послідовною передачею пакетів даних. Таке налаштування можливе лише за умови повного знеструмлення пристрою та під'єднання його до ПК за допомогою USB кабелю. Вимкнення пристрою для внесення змін в конфігурацію системи не є безпечним для підконтрольного об'єкту.

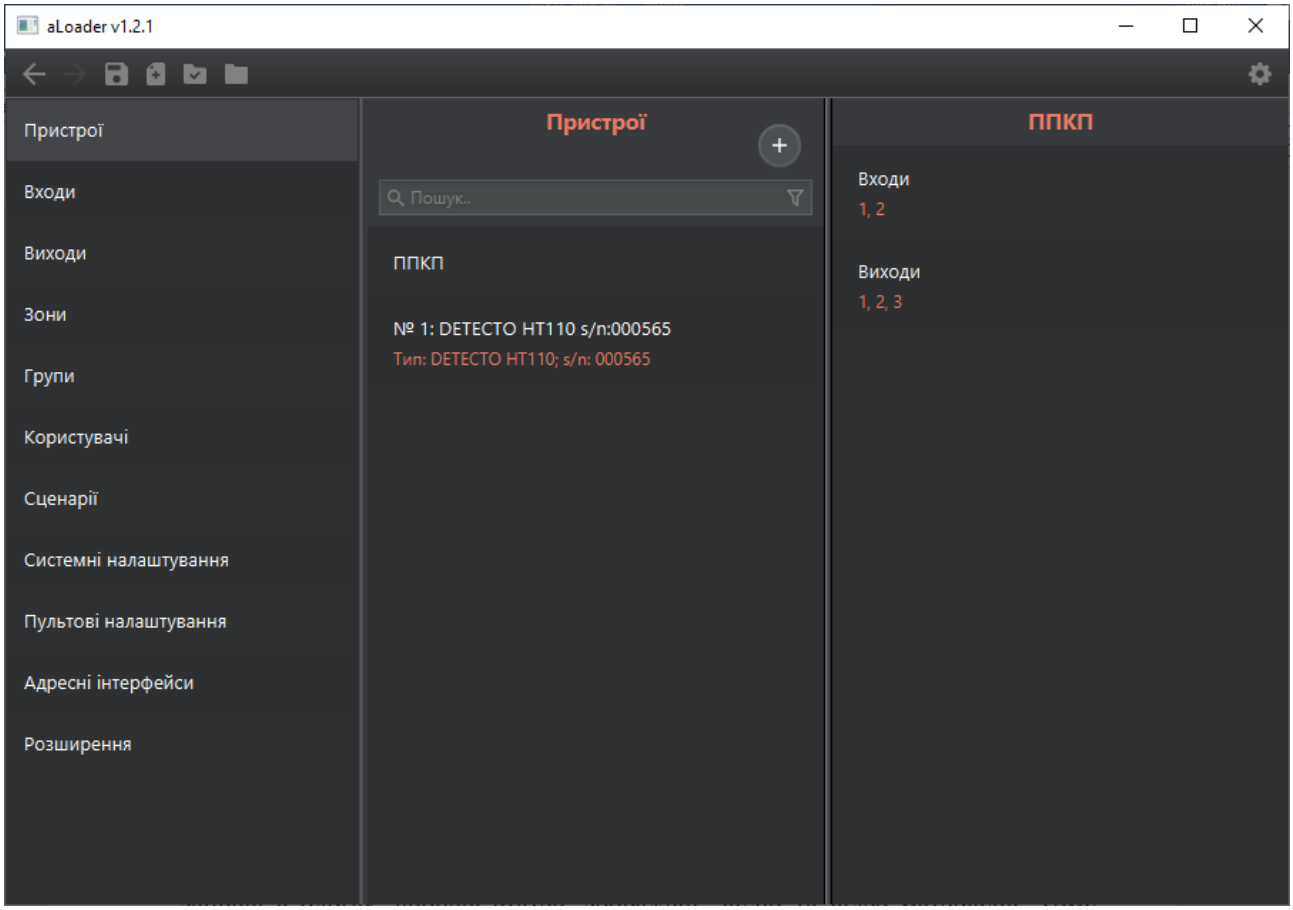


Рисунок 1.4 – Конфігуратор aLoader

Враховуючи недоліки зазначених вище програм можна зробити висновок, що розробка власної реалізації саме мобільного додатку буде доцільною. Буде розроблено мобільний Android додаток для конфігурування приймально-контрольних приладів.

1.3 Аналіз методів і засобів реалізації програмного продукту

Android підтримує різні периферійні пристрої USB і аксесуари USB для Android (аксесуари USB для Android, спеціальні апаратні пристрої, що підтримують протокол аксесуарів Android). При цьому Android може працювати в двох режимах: як аксесуар USB (Пристрій USB) і як хост USB (Головний шині USB, який управляє USB-пристроями).

У режимі аксесуара USB зовнішня апаратура USB працює як хост USB (зазвичай в такій ролі виступає комп'ютер, в порт USB якого підключено портом USB пристрій Android). Прикладами таких аксесуарів можуть служити роботизовані контролери, док-станції, діагностичне та музичне обладнання, кіоски, картрідер, і багато іншого. Це дає пристроїв на основі Android, які не мають можливостей хоста, засіб взаємодіяти з апаратурою USB.

Аксесуари Android USB повинні бути розроблені так, щоб щоб вони могли працювати з пристроями Android, з підтримкою протоколу обміну аксесуарів Android (Android accessory communication protocol). У режимі хоста Android-пристрій працює як хост. В цьому режимі воно може обмінюватися даними з таким пристроями USB, як цифрові камери, клавіатури, миші і ігрові контролери. Пристрої USB, які були розроблені для широкого спектра застосування, все ще можуть взаємодіяти з додатками Android[9].

Перш ніж почати, важливо зрозуміти класи, з якими потрібно працювати. У наведеній нижче таблиці описано API хостів USB у пакеті android.hardware.usb.

Таблиця 1.1 – USB Host APIs

Клас	Опис
UsbManager	Дозволяє перераховувати та спілкуватися з підключеними USB -пристроями.
UsbDevice	Представляє підключений USB -пристрій і містить методи доступу до його ідентифікаційної інформації, інтерфейсів та кінцевих точок.

Продовження таблиці 1.1

Клас	Опис
UsbInterface	Представляє інтерфейс USB -пристрою, який визначає набір функцій для пристрою. Пристрій може мати один або кілька інтерфейсів, за допомогою яких можна спілкуватися.
UsbEndpoint	Представляє кінцеву точку інтерфейсу, яка є каналом зв'язку для цього інтерфейсу. Інтерфейс може мати одну або кілька кінцевих точок і, як правило, має вхідні та вихідні кінцеві точки для двостороннього зв'язку з пристроєм.
UsbDeviceConnection	Представляє з'єднання з пристроєм, який передає дані про кінцеві точки. Цей клас дозволяє надсилати дані туди і назад синхронно або асинхронно.
UsbRequest	Представляє асинхронний запит на зв'язок із пристроєм через UsbDeviceConnection.

User Datagram Protocol, UDP — один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання. UDP — це один з найпростіших протоколів транспортного рівня моделі OSI, котрий виконує обмін повідомленнями (датаграмами) без підтвердження та гарантії доставки. При використанні протоколу UDP відповідальність за обробку помилок і повторну передачу даних покладена на протокол рівнем вище. Але попри всі недоліки, протокол UDP є ефективним для серверів, що надсилають невеликі відповіді великій кількості клієнтів[10].

На відміну від TCP UDP — дуже швидкий протокол, оскільки у ньому визначено мінімальний механізм, необхідний передачі даних. Звичайно, вона має деякі недоліки. Повідомлення надходять у будь-якому порядку, і те, що надіслано першим, може бути отримане останнім. Доставка повідомлень UDP зовсім не гарантується, повідомлення може загубитися, і можуть бути отримані дві копії одного повідомлення. Останній випадок виникає, якщо для надсилання повідомлень на одну адресу використовувати два різні маршрути.

UDP не вимагає відкрити з'єднання, і дані можуть бути надіслані відразу ж, як тільки вони підготовлені. UDP не надсилає підтвердження, тому дані можуть бути отримані або втрачені. Якщо під час використання UDP потрібна надійна передача даних, її слід реалізувати у протоколі вищого рівня.

Тож у чому переваги UDP, навіщо може знадобитися такий ненадійний протокол? Щоб зрозуміти причину використання UDP, потрібно розрізняти однонаправлену передачу, широкомовну передачу та групове розсилання.

Однонаправлене (unicast) повідомлення надсилається з одного вузла тільки один інший вузол. Це також називається зв'язком "крапка-крапка". Протокол TCP підтримує лише односпрямований зв'язок. Якщо серверу потрібно за допомогою TCP взаємодіяти з кількома клієнтами, кожен клієнт має встановити з'єднання, оскільки повідомлення можуть надсилатися лише одиночним вузлам.

Широкомовна передача (broadcast) означає, що повідомлення надсилається всім вузлам мережі. Групове розсилання (multicast) – це проміжний механізм: повідомлення надсилаються вибраним групам вузлів.

UDP може використовуватися для однонаправленого зв'язку, якщо потрібна швидка передача, наприклад для доставки мультимедійних даних, але головні переваги UDP стосуються широкомовної передачі та групового розсилання.

Зазвичай, коли ми надсилаємо широкомовні або групові повідомлення, не потрібно отримувати підтвердження з кожного вузла, оскільки сервер буде наповнений підтвердженнями, а завантаження мережі зросте занадто сильно. Прикладом широкомовної передачі служба часу.

Сервер часу відправляє широкомовне повідомлення, що містить поточний час, і будь-який хост, якщо захоче, може синхронізувати свій час з часом з широкомовного повідомлення[11].

Заголовок UDP набагато коротше і простіше заголовка TCP:

Таблиця 1.2 – Заголовок UDP пакету

Поле	Довжина	Опис
Порт адресанта	2 байти	Вказувати порту джерела для UDP не обов'язково. Якщо це поле використовується, одержувач може надіслати відповідь порту.
Порт адресата	2 байти	Номер порту призначення
Довжина	2 байти	Довжина повідомлення, враховуючи заголовок та дані.
Контрольна сума	2 байти	Контрольна сума заголовку та даних для перевірки.

UDP – це швидкий протокол, який не гарантує доставки. Якщо потрібна підтримка порядку повідомлень та надійна доставка, потрібно використовувати TCP. UDP головним чином призначений для широкомовної та групової передачі. Протокол UDP визначено RFC 786.

TCP і UDP є частиною набору протоколів TCP/IP, який включає низку протоколів для здійснення мережевого зв'язку.

TCP став домінуючим протоколом, який використовується для більшості інтернет-з’єднань завдяки його здатності розбивати великі набори даних на окремі пакети, перевіряти та повторно надсилати втрачені пакети та повторно збирати пакети в правильній послідовності. Але ці додаткові послуги коштують з точки зору додаткових витрат даних і затримок.

На відміну від цього, UDP вважається протоколом без з’єднання, оскільки він не вимагає створення віртуальної схеми перед перенесенням даних. Протокол зв’язку просто надсилає пакети, а це означає, що він має набагато меншу пропускну здатність і затримку. З UDP пакети можуть приймати різні шляхи між відправником і одержувачем. В результаті деякі пакети можуть бути втрачені або отримані не в порядку.

Характеристики UDP включають наступне:

- Це протокол без з'єднання.
- Він використовується для VoIP, потокового відео, ігор та прямих трансляцій.

- Це швидше і потребує менше ресурсів.
- Пакети не обов’язково надходять в порядку.
- Він дозволяє пропускати пакети - відправник не може знати, чи був отриманий пакет.
- Він краще підходить для програм, яким потрібна швидка та ефективна передача, наприклад ігор.

1.4 Постановка задач магістерської кваліфікаційної роботи

Після аналізу стану Android USB HID пристроїв, порівняння існуючих аналогів та аналізу методів і засобів реалізації програмного продукту було визначено наступні завдання, які необхідно виконати для розробки програмного продукту:

- розробити програмний додаток для конфігурування пожежних приймально-контрольних приладів;
- розробити модуль підтримки HID пристроїв;
- розробити модуль передавання потоку бітових даних;
- розробити модуль побудови конфігурації протипожежної системи;
- розробити модуль формування та транспортування даних за внутрішнім протоколом обміну;
- розробити зрозумілий інтерфейс програмного продукту;
- провести тестування програмного продукту.

1.5 Висновки

В даному розділі був розглянутий аналіз стану Android USB HID пристроїв який показав, що HID пристрої активно застосовуються в роботі з ОС Android. Були проаналізовані такі аналоги як: tLoader, tLoader Prime, oLoaderII, aLoader. При аналізі існуючих додатків для конфігурування приймально-контрольних приладів було акцентовано увагу на відсутність мобільних аналогів, тому розробка власного

додатку є цілком доцільною. Також було проведено аналіз методів реалізації програмного продукту. Були розглянуті методи для реалізації програмного продукту. В результаті цього було розроблено основні завдання, які необхідно виконати при розробці програмного додатку.

2 РОЗРОБКА МЕТОДУ СПОЛУЧЕННЯ ПРИЙМАЛЬНО-КОНТРОЛЬНИХ ПОЖЕЖНИХ ПРИЛАДІВ ДЛЯ ЗОВНІШНЬОГО КОНФІГУРУВАННЯ

2.1 Аналіз прикладного забезпечення системи

Прикладні системи утворюють рівень програмного забезпечення, що надається користувачеві для розв'язання своїх задач. Процедури інформаційних технологій спрямовуються на обробку інформації певного класу (даних, тексту, графіки, об'єктів реального світу) і реалізуються за допомогою програмних комплексів різного рівня, складності та призначення. Прикладне програмне забезпечення призначене для розв'язування конкретних задач користувача й організації обчислювального процесу інформаційної системи загалом.

Розрізняють кілька основних класів прикладних систем, що використовуються на персональних комп'ютерах: прикладні пакети і програми загального призначення (універсальні); пакети і програми проблемозорієнтовані; глобальних мереж; методозорієнтовані; організації (адміністрування) обчислювального процесу.

Системне програмне забезпечення (СПЗ) – це ряд програм, які використовуються для апаратних засобів, що призначена для розподілення програмі між собою та відокремлення інших програм від безпосередньої взаємодії з обладнанням та організації процесу обробки інформації у комп'ютері. До СПЗ належать такі типи програм, як операційні системи (ОС), різні сервісні засоби, що функціонально доповнюють можливості ОС, інструментальні засоби (системи керування базами даних, мови програмування, оболонки експертних систем). Прикладне ПЗ призначено для розв'язання певних завдань користувача[12].

Розроблюваний додаток «aLoader» буде працювати з HID-пристроями, отримувати та передавати дані за розробленим протоколом обміну. Налаштування конфігурації системи може проходити без постійного з’єднання з самим пристроєм. Фізичне з’єднання потрібне тільки для запису або ж вичитування файлу конфігурації системи. Всі дані зберігатимуться в локальній базі даних.

2.2 Розробка загальної моделі системи

Загальна модель системи зображена на рисунку 2.1.

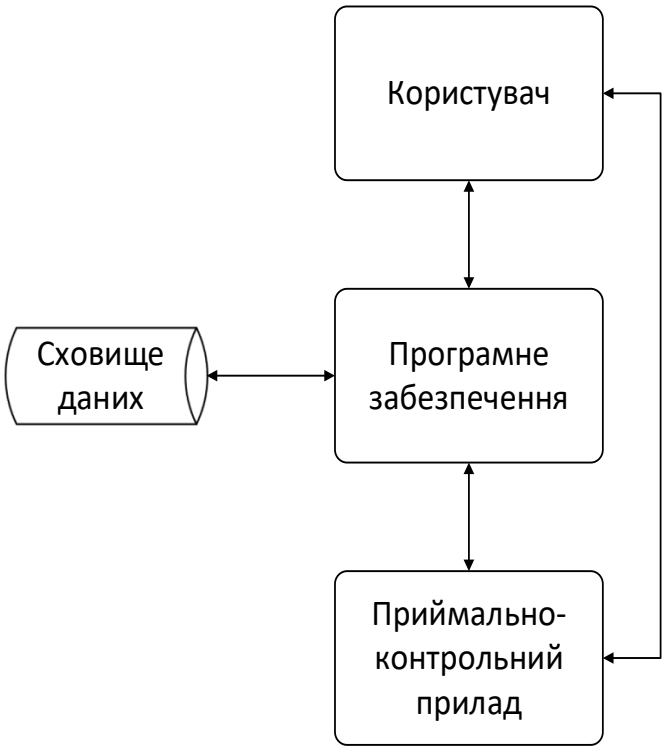


Рисунок 2.1 – Загальна модель роботи додатку

Додаток-конфігуратор представляє собою локальний мобільний додаток з внутрішньою базою даних. Всі зміни в конфігурацію пристрою вносяться не динамічно, за для забезпечення правильної та безперешкодної роботи безпосередньо, самого конфігурованого приймально-контрольного пристрою. Так як головна його задача зберігати підконтрольний об’єкт в пожежній безпеці.

Працювати з конфігурацією може тільки один користувач, для початкового налаштування авторизація не потрібна. Тільки при записуванні або вичитуванні конфігурації потрібно вже авторизуватись від імені користувача з правами «Інсталятор» (3-й рівень доступу).

Взаємодія Android-пристрою з HID-пристроєм відбувається за розробленим протоколом “TirasTransporter”, в якості транспортного рівня використовується технологія UDP, для гарантування передачі повідомлень між двома вузлами.

2.3 Розробка алгоритму взаємодії Android з HID-пристроєм

Для спілкування через USB на периферійному пристрої необхідно реалізувати інтерфейс взаємодії. Різні функції (наприклад, USB HID, USB Mass Strorage або USB CDC) будуть реалізовувати свої інтерфейси, деякі будуть мати кілька інтерфейсів. Кожен інтерфейс містить у собі кінцеві точки — спеціальні канали зв'язку, свого роду буфери обміну[13].

На даному приймально-контрольному пожежному пристрої реалізований Custom HID з одним інтерфейсом і двома кінцевими точками, однією для прийому, інший для передачі. Зазвичай інформація з існуючими на пристрої інтерфейсами і кінцевими точками написана специфікації на пристрій.

Пристрої USB HID спілкуються через репорти. Так як дані передаються через кінцеві точки, тобто, обов’язкова необхідність ідентифікувати, а також розпарсувати у відповідність до протоколу. Пристрої не просто кидають один одному байти даних, а обмінюються пакетами, що мають чітко визначену структуру, яка описується на пристрої у спеціальному репортовому дескрипторі.

Таким чином, за дескриптором репорту, визначається який ідентифікатор, структура, розмір та частоту передачі мають ті чи інші дані. Ідентифікація пакета відбувається за першим байтом, який є ID репорту. Наприклад дані про стан кнопки, йдуть у репорт з ID = 1, а світлодіодом ми керуємо через репорт з ID = 2.

Для отримання даних необхідно знати розмір даних, які можна, як знати заздалегідь, так і отримати з кінцевої точки[14].

Метод отримання даних USB HID є синхронним і блокуючим і виконувати його необхідно в іншому потоці, крім того, репорти від пристрою можуть приходити постійно, або в будь-який час, тому необхідно реалізувати постійне опитування репорту, щоб не пропустити дані.

У Android підтримка USB пристроїв з'явилася починаючи з API версії 12 (Android 3.1) Для роботи з периферійним пристроєм необхідно реалізувати режим USB-хост. Робота з USB досить не погано описана в документації.

Для початку необхідно ідентифікувати підключений пристрій, серед усього різноманітності USB-девайсів. USB-девайси ідентифікуються по поєднанню vid (ідентифікатор постачальника) та pid (ідентифікатор продукту).

Протокол шини USB забезпечує обмін даними між хостом та пристроєм. На протокольному рівні вирішуються такі завдання, як забезпечення достовірності та надійності передачі, керування потоком. Весь трафік на шині USB передається за допомогою транзакцій, у кожній транзакції можливий обмін тільки між хостом і пристроєм, що адресується (його кінцевою точкою)[15].

Усі транзакції (обміни) з пристроями USB складаються із двох-трьох пакетів, типові послідовності пакетів у транзакціях наведено на рисунку 2.2. Кожна транзакція планується і починається з ініціативи хост-контролера, який надсилає пакет-маркер транзакції (token packet). Маркер транзакції описує тип і напрямок передачі, адресу вибраного пристрою USB та номер кінцевої точки. Пристрій, що адресується маркером, розпізнає свою адресу і готується до обміну. Джерело даних, визначене маркером, передає пакет даних. На цьому етапі транзакції, що стосуються ізохронних передач, завершуються - тут немає підтвердження прийому пакетів[16].

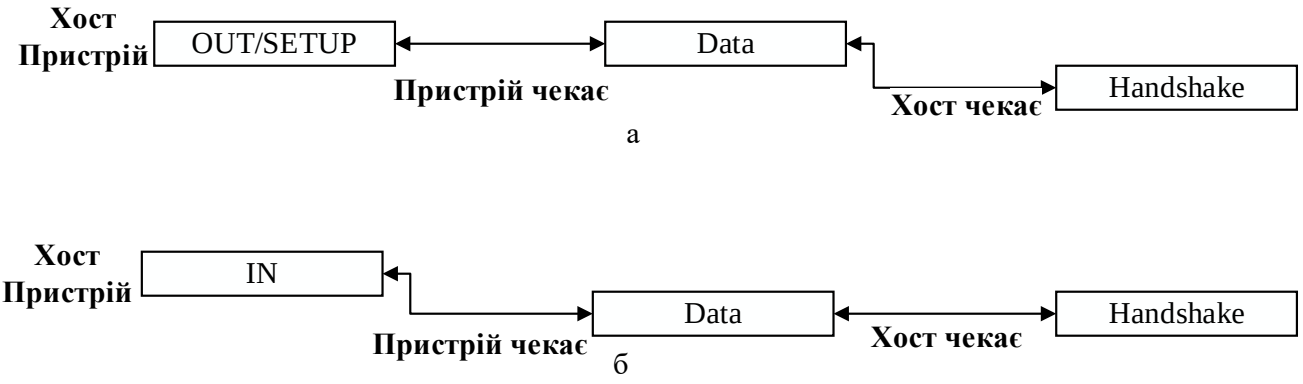


Рисунок 2.2 – Послідовність пакетів в транзакціях на шині USB.

а – вихід; б – вхід даних

Для інших типів передач працює механізм підтвердження, що забезпечує гарантовану доставку даних. Формати пакетів наведено на рисунку 2.3, типи пакетів - у таблиці. У всіх полях пакетів, крім поля CRC, дані передаються молодшим бітом вперед (на тимчасових діаграмах молодший біт зображується зліва).

Пакет починається з синхрослідковості Sync і завершується ознакою кінця - EOP. Тип пакета визначається полем PID. Призначення інших полів відкривається далі. Довжина полів Sync та EOP вказана для передач на FS/LS, для високошвидкісних передач поле Sync подовжено до 32 бітових інтервалів, а EOP до 8 (у пакетах SOF поле EOP має довжину 40 біт).

Sync	PID	Check	Addr	EndP	CRC	EOP
8	4	4	7	4	5	2

а

Sync	PID	Check	Data	CRC	EOP
8	4	4	8xn	16	2

б

Sync	PID	Check	EOP
8	4	4	2

в

Рисунок 2.3 – Формат пакетів USB:
а – маркер; б – пакет даних; в – пакет квитування

Всі пакети, що приймаються, перевіряються на наявність помилок, що дозволяють прийняті формати пакета і деякі угоди:

- пакет починається з синхронізуючої послідовності, за якою слідує його ідентифікатор PID (Packet Identificator). За ідентифікатором слідує його інверсна копія — Check. Розбіжність двох копій вважається ознакою помилки;
- тіло пакета (всі поля пакету, крім PID та ознака EOP) захищається CRC кодом: 5-бітним для пакетів-маркерів, 16-бітним — для пакетів даних. Розбіжність CRC з очікуваним значенням вважається ознакою помилки;

- пакет завершується спеціальним сигналом EOP; якщо в пакеті виявляється не ціле число байт, він вважається хибним. Помилковий EOP, навіть на межі байта, не дозволить прийняти пакет через практично неминучу в цій ситуації помилку щодо CRC-контролю;
- на фізичний рівень (в шину) дані пакета передаються з використанням вставки біт (bit stuffing, після шести одиничних біт вставляється нулик), що запобігає втраті бітової синхронізації при монотонному сигналі. Прийом понад шість одиничних біт поспіль вважається помилкою (на HS — ознакою кінця кадру).

Виявлення будь-якої з перелічених помилок у пакеті змушує приймач вважати його недійсним. На пакети, прийняті помилково, ні пристрій, ні хост-контролер ніяк не відповідають. При ізохронній передачі дані недійсного пакета повинні просто ігноруватися (вони губляться); інших типів передач використовуються засоби забезпечення надійної доставки.

Для виявлення відсутності відповіді партнера на пакет кожен пристрій має лічильник тайм-ауту, який перериває очікування відповіді через деякий час. У USB є обмеження на час обороту по шині (roundtrip time): час від кінця EOP сформованого пакета до отримання початку пакету у відповідь.

Для кінцевого пристрою (і хост-контролера) унормується максимальна затримка відповіді (response time) від кінця побаченого EOP до введення ним початку пакета. Для хабів нормується затримка трансляції пакетів, кабелів — затримка поширення сигналів. Лічильник тайм-ауту повинен враховувати максимальну затримку, можливу для допустимої конфігурації шини: до 5 проміжних хабів, 5 метрів кожен кабель.

Для підтвердження прийому, управління потоком та сигналізації помилок використовуються пакети квітування (handshake packets). З цих пакетів хост-контролер може надсилати пристрою лише пакет ACK, що підтверджує безпомилковий прийом пакета даних. Пристрій для відповіді хосту використовує такі пакети квитування:

- ACK - підтвердження (позитивне) успішного виконання транзакції виведення або керування;
- NAK - негативне підтвердження, є ознакою неготовності пристрою до виконання цієї транзакції (немає даних для передачі хосту, відсутнє місце в буфері для прийому, не завершено операцію управління). Це є нормальною відповіддю, про яку не дізнається ніхто, крім хост-контролера, змушеного повторити цю транзакцію пізніше. У транзакціях введення відповідь NAK пристрій дає замість пакета даних, якщо вони готові;
- STALL — повідомлення про серйозну помилку, яке означає, що без спеціального програмного втручання робота з кінцевою точкою стає неможливою. Ця відповідь доводиться до відома і драйвера USBD, що скасовує подальші транзакції з цією точкою, і до клієнтського драйвера, від якого очікується програмне втручання, що розблокує точку. У керуючих транзакціях (Control) відповідь STALL означає нездійсненність цього запиту; розблокування точки при цьому не потрібне.

Управління потоком при виведенні даних, засноване тільки на можливості відповіді NAK у разі неготовності пристрою, вельми неефективно витрачає пропускну здатність шини: щоб переконатися в неготовності пристрою, марно по шині передається великий пакет даних. У USB 2.0 цієї проблеми в транзакціях Bulk-OUT і Control уникають, застосувавши протокол проб (Ping Protocol)[17].

Хост може опитати готовність пристрою до прийому пакета максимального розміру, надіславши йому маркер-пробник PING. На цей маркер пристрій може відповісти підтвердженням ACK (при готовності) або NAK (якщо не може прийняти пакет максимального розміру). Негативна відповідь змусить хост повторити пробу пізніше, позитивний дозволить йому виконати транзакцію виведення даних

2.4 Розробка загального алгоритму роботи програми

На початку роботи у користувача є вибір як почати конфігурування приймально-контрольного пристрою, з початковою авторизацією та налаштуванням конфігурації або ж спочатку налаштування конфігурації і авторизації вже безпосередньо при записі конфігурації в прилад. Алгоритм авторизації однаковий, але має деякі розбіжності у послідовності.

У випадку коли авторизація відбувається перед створенням конфігурації відтворюється наступний алгоритм, як на рисунку 2.4.

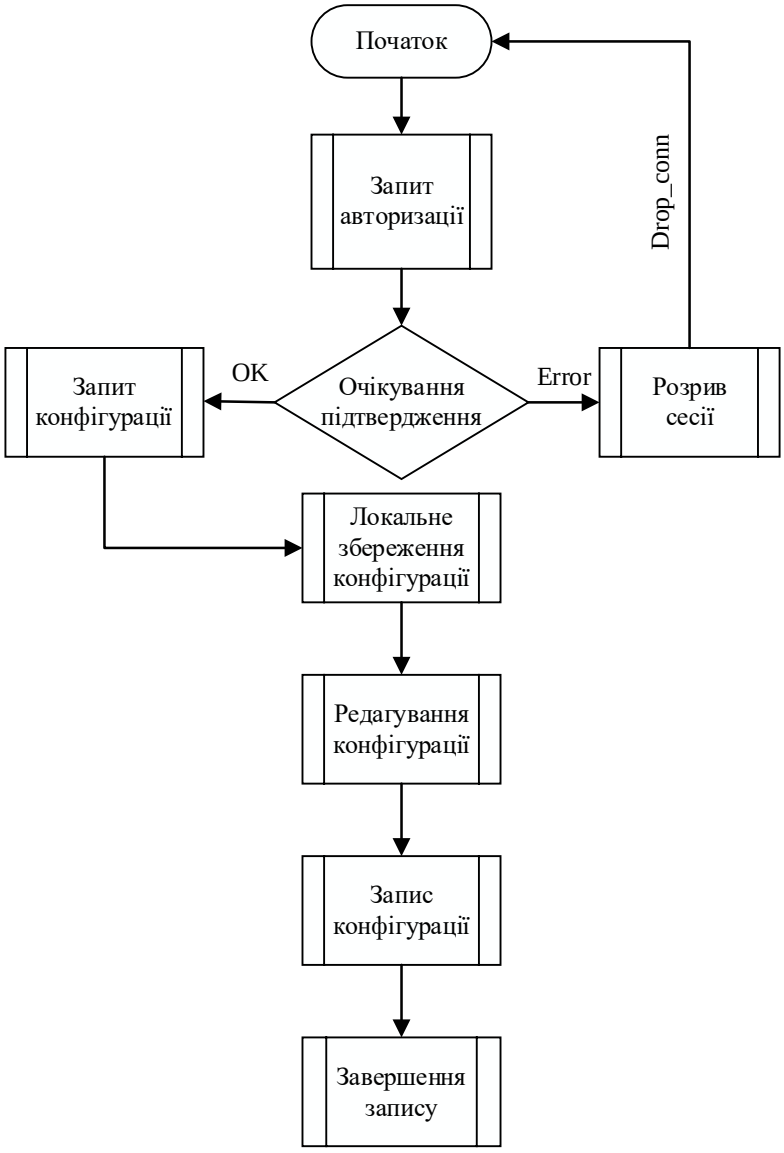


Рисунок 2.4 – Робота програми з початковою авторизацією

У випадку коли спочатку користувач створює конфігурацію без попередньо приєднаного пристрою, алгоритм відмінний від попереднього. Авторизація проходить перед записом конфігурації, рисунок 2.5.

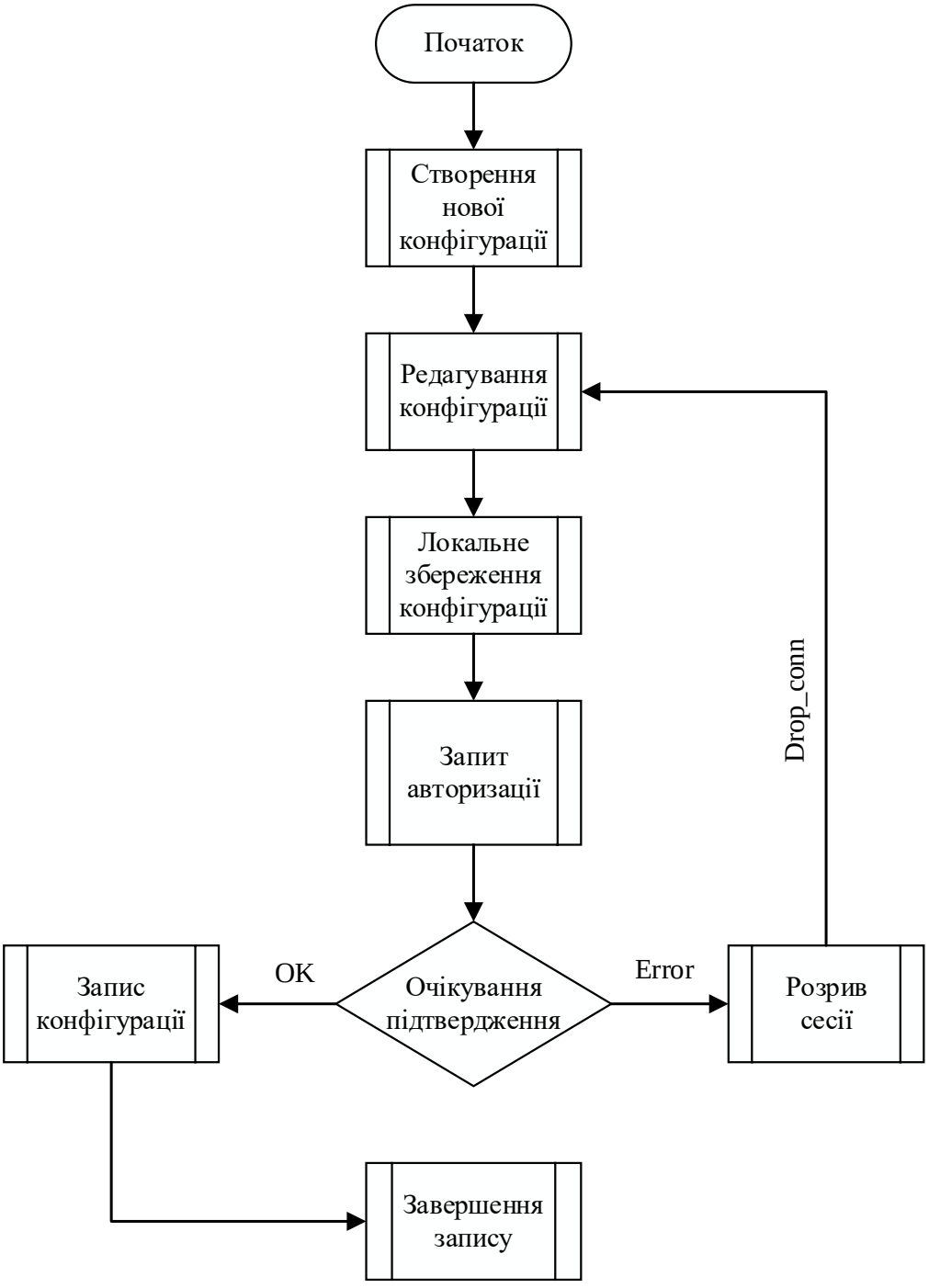


Рисунок 2.5 – Робота програми з авторизацією після налаштування

2.5 Розробка методу серіалізації об’єктів у вихідний потік

Для побудови даних та подальшого обміну між Android та HID-пристроєм було розроблено окремий модуль, завдяки модульному розділенні додаток буде мати високу надійність та стабільність роботи.

Після проведення всіх необхідних налаштувань конфігурації її потрібно структурувати та підготувати до обміну транспортними пакетами відповідно до протоколу передачі «TirasTransporter».

Вимоги до протоколу:

- Надійна доставка пакетів, реалізована через механізм позитивного зворотного зв'язку (так званий positive acknowledgment);
- Необхідність ефективної передачі даних, тобто. протокол повинен уникати зайвих ретрансляцій пакетів;
- Повинна бути можливість скасування механізму підтвердження доставки (можливість функціонувати як «чистий» протокол UDP);
- Можливість реалізації командного режиму, з підтвердженням кожного повідомлення;
- Базовою одиницею передачі даних за протоколом має бути повідомлення.

UDP не робить жодних кроків щодо виявлення втрат. Контроль помилок передачі в протоколі UDP повністю покладається на додаток.

Виявлення помилок у протоколі TCP досягається завдяки установці з'єднання з кінцевим вузлом, збереженню стану цього з'єднання, вказівці номера відправлених байт у кожному заголовку пакета, і повідомлення про отримання за допомогою номера підтвердження «acknowledge number».

Додатково, для підвищення продуктивності (тобто відправки більше одного сегмента без отримання підтвердження) TCP протокол використовує так зване вікно передачі - число байт даних, які відправник сегмента очікує прийняти.

З вищеописаного, зрозуміло, що для створення надійного протоколу доставки повідомлень поверх UDP (надалі називатимемо Reliable UDP), потрібно реалізувати схожі з TCP механізми передачі даних. А саме:

- зберігати стан з'єднання;
- використовувати нумерацію сегментів;
- використовувати спеціальні пакети підтвердження;
- використовувати спрощений механізм вікна, для збільшення пропускної спроможності протоколу.

Додатково потрібно:

- сигналізувати про початок повідомлення, виділення ресурсів під з'єднання;
- сигналізувати про закінчення повідомлення, передачі отриманого повідомлення вищому додатку і вивільнення ресурсів протоколу;
- дозволити протоколу для конкретних з'єднань відключати механізм підтвердження доставки, щоб функціонувати як «чистий» UDP[18].

Структура заголовку Reliable UDP на рисунку 2.6.

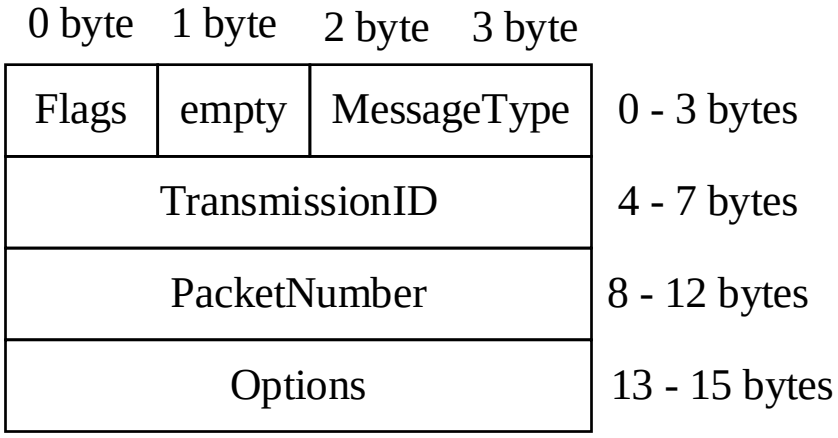


Рисунок 2.6 – Заголовок UDP пакету

Flags – керуючі прапори пакету.

MessageType – тип повідомлення, що використовується вищими програмами, для передплати певних повідомлень.

TransmissionId — номер передачі, разом із адресою та портом одержувача унікально визначає з'єднання.

PacketNumber – номер пакету.

Options – додаткові опції протоколу. У разі першого пакета використовується для вказівки розміру повідомлення.

Оскільки Reliable UDP орієнтований на гарантовану передачу повідомлення між двома вузлами, він має вміти встановлювати з'єднання з іншою стороною. Для встановлення з'єднання сторона-відправник посилає пакет із прапором FirstPacket, відповідь на який означатиме встановлення з'єднання. Всі пакети у відповідь, або, по-іншому, пакети підтвердження, завжди виставляють значення поля PacketNumber на одиницю більше, ніж найбільше значення PacketNumber у пакетів, що успішно прийшли. У полі Options для першого надісланого пакета записується розмір повідомлення.

Для завершення з'єднання використовується подібний механізм. В останньому пакеті повідомлення встановлюється прапорець LastPacket. У пакеті у відповідь вказується номер останнього пакета + 1, що для приймальної сторони означає успішну доставку повідомлення[19].

Коли з'єднання встановлено, починається надсилання даних. Дані передаються блоками пакетів. Кожен блок, крім останнього, містить фіксовану кількість пакетів. Воно дорівнює розміру вікна прийому/передачі. Останній блок даних може мати менше пакетів.

Після відправлення кожного блоку, сторона-відправник очікує на підтвердження про доставку, або запиту на повторну доставку втрачених пакетів, залишаючи відкритим вікно прийому/передачі для отримання відповідей. Після отримання підтвердження про доставку блоку, вікно прийому/передачі зсувається і відправляється наступний блок даних.

Діаграма обміну за протоколом UDP наведена на рисунку 2.7.

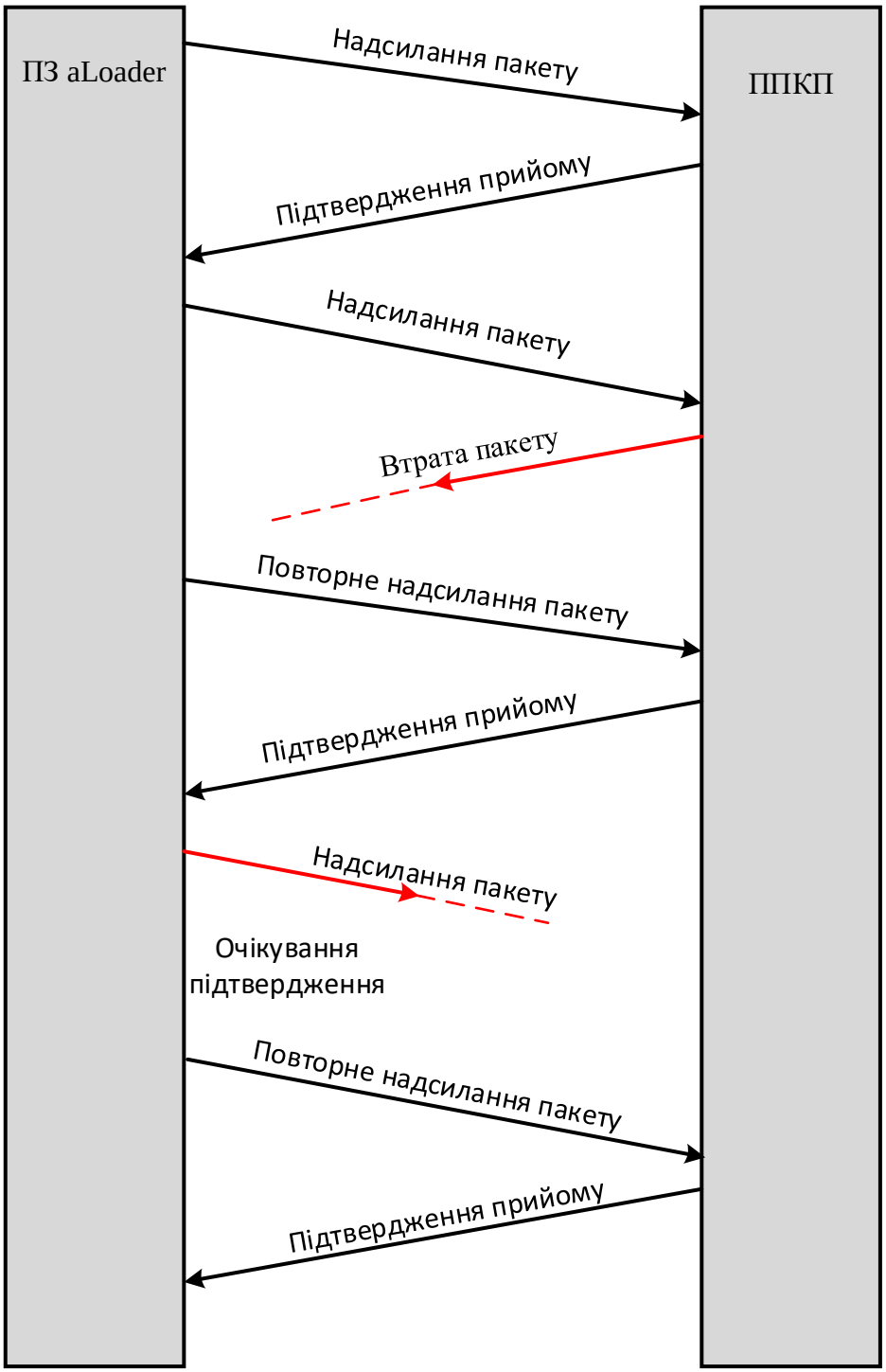


Рисунок 2.7 - Обміну за протоколом UDP

Існує кілька причин, через які не може бути встановлено з'єднання. Наприклад, якщо сторона, що приймає, поза мережею. У такому разі, при спробі встановити з'єднання, з'єднання буде закрито за тайм-аутом. У реалізації Reliable UDP використовуються два таймери для встановлення тайм-аутів. Перший, робочий таймер, служить для очікування відповіді віддаленого хоста. Якщо він спрацьовує на стороні відправника, то виконується повторне відправлення останнього відправленого пакета. Якщо ж таймер спрацьовує в одержувача, то перевіряється на втрачені пакети і надсилаються запити на повторну доставку.

Другий таймер – необхідний закриття з'єднання у разі відсутності зв'язку між вузлами. Для сторони-відправника він запускається відразу після спрацьовування робочого таймера і чекає відповіді від віддаленого вузла. У разі відсутності відповіді за встановлений період – з'єднання завершується та ресурси звільняються. Для сторони-одержувача таймер закриття з'єднання запускається після подвійного спрацьовування робочого таймера. Це необхідно для страховки від втрати пакета підтвердження. При спрацьовуванні таймера також завершується з'єднання і вивільняються ресурси.

Принципи роботи протоколу реалізовані кінцевому автоматі, кожен стан якого відповідає певну логіку обробки пакетів.

Closed – насправді не є станом, це стартова та кінцева точка для автомата. За стан Closed приймається блок управління передачею, який, реалізуючи асинхронний UDP сервер, перенаправляє пакети у відповідні з'єднання та запускає обробку станів.

FirstPacketSending – початковий стан, у якому знаходиться вихідне з'єднання під час надсилання повідомлення.

У цьому стані надсилається перший пакет для звичайних повідомлень. Для повідомлень без підтвердження надсилання, це єдиний стан – у ньому відбувається надсилання всього повідомлення.

SendingCycle – основний стан передачі пакетів повідомлення. Перехід у нього зі стану FirstPacketSending здійснюється після надсилання першого пакета повідомлення. Саме в цей стан приходять усі підтвердження та запити на повторні

передачі. Вихід із нього можливий у двох випадках – у разі успішної доставки повідомлення або за тайм-аутом.

FirstPacketReceived – початковий стан одержувача повідомлення. У ньому перевіряється коректність початку передачі, створюються необхідні структури, і надсилається підтвердження прийом першого пакета.

Для повідомлення, що складається з єдиного пакета та надісланого без використання підтвердження доставки, це єдиний стан. Після обробки повідомлення з'єднання закривається.

Assembling – основний стан прийому пакетів повідомлення.

У ньому проводиться запис пакетів у тимчасове сховище, перевірка відсутність втрат пакетів, відправка підтверджень про доставку блоку пакетів і повідомлення повністю, і відправлення запитів на повторну доставку втрачених пакетів. У разі успішного отримання всього повідомлення – з'єднання переходить у стан Completed, інакше виконується вихід тайм-ауту.

Completed – закриття з'єднання у разі успішного отримання повідомлення.

Цей стан потрібний для складання повідомлення та випадку, коли підтвердження про доставку повідомлення було втрачено на шляху до відправника. Вихід із цього стану проводиться за тайм-аутом, але з'єднання вважається успішно закритим[20].

Один із ключових елементів Reliable UDP – блок управління передачею. Завдання даного блоку - зберігання поточних з'єднань і допоміжних елементів, розподіл пакетів, що прийшли по відповідним з'єднанням, надання інтерфейсу для відправки пакетів з'єднанню і реалізація API протоколу. Блок управління передачею приймає пакети від UDP рівня та перенаправляє їх на обробку в кінцевий автомат. Для прийому пакетів у ньому реалізовано асинхронний UDP сервер.

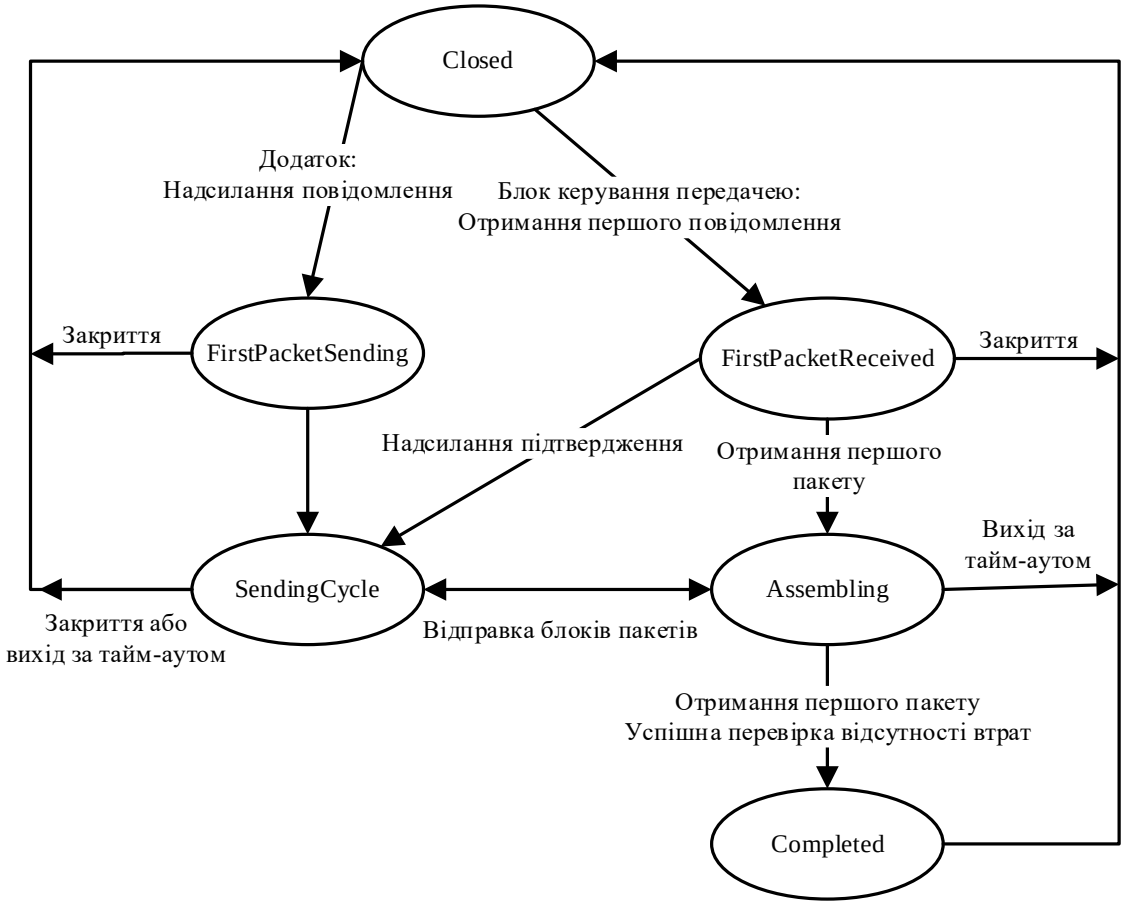


Рисунок 2.8 – Діаграма станів Reliable UDP.

2.6 Висновки

В даному розділі було описано загальний алгоритм роботи програми та алгоритм модуля обміну. Модель роботи програми побудовано на основі клієнт-серверної архітектури, де вся бізнес логіка працює на ПК(сервері), а дані зберігаються на приймально-контрольному пристрою або в збережено локальному XML-файлі конфігурації.

3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ КОНФІГУРУВАННЯ ПРИЙМАЛЬНО-КОНТРОЛЬНОГО ПРИСТРОЮ

3.1 Варіантний аналіз і обґрунтування вибору засобів реалізації програмного засобу

Існує безліч абсолютно різних способів використання мов програмування, адже кожен інструмент розрахований на виконання тих задач, на яких поставили його їх розробники. Як і при виборі інструмента, кожна мова програмування розрахована на виконання тих задач, на яких вона спроможна дати рішення[18]. Тому визначимо оптимальний варіант і використаємо його при розробці програмного засобу.

Програмний засіб розрахований на запуск у операційній системі Android, це означає, що такі мови програмування, як Java, C++, Python розраховані на створення додатків на Linux, MacOS, Android та iOS допоможуть у вирішенні поставленої задачі. При цьому не допоможуть мови JavaScript, PHP та Ruby, що розраховані на створення застосунків в браузерах. Так як, програмний засіб повинен бути реалізований у простому форматі, з можливістю підключення бази даних, мати графічний інтерфейс та при розробці витратити якомога меншу кількість годин, розглядати необхідно наступні мови програмування – C++, C#, Java та. Розглянемо їх детально та переглянемо їх переваги та недоліки [21].

1) Особливості мови Java:

– призначений для розподілених обчислень;

– легко компілювати і налагоджувати програмні застосунки;

– високий ступінь стійкості;

– багатопоточна, здатна одночасно виконувати різні завдання в межах програми;

– об'єктно-орієнтована, дозволяє створювати модульні програми і багаторазовий код;

– незалежний від платформи, легко мігрувати з однієї системи в іншу;

– підтримує автоматичне виділення пам'яті та збір сміття [22].

2) Особливості мови C++ / C#:

– широкий спектр застосувань, від простих додатків до графічних інтерфейсів до яскравих 3d-ігор та математичного моделювання в реальному часі;

– ефективний, швидкий і потужний;

– дуже портативний, найкращий вибір для кросплатформеної розробки;

– об'єктно-орієнтовані класи мови програмування, абстракція даних та інкапсуляція, успадкування та поліморфізм;

– багаті бібліотеки функцій;

– підтримує обробку виключень і перевантаження функцій .

3) Особливості мови Python:

– Активна підтримка спільнот та швидкий розвиток;

– легкий в освоєнні, зрозумілий синтаксис;

– розширюється більшою мірою;

– вільний, з відкритим вихідним кодом, кросплатформенний;

– високорівнева мова з високою читабельністю і надійністю;

– об'єктно-орієнтована;

– програми python легко модифікувати для підтримки різних рушіїв бази даних;

– використовується для створення прототипів і тестування коду, який пізніше буде реалізовано з використанням інших мов програмування [23].

Найбільшою перевагою використання Java над C++ є портативність. Код вимагає перекомпіляції для кожної платформи, на яку призначена програма C++. Це не стосується байт-коду Java, який працює на різних операційних системах. Незважаючи на те, що C++ підтримує перевантаження оператора, в Java немає положення про те, що це відбувається. З іншого боку, Java підтримує коментарі щодо документації. Цього немає в C++.

Java поставляється з вбудованою підтримкою потоків, в той час як немає підтримки потоків у C++. Хоча Java має багаті бібліотеки, з широким діапазоном класів, C++ постачається з бібліотеками з низькими можливостями. Обидві функції і змінні можуть знаходитися всередині класів Java. Проте, те ж саме перебувають

поза класами в C++.

Хоча обидва використовуються для сценаріїв програмування загального призначення, існує багато відмінностей між C++ і Python. Завдяки філософії дизайну WORA (Write Once, Run Anywhere), код Python виконується на всіх операційних системах, на яких встановлено Python. Проте це не стосується C++, що вимагає повторної компіляції кожного разу, коли код потрібно запускати на машині з іншою операційною системою.

На відміну від C++, змінну можна використовувати безпосередньо без необхідності декларування в коді Python. C++ в значній мірі використовує покажчики і не пропонує збір сміття. Отже, він схильний до витоку пам'яті. Для ефективного управління пам'яттю, Python поставляється з вбудованим динамічним виділенням пам'яті та функціями збору сміття.

Можливо, найбільшою перевагою використання Java над Python є швидкість. Код Java набагато швидше, ніж код Python. Java сильно навантажена. Це означає, що потрібно визначити точний тип даних змінних. Це не той випадок у Python, оскільки він є динамічним, тобто не потрібно визначати точний тип даних змінних. Python є інтерпретованою мовою програмування. Навпаки, Java є складеною мовою програмування.

Для поставленої задачі, найкращою та оптимальною мовою програмування доцільно використати Java через те що більшість програм на операційній системі Android написані мовою Java, також дана мова є об’єктно-орієнтованою, для написання програм для мобільних пристроїв, а програмний засіб повинен мати можливість відображати інформацію у графічному інтерфейсі та працювати з пам‘яттю.

Програмний додаток розрахований на розміщення в мобільному пристрої користувача з операційною системою Android. Також вимагає доступ до пам’яті мобільного пристрою.

Коли справа доходить до вибору середовища розробки для створення Java-додатків, є кілька варіантів - Eclipse, NetBeans і IntelliJ IDEA Community Edition (доступна під ліцензією Apache 2.0) та ін. Всі ці середовища розробки

поширюються з відкритим вихідним кодом. Переваги використання цих середовищ розробки очевидні:

- безкоштовне розповсюдження;
- можливість додавання нових функцій і отримання відповідно нових можливостей;
- можливість поліпшення IDE.

Чим більше людей, які хочуть ввести поліпшення, тим легше тестувати нові функції, і, отже, середовище розробки містить менше помилок у роботі. На сьогоднішній день найпопулярнішими є Eclipse і IntelliJ IDEA. Загалом всі вони мають приблизно однакові функціональні можливості, і досить важко оцінити яке з них є кращим. Стандартні можливості вищевказаних IDEs: підсвічування синтаксису; компіляція коду; система підказок; автодоповнення; інтеграція з бібліотеками і програмними каркасами; можливість рефакторингу; автогенерація коду; налагоджувач коду; перевірка помилок; компіляція.

Йдуть постійні суперечки про те, що ж краще Eclipse і IntelliJ IDEA. IDE приблизно однакові за своїми можливостями, і вибір однієї з них - це справа смаку, але все-таки ми постараємося визначити, яке з середовищ є кращим для написання програм мовою Java.

IntelliJ IDEA – інтегроване середовище розробки програмного забезпечення багатьма мовами програмування. Community версія середовища IntelliJ IDEA підтримує інструменти для проведення тестування TestNG і JUnit, системи контролю версій CVS, Subversion, Mercurial і Git, засоби збирання Maven і Ant, мови програмування Java, 74 Foss Lviv 2013 Java ME, Scala, Clojure і Groovy. Середовище містить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, XML редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse.

Android Studio – є офіційним IDE для розробки додатків Android, на основі IntelliJ IDEA. На вершині можливостей, які ви очікуєте від IntelliJ, Android Studio пропонує:

- гнучку Gradle-система збірки;
- побудова варіантів програми на кілька АПК покоління;
- шаблони коду, щоб допомогти побудувати загальні риси додатку;
- багатий редактор макетів з підтримкою перетягування і падіння редагування теми;
- інструменти, щоб фіксувати продуктивність, зручність використання, сумісність версії, і інші проблеми;
- вбудована підтримка для Google Cloud Platform, що дозволяє легко інтегрувати Google Cloud повідомленнями [24].

Android SDK - включає в себе різноманітні бібліотеки, документацію та інструменти, які допомагають розробляти мобільні додатки для платформи Android.

API Android SDK-API бібліотеки Android, що надаються для розробки додатків.

Документація SDK-включає велику довідкову інформацію, що деталізує, що включено в кожен пакет і клас і як це використовувати при розробці додатків.

AVD (Android Virtual Device) - інтерактивний емулятор мобільного пристрою Android. Використовуючи емулятор, можна запускати і тестувати програми без використання реального Android пристрою.

Development Tools – SDK включає кілька інструментальних засобів для розробки, які дозволяють компілювати і налагоджувати створювані додатки.

Sample Code – Android SDK надає типові додатки, які демонструють деякі з можливостей Android, і прості програми, які показують, як використовувати індивідуальні особливості API у вашому коді.

Крім емулятора, SDK також включає безліч інших інструментальних засобів для налагодження та установки створюваних додатків. Якщо При розробці програми для Android за допомогою IDE Android Studio, багато інструментів командного рядка, що входять до складу SDK, вже використовуються при складанні і компіляції проекту. Android – важливий інструмент розробки, що запускається з командного рядка, який дозволяє створювати, видаляти і

конфігурувати віртуальні пристрої, створювати і оновлювати Android проекти (при роботі поза середовища Eclipse) і оновлювати Android SDK новими платформами, доповненнями і документацією.

Dalvik Debug Monitor Service (DDMS) – інтегрований з Dalvik Virtual Machine, стандартної віртуальної машини платформи Android, цей інструмент дозволяє управляти процесами на емуляторі, а також допомагає в налагодженні додатків.

Можна використовувати цей сервіс для завершення процесів, вибору певного процесу для налагодження, генерування трасувань даних, перегляду "купи" або інформації про потоки, робити скріншоти емулятора та багато іншого [25].

Hierarchy Viewer – візуальний інструмент, який дозволяє налагоджувати і оптимізувати користувальницький інтерфейс розробляється. Він показує візуальне дерево ієрархії уявлень, аналізує швидкодію перемальовування графічних зображень на екрані і може виконувати ще багато інших функцій для аналізу графічного інтерфейсу додатків.

Layoutopt – інструмент командного рядка, який допомагає оптимізувати схеми розмітки та ієрархії розміток в створюваному додатку. Необхідний для вирішення проблем при створенні складних графічних інтерфейсів, які можуть зачіпати продуктивність програми.

Середовище розробки Android Studio зображено на рис. 3.1.

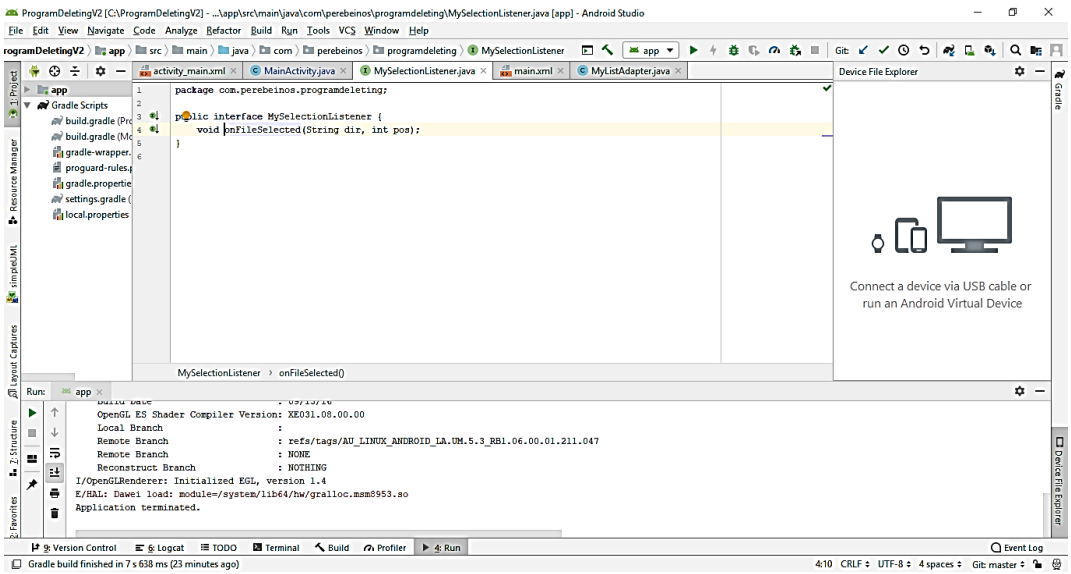


Рисунок 3.1 – Середовище розробки Android Studio

Середовище Android Studio дозволяє здійснювати відлагодження програмного засобу всередині за допомогою емулятора. Така можливість спрощує процес розробки оскільки не потрібно кожен раз створювати apk файл і завантажувати на пристрій.

3.2 Розробка модуля ідентифікування та з'єднання з HID-пристроями

Для початку необхідно ідентифікувати HID пристрій, що підключається, серед всього розмаїття USB девайсів. USB девайси ідентифікуються за поєднанням vid (vendor id) і pid (product id). XML-файл device_filter.xml буде містити відомості про конкретний HID-пристрій, тобто додаток з усіх можливих підключених пристроїв буде відображати тільки потрібний ППКП Tiras PrimeA:

```
<resources>
   <usb-device vendor-id="1155" product-id="22352" />
</resources>
```

Тепер необхідно внести відповідні дозволи та action до маніфесту програми:

```
<uses-permission android:name="android.permission.USB_PERMISSION" />
<uses-feature android:name="android.hardware.usb.host" />

<activity android:name=".MainActivity">
<intent-ilter>
   <action android:name="android.intent.action.MAIN" />
   <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<meta-data
   android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
   android:resource="@xml/device_filter" />
```

В android:resource необхідно вказати файл із необхідними фільтрами для пристроїв. Також можна призначити intent фільтри для запуску програми, наприклад, в результаті підключення пристрою.

Для початку необхідно отримати UsbManager, знайти пристрій, інтерфейс та кінцеві точки пристрою. Це необхідно робити при кожному підключенні пристрою.

```
val usbManager = context.getSystemService(Context.USB_SERVICE) as UsbManager

private var usbConnection: UsbDeviceConnection? = null
```

```
private var usbInterface: UsbInterface? = null
private var usbRequest: UsbRequest? = null
private var usbInEndpoint: UsbEndpoint? = null
private var usbOutEndpoint: UsbEndpoint? = null

fun enumerate(): Boolean {
    val deviceList = usbManager.deviceList
    for (device in deviceList.values) {

        /* Знайти девайс VID и PID */
        if ((device.vendorId == VENDOR_ID) and (device.productId == PRODUCT_ID))
{

            /* Інтерфейс за номером */
            usbInterface = device.getInterface(CUSTOM_HID_INTERFACE)

            /* Підбір кінцевих точок*/
            for (idx in 0 until usbInterface!!.endpointCount) {
                if (usbInterface?.getEndpoint(idx)?.direction == USB_DIR_IN)
                    usbInEndpoint = usbInterface?.getEndpoint(idx)
                else
                    usbOutEndpoint = usbInterface?.getEndpoint(idx)
            }

            usbConnection = usbManager.openDevice(device)
            usbConnection?.claimInterface(usbInterface, true)

            usbRequest = UsbRequest()
            usbRequest?.initialize(usbConnection, usbInEndpoint)
        }
    }

    /* Статус підключення */
    return usbConnection != null
```

У наведеному лістингу можна побачити інтерфейси та кінцеві точки пристрою для організації обміну даними. Знаючи номер інтерфейсу, необхідно знаходимо обидві кінцеві точки, прийом і передачу, і ініціювати usb з'єднання.

HID-пристрої спілкуються через репорти.

```
fun sendReport(data: ByteArray) {
    usbConnection?.bulkTransfer(usbOutEndpoint, data, data.size, 0)
}
```

```
fun getReport(): ByteArray {
    val buffer = ByteBuffer.allocate(REPORT_SIZE)
    val report = ByteArray(buffer.remaining())

    if (usbRequest.queue(buffer, REPORT_SIZE)) {
        usbConnection?.requestWait()

        buffer.rewind()
        buffer.get(report, 0, report.size)
        buffer.clear()
    }

    return report
        }
```

У метод sendReport передається масив байт, в якому нульовим байтом є репорт ID, беремо USB підключення до пристрою і виконуємо передачу. Як параметри в метод BulkTransfer передаємо номер кінцевої точки, дані, їх розмір та тайм передачі. Варто зазначити, що клас UsbDeviceConnection має методи реалізації обміну даними з пристроєм USB — методи bulkTransfer і controlTransfer. Їх використання залежить від типу передачі, який підтримує та чи інша кінцева точка. У даному випадку використовуємо bulkTransfer, хоча HID найчастіше характерне використання кінцевих точок з типом control.

Для отримання даних необхідно знати розмір даних, які можна, як знати заздалегідь, так і отримати з кінцевої точки.

Метод отримання даних USB HID є синхронним і блокуючим і виконувати його необхідно в іншому потоці, крім того, репорти від пристрою можуть приходити постійно, або в будь-який час, тому необхідно реалізувати постійне опитування репорту, щоб не пропустити дані. Реалізувати можна за допомогою RxJava:

```
fun receive() {
    Observable.fromCallable<ByteArray> { getReport() }
            .subscribeOn(Schedulers.io())
            .observeOn(Schedulers.computation())
            .repeat()
                .subscribe({
```

```
                    /* check it[0] (this is report id) and handle data */
                },{
                    /* handle exeption */
                })
        }
```

Отримавши масив байт, потрібно перевірити нульовий байт, оскільки він є report ID і відповідно до нього парсить отримані дані.

Після завершення всіх дій з USB потрібно закрити з'єднання. Необхідно виконати в onDestroy activity або в onCleared в ViewModel.

```
fun close() {
    usbRequest?.close()
    usbConnection?.releaseInterface(usbInterface)
    usbConnection?.close()
      }
```

3.3 Розробка модуля обміну даними за внутрішнім протоколом UDP

Java постачається з вбудованою мережевою підтримкою UDP, яка є частиною пакета java.net. Тому для виконання мережевих операцій через UDP потрібно лише імпортувати класи з пакету java.net: java.net.DatagramSocket і java.net.DatagramPacket.

Спочатку створимо echo-сервер, який надсилає назад будь-які повідомлення, надіслані йому, потім клієнт echo, який просто надсилає будь-яке довільне повідомлення на сервер, і, нарешті, ми перевіримо програму, щоб переконатися, що все працює нормально.

У зв’язку UDP одне повідомлення інкапсулюється в DatagramPacket, який надсилається через DatagramSocket.

```
public class DatagramSocketSend extends Thread {

   private DatagramSocket socket;
   private boolean running;
   private byte[] buf = new byte[1024];

   public DatagramSocketSend() {
      socket = new DatagramSocket(new BufferedInputStream().read(socket, buf[1024],
1024));
```

```
    }

    public void run() {
        running = true;

        while (running) {
            DatagramPacket packet
                    = new DatagramPacket(buf, buf.length);
            socket.receive(packet);

            InetAddress address = packet.getAddress();
            int port = packet.getPort();
            packet = new DatagramPacket(buf, buf.length, address, port);
            String received
                    = new String(packet.getData(), 0, packet.getLength());

            if (received.equals("end")) {
                running = false;
                continue;
            }
            socket.send(packet);
        }
        socket.close();
    }
}
void write(final String string) throws IOException {
    if(this.ds != null && this.address != null) {
        byte[] bytes = string.getBytes();
        //
        //  syslog packets must be less than 1024 bytes
        //
        int bytesLength = bytes.length;
        if (bytesLength >= 1024) {
            bytesLength = 1024;
        }
        DatagramPacket packet = new DatagramPacket(bytes, bytesLength,
                address, port);
        ds.send(packet);
    }

}
```

Створюється глобальний DatagramSocket, який буде використовуватись всюди для надсилання пакетів, масив байтів для обгортання повідомлень і змінну

стану, яка називається Running.

Для простоти сервер розширює Thread, тому є можливість реалізувати все всередині методу run.

Усередині запуску ми створюється цикл while, який просто виконується, доки виконання не буде змінено на false через якусь помилку або повідомлення про завершення від клієнта(приладу).

У верхній частині циклу ми створюємо DatagramPacket для отримання вхідних повідомлень.

Далі викликається метод прийому на сокеті. Цей метод блокується до тих пір, поки не надійде повідомлення, і воно не збереже повідомлення всередині масиву байтів DatagramPacket, переданого йому.

Після отримання повідомлення отримуємо адресу та порт клієнта, оскільки має відправити відповідь назад.

Далі створюється DatagramPacket для відправки повідомлення клієнту.

Для налагодження правильної роботи модуля розроблено клас, який логує взаємодію модуля обміну даними.

```
public class LogClientConnection {

   private static final Logger LOG = LoggerFactory.getLogger(EchoClient.class);

   private Socket clientSocket;
   private PrintWriter out;
   private BufferedReader in;

   public void startConnection(String ip, int port) {
      try {
         clientSocket = new Socket(ip, port);
         out = new PrintWriter(clientSocket.getOutputStream(), true);
         in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      } catch (IOException e) {
         LOG.debug("Error when initializing connection", e);
      }

   }
```

```
public String sendMessage(String msg) {
    try {
        out.println(msg);
        return in.readLine();
    } catch (Exception e) {
        return null;
    }
}

public void stopConnection() {
    try {
        in.close();
        out.close();
        clientSocket.close();

    } catch (IOException e) {
        LOG.debug("error when closing", e);
    }
```

3.4 Розробка головного модуля побудови конфігурації системи

У системі яка буде працювати під управлінням приймально-контрольного пожежного приладу мають бути сповіщувачі, оповіщувачі, модулі розширення.

Найменшою структурною одиницею конфігурації є канал “Channel”. Канал представляє собою один виділений унікальний параметр даних, який контролює приймально-контрольний пристрій. Кожен доданий в конфігурацію пристрій має свою певну кількість каналів, від 1-го до 16-ти для підключених зовнішніх пристроїв, також сам приймально-контрольний пристрій має свої канали.

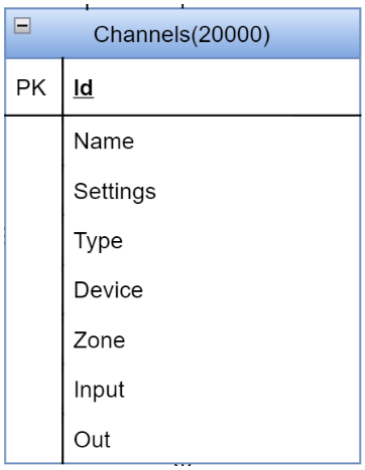


Рисунок 3.2 – Клас опису каналів

Модуль для створення конфігурації містить в собі описи класів відповідно до можливостей системи. В модулі вся структура класів відтворює реальну конфігурацію системи з підлеглими пристроями та модулями.

Приймально-контрольний пожежний пристрій має системні обмеження щодо величини конфігурації. Наприклад, об’єктів типу channel максимально можлива кількість 20000, device – 4000, addresses interfaces – 128, in – 128, out – 250, zones – 1000, groups – 128, users – 17, ecds – 32, m-out8r – 16.

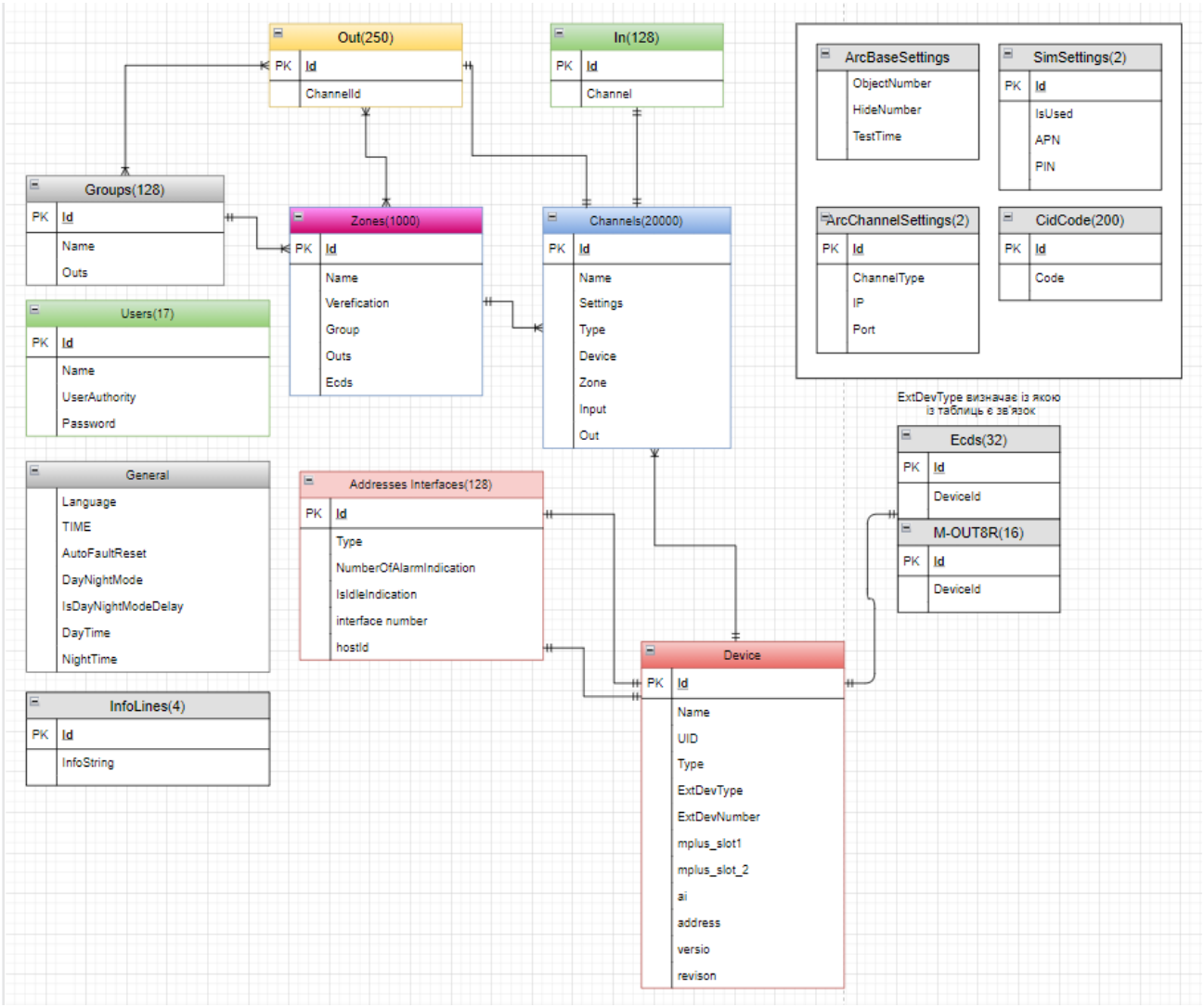


Рисунок 3.3 – Діаграма класів модуля побудови конфігурації

3.5 Розробка модуля локального збереження конфігурації

При розробці цього модуля було реалізовано не тільки функцію локального збереження конфігурації, а й роботу з вже готовим файлом конфігурації, тобто вичитування даних з файлу.

В файлі конфігурації данні зберігаються і форматі XML-таблиці. Найкращий метод для організації сереалізації даних. Модуль MyXMLParser з готового об’єкту конфігурації MainConfig створює буфер потоку даних для запису в файл за типом HashTable(ключ-значення).

```
public final class MyXMLBuilder {
    public static final String DATA_TYPE = "dType";
    public static final String PARAMETER_TYPE = "pType";
    private static Callback sCallback;

    private MyXMLBuilder() {
    }

    public static void convertToXML(MainConfig var0, String var1, Callback var2) {
        sCallback = var2;
        (new ConfigToXMLTask(var1, var0)).execute();
    }

    public static Callback access$000() {
        return null;
    }

    public interface Callback {
        void onBuildXmlPreExecute();

        void onBuildXmlResult(File var1, Exception var2);
    }

    private static class ConfigToXMLTask extends BackgroundWorker<File> {
        private final MainConfig config;
        private Exception exception;
        private final File file;

        public ConfigToXMLTask(String var1, MainConfig var2) {
            this.file = new File(var1);
            this.config = var2;
        }

        public File doInBackground() {
            // $FF: Couldn't be decompiled
            return null;
        }

        public void onPostExecute(File var1) {
            if (MyXMLBuilder.sCallback != null) {
                MyXMLBuilder.sCallback.onBuildXmlResult(var1, this.exception);
            }

        }
```

```
        public void onPreExecute() {
            if (MyXMLBuilder.sCallback != null) {
                MyXMLBuilder.sCallback.onBuildXmlPreExecute();
            }

        }

        public void execute() {
        }
    }
}
```

У вже збереженому файлі структура відображається так:

```
<i id="999">
    <name>ЗОНА 1000</name>
    <verification_type>NONE</verification_type>
    <verification_time>0</verification_time>
    <group_id>128</group_id>
    <out_ids/>
    <ecd_ids>32,32</ecd_ids>
  </i>
 </FxZonesModule>
 <FxDevicesModule>
  <i id="0">
    <name>M-OUT8R s/n:070000002</name>
    <type>AM_CONVERTER</type>
    <uid>000001</uid>
    <ext_device_type>M_OUT8R</ext_device_type>
    <ext_device_id>0</ext_device_id>
    <version>1</version>
    <revision>0</revision>
    <channel_ids>25,26,27,28,29,30,31,32,33</channel_ids>
    <ai_id>3</ai_id>
    <ai_address>1</ai_address>
    <connected_device>4000</connected_device>
  </i>
```

3.6 Висновки

В даному розділі було проведено аналіз та обґрунтування вибору мови програмування для реалізації програмного додатку, розроблено модуль для розпізнавання потрібного HID-пристрою та встановлення потоку з’єднання, описано розробку модулів для побудови конфігурації приймально-контрольного пристрою та обміну даними між Android і HID-пристроями, а також реалізовано модуль для локального збереження налаштованої конфігурації та роботу з готовим файлом конфігурації.

4 ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ КОНФІГУРУВАННЯ ПРИЙМАЛЬНО-КОНТРОЛЬНОГО ПОЖЕЖНОГО ПРИЛАДУ

4.1 Методи тестування програмного додатку

Тестування програмного забезпечення – це процес оцінки функціональних можливостей програмного забезпечення з метою з'ясувати, чи відповідає розроблене програмне забезпечення зазначеним вимогам чи ні, та виявити дефекти, щоб забезпечити безвідмовність продукту для отримання якісного продукту .

Існує стандартне визначення про такі типи тестування: ручне тестування та автоматизоване, методи тестування, підходи до тестування та типи тестування чорних ящиків.

Виявляти дефекти на ранніх етапах життєвого циклу розробки програмного забезпечення дозволяє підвищити ефективність розробки та знизити загальну вартість проекту. Зазвичай тестування роблять при розробці нового функціоналу – функціональне тестування та при релізі нової версії продукту – регресійне тестування.

Функціональне тестування проводять для перевірки якості розробленого функціоналу. На всі виявлені дефекти створюються баг репорти. Коли протестований функціонал затверджується, як якісний, для нього створюються тест кейси для подальшої перевірки при регресійному тестуванні.

Регресійне тестування проводиться командою тестувальників, для перевірки коректності функціонування всього додатку перед виходом нової версії продукту. Для забезпечення повного покриття функціоналу зазвичай використовують тест-кейси. Це послідовний набір кроків для тестування та чітко описаний очікуваний результат. У разі не відповідності між актуальним і очікуваним результатом, тест-кейс вважається проваленим, тестувальники створюють баг-репорти. Недоліком тест-кейсів можна вважати неможливість повного покриття всього функціоналу додатку, тому досвідчені тестувальники зазвичай відтворюють не тільки кроки з тест-кейсів, а й перевіряють суміжний функціонал. Для коректності проведення регресійного тестування зазвичай впроваджують feature freaze.

Black box testing (тестування методом “чорного ящика”) – проведення функціонального тестування без відкритого доступу до коду системи. Ґрунтується на знанні тестувальником тільки вимог і специфікацій до продукту, що розробляється, а також світових стандартів.

White box testing (тестування методом “білого ящика”) є повною протилежністю техніки “чорного ящика”.

Це проведення функціонального тестування з відкритим доступом до коду системи. Ґрунтується на знанні тестувальником програмного коду продукту, розуміння його бізнес-логіки, структури та реалізації. Такий тип тестування вимагає знань програмування і здійснюється в основному саме програмістами, які розуміють як реалізована програма зсередини, розуміють алгоритми роботи програми, читають код і здатні знайти помилки в його роботі. В такому випадку, тестувальник вибирає вхідні значення, ґрунтуючись на знанні коду, який буде їх обробляти. Так саме він знає, яким повинен бути результат цієї обробки[26].

4.2 Тестування пошуку та встановлення з’єднання з приймально-контрольним пожежним пристроєм

Модуль ідентифікування та встановлення з’єднання реалізовано таким чином, що після запуску, додаток знаходиться в постійному пошуку потрібного ідентифікатору пристрою, тобто, vid (vendor id) і pid (product id).

В стартовому пакеті від приймально-контрольного пристрою записано статичні vid та pid:

```
<usb-device vendor-id="2522" product-id="4294965393" />
```

Число 2522 – це 0x9DA в 16-й системі, а число 4294965393 – це 0xFFFFF89.

На даному етапі розробки конфігуратор призначений тільки для одного пристрою, серії Tiras PRIME A, при подальшій розробці, якщо виникне необхідність в додаванні ще декількох девайсів для конфігурування, додати до файлу ще декілька необхідних ідентифікаторів нових HID-пристроїв не буде важко. Та додаток буде працювати належним чином.

```
12535
HidDevice [path=\\?\hid#vid_09da&pid_f891&mi_00#8
 &2dc4a061&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030
 }, vendorId=0x9da, productId=0xfffff891,
 serialNumber=null, releaseNumber=0xffff9900,
 manufacturer=COMPANY, product=USB Device,
 usagePage=0x1, usage=0x2, interfaceNumber=0]
```

Рисунок 4.1 – Ідентифікація HID-пристрою

На рисунку відображено інформацію про виявлений HID-пристрій при дебагу додатку з ПК. В інтерфейсі додатку він відображається виділеною іконкою з назвою пристрою “Tiras PRIME A”, як зображено на рисунку 4.2.

Рисунок 4.2 – відображення підключено пристрою.

Встановлення зв’язку з пристроєм – це по суті початок обміну даними за розробленим протоколом.

Після натискання на іконку додаток надсилає вже відомому HID-пристрою команду для авторизації, тобто отримання доступу до 3-го рівня доступу конфігурування пристрою, пакет містить 0xF0F6.

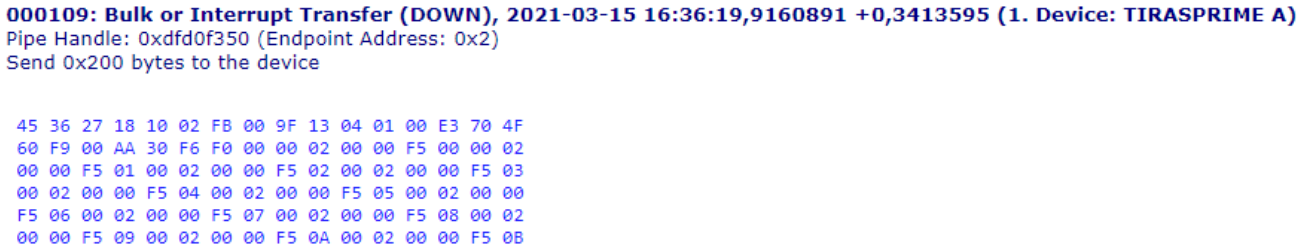


Рисунок 4.3 – Пакет авторизації

Після отримання пакету авторизації приймально-контрольний пристрій відсилає підтвердження на отриманий пакет з кодом 0x30F600 – де 00 це статус успішного виконання команди. Та очікує пакет з кодом доступу до 3-го рівня доступу. Пакет 0xF681 який надсилає додаток має отримати підтвердження 0xF68100.

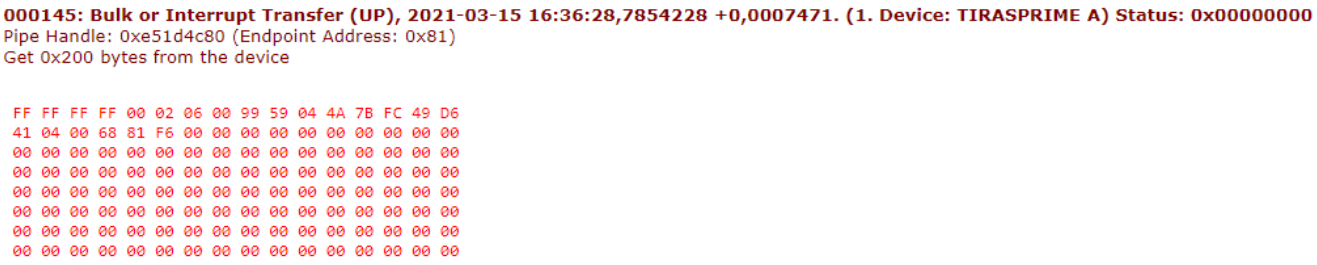


Рисунок 4.4 – Отриманий пакет підтвердження на запит авторизації.

4.3 Тестування обміну даними та побудови пакетів протоколу

Модуль побудови пакетів UDP та контролю обміну даними між додатком(Android) та HID-пристроєм(приймально-контрольний пристрій) важлива частина всього додатку, оскільки якість роботи модуля на пряму відноситься до

правильного конфігурування приймально-контрольного пожежного пристрою.

Для спостерігання та контролювання обміном даними на час тестування додатку ввімкнено додаткове логування у текстовий файл txt. За допомогою логування можна легко та швидко виявити чи правильно формуються та надсилаються UDP пакети зі сторони Android-додатку та зі сторони приймально-контрольного пристрою. Перевірити їх відповідність розробленому протоколу обміну даними.

Надсилання конфігурації на приймально-контрольний пристрій відбувається у два етапи. Перший – формування та надсилання пакетів CRC – інформація про об’єм та складові конфігурації. Та другий – формування та надсилання деталізованих пакетів повної конфігурації, тобто, назви та налаштування каналів, пристроїв, зон, груп та системні налаштування.

```
WARNING: New config transfer
Nov 10, 2021 4:23:41 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl sendCrc$lambda-8
INFO: ---------------CRC start -----------------
Nov 10, 2021 4:23:41 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl sendCrc$lambda-8
INFO: OptionsSet(totalLength=0, id=5, code=-2512, options=[Option(optCode=-16, optNum=0, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-11, optNum=0, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-11, optNum=16, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-10, optNum=0, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-8, optNum=0, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-13, optNum=0, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=0, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=1, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=2, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=3, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=4, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=5, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=6, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=7, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=8, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=9, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=10, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=11, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=12, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=13, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=14, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=15, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=16, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=17, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=18, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=19, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=20, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=21, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=22, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-15, optNum=23, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-15, optNum=24, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-12, optNum=0, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-9, optNum=0, optLength=2, optData=[00 00], fullSize=6),
Option(optCode=-9, optNum=1, optLength=2, optData=[00 00], fullSize=6), Option(optCode=-1, optNum=0, optLength=2, optData=[00 00], fullSize=6)],
tag=crc)
Nov 10, 2021 4:23:41 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl sendCrc$lambda-8
INFO: ---------------CRC end -----------------
```

Рисунок 4.5 – Формування CRC пакетів конфігурації

Після формування блоку CRC додаток починає надсилання сформованих даних на приймально-контрольний пристрій та згідно розробленого модулю додаток очікує підтвердження від пристрою про кожен отриманий пакет. Якщо ж підтвердження на конкретний пакет даних не було отримано, з деяких не відомих причин, то він буде надісланий повторно. Спроб для надсилання одного і того ж пакету є 5, у випадку не отримання підтвердження на всі 5 повторних пакетів додаток перериває встановлену сесію для обміну даними та видаю діалогове вікно про помилку надсилання конфігурації.

```
INFO: Sending frame [[id: 7, code: -2512]]
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl onNewMessage
INFO: New incoming packet: 15 cd 5b 07 00 02 06 00 85 0f 04 67 7b 53 ff 8b 61 04 00 07 30 f6 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 02 00 00 00 06 00 00 00 90 7a 06
00 86 41 1d 00 b0 8b 00 00 00 00 00 00 1f 00 00 60 10 11 21 00 64 b5 22 00 38 b5 22 00 79 00 00
00 00 00 00 00 89 21 0e 00 0a 00 00 00 09 00 00 00 14 00 00 00 11 00 00 00 13 00 00 00 9a 41 1d
00 06 00 00 00 01 00 00 00 07 07 07 07 ac 27 00 00 00 00 00 00 53 ff 8b 61 13 00 00 00 14 00 00
00 11 00 00 00 0a 00 00 00 0a 00 00 00 79 00 00 00 03 00 00 00 39 01 00 00 ff ff ff ff 00 00 00
00 00 00 00 00 01 ff 00 00 13 00 00 00 14 00 00 00 11 00 00 00 03 00 00 00 0a 00 00 00 0b 00 00
00 d6 34 1d 00 0d 00 00 00
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl onNewMessage
INFO: SuperFrame parsed: SuperFrame (serial: 123456789, initiator: DEVICE, encryption: NONE,
protocolRev: 2,dataLength: 6,channelType: OTHER,CRC: 3973,identifier: 31591,time:
1636564819,data: 04 00 07 30 f6 00,uid: null,tag: null)
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl handleFrameData
INFO: Received with id: 7
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl onAckReceived
INFO: Ack received ([4, 0, 7, 48, -10, 0]), parsed: FrameAck(id=7, code=-2512, executeStatus=0)
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl
incrementConfigProgressState
INFO: config transfer progress changed ProgressState[2 of 4 (50.0%)]
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl
deviceEmitter$lambda-0
INFO: Sending frame [[id: 8, code: -2512]]
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl onNewMessage
INFO: New incoming packet: 15 cd 5b 07 00 02 06 00 88 c8 04 68 7b 53 ff 8b 61 04 00 08 30 f6 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa
00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 fa 00 02 00 00 00 14 00 00 00 90 7a 06
00 94 41 1d 00 02 00 00 00 14 00 00 00 01 00 00 00 05 00 00 00 84 89 00 00 38 b5 22 00 79 00 00
00 00 00 00 00 89 21 0e 00 0a 00 00 00 09 00 00 00 14 00 00 00 11 00 00 00 13 00 00 00 9a 41 1d
00 06 00 00 00 01 00 00 00 07 07 07 07 ac 27 00 00 00 00 00 00 53 ff 8b 61 13 00 00 00 14 00 00
00 11 00 00 00 00 00 00 00 1f 00 00 a0 d6 34 1d 00 0d 00 00 00 00 00 00 00 30 00 00 00 d6 34 1d
00 0d 00 00 00 01 00 00 00 07 07 07 07 08 08 08 08 09 09 09 09 10 10 10 10 11 11 11 11 f6 00 00
00 d6 34 1d 00 0d 00 00 00
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl onNewMessage
INFO: SuperFrame parsed: SuperFrame (serial: 123456789, initiator: DEVICE, encryption: NONE,
protocolRev: 2,dataLength: 6,channelType: OTHER,CRC: -14200,identifier: 31592,time:
1636564819,data: 04 00 08 30 f6 00,uid: null,tag: null)
Nov 10, 2021 4:23:49 PM ua.tiras.aloader.core.interaction.DeviceInteractorImpl handleFrameData
INFO: Received with id: 8
```

Рисунок 4.6 – Лог обміну даними за протоколом

Розшифрування лог-файлу:

INFO: Sending frame [[id: 7, code: -2512]] – надсилання пакету з id 7. Сформований пакет можна побачити на етапі формування попереднього блоку CRC.

INFO: New incoming packet: 15 cd 5b 07 00 02 06 00 – розгорнутий пакет підтвердження отримання надісланого раніше пакету.

INFO: SuperFrame parsed: SuperFrame – розібраний пакет про підтвердження.

INFO: Received with id: 7 – видалення пакету на який отримано підтвердження з черги на надсилання.

INFO: config transfer progress changed ProgressState[2 of 4 (50.0%)] – прогрес успішного надсилання пакетів.

Таким чином було перевірено правильність роботи модуля побудови пакетів UDP та контролю обміну даними між додатком(Android) та HID-пристроєм(приймально-контрольний пристрій). Пакети відповідають структурі побудованого проколу передачі, та розроблений модуль працює правильно забезпечуючи цим надійність коректного передавання конфігурації на приймально-контрольний пристрій.

4.4 Тестування створення максимальної конфігурації приладу

З боку звичайного користувача додаток-конфігуратор призначений в першу чергу для створення конфігурації. Отже потрібно реалізувати максимально зручний та інтуїтивно зрозумілий інтерфейс мобільного додатку.

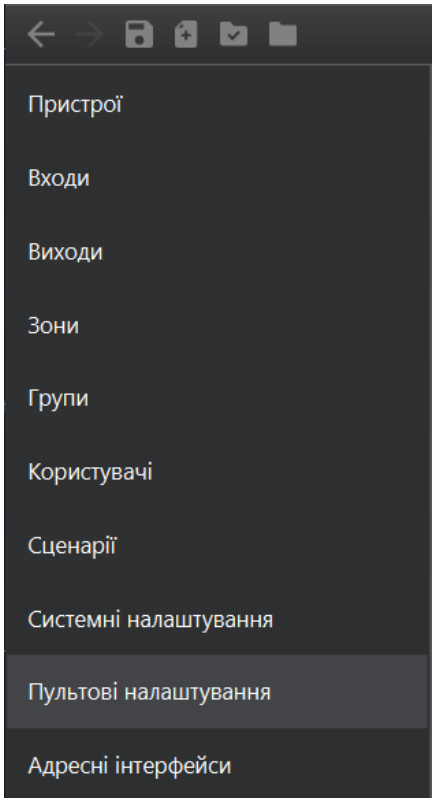


Рисунок 4.7 – Початкове вікно додатку

Отже, початковий екран додатку містить всі необхідні пункти налаштувань, такі як: Пристрої; Входи; Виходи; Зони; Групи; Користувачі; Сценарії; Системні налаштування; Пультові налаштування; Адресні інтерфейси.

Налаштування конфігурації відбувається саме в такому порядку, як розміщені пункти конфігурації.

В кожного із пунктів є свої відкриті меню, де конфігуруються окремо кожна частина загальної конфігурації, наприклад, Пристрої.

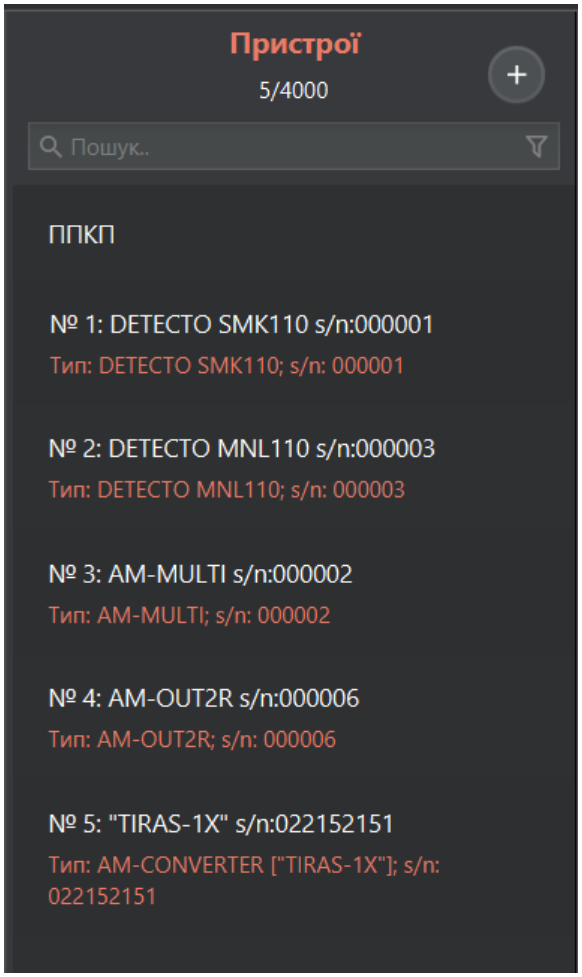


Рисунок 4.8 – Вкладка Пристрої

Інсталятор - людина яка виконує встановлення та налаштування системи пожежної безпеки, додає в систему необхідну кількість потрібних йому пристроїв та модулів, вони всі відображаються послідовним списком. Для зручності реалізований пошук пристроїв за ключовими словами(ідентифікаторами) і також фільтр за типами пристроїв.

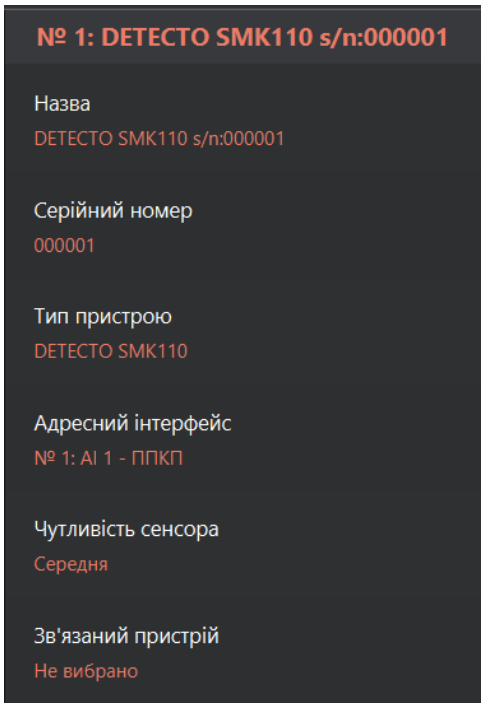


Рисунок 4.9 – Налаштування окремого пристрою

Інтерфейс додатку доволі зрозумілий та однотипний, в пунктах де можна додати новий елемент списку, є кнопка «додати». В пунктах де є можливість тільки змінювати наявні елементи конфігурації потрібні поля відкриті для редагування та після натиснення на них відкривається діалог введення/зміни параметрів.

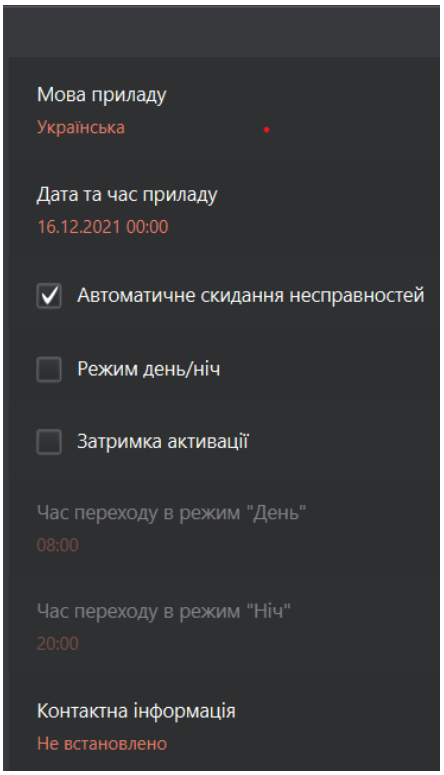


Рисунок 4.10 – Вкладка системних налаштувань

Створення конфігурації звісно важлива частина процесу налаштування приймально-контрольного пожежного пристрою, але ж запис цієї конфігурації переважає за значущістю, отже є необхідність у тестуванні запису вже готової повної(максимально можливої) конфігурації.

Отже, створивши максимальну можливу конфігурацію системи, для початку необхідно перевірити правильність запису її у текстовий файл, так як це є одною із вимог додатку.

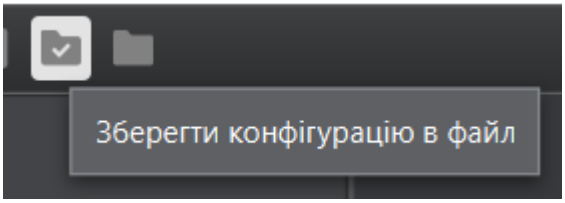


Рисунок 4.11 – Кнопка для збереження конфігурації

Натискаючи на кнопку відкривається файловий менеджер пристрою та потрібно обрати папку та ввести назву файлу, в який буде збережено дану конфігурацію, є можливість перезаписати вже існуючий файл, якщо він формату XML або ж створити новий.

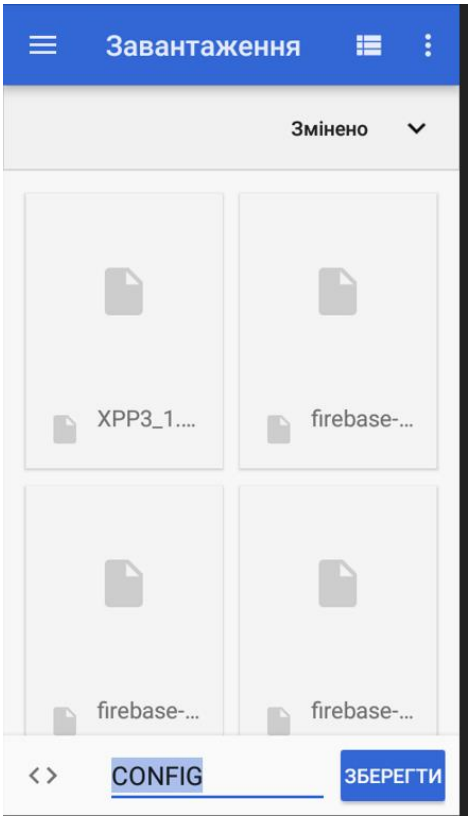


Рисунок 4.12 – Файловий менеджер пристрою

```
<?xml version="1.0" encoding="WINDOWS-1251" standalone="yes"?>
<config hw="1" ver="3">
  <modules>
    <FxUsersModule>
      <i id="0">
        <name>АДМІНІСТРАТОР</name>
        <authority>LEVEL_2</authority>
        <access_code>2</access_code>
      </i>
      <i id="16">
    <FxSystemParamsModule>
      <i id="0">
        <language>UKR</language>
        <faults_auto_reset>true</faults_auto_reset>
        <day_night>false</day_night>
        <day_night_delay_ignored>false</day_night_delay_ignored>
        <day_time>08:00</day_time>
        <night_time>20:00</night_time>
        <ppk_name>TIRAS PRIME A - 115865</ppk_name>
        <ppk_acr_number>0</ppk_acr_number>
        <line_control>false</line_control>
        <main_ppk_mode>false</main_ppk_mode>
        <show_other_events>false</show_other_events>
        <contact_info>
          <row index="0"/>
          <row index="1"/>
          <row index="2"/>
          <row index="3"/>
        </contact_info>
      </i>
    </FxSystemParamsModule>
    <FxMout8RModule>
      <i id="0">
        <device_id>0</device_id>
        <uid>000000000</uid>
      </i>
    <FxNetworkModule>
      <i id="0">
        <enabled>false</enabled>
        <object_number>0</object_number>
        <hidden_number>0</hidden_number>
        <test_period>1</test_period>
      </i>
      <FxArcChannelModule>
        <i id="0">
          <type>NOT_USED</type>
          <ip/>
          <port>0</port>
          <cid_code>0</cid_code>
        </i>
    <FxDevicesModule>
      <i id="0">
        <name>MNL s/n:044788</name>
        <type>MANUAL</type>
        <uid>044788</uid>
        <ext_device_type>NONE</ext_device_type>
        <ext_device_id>255</ext_device_id>
        <version>2</version>
        <revision>1</revision>
        <channel_ids>25</channel_ids>
        <ai_id>0</ai_id>
        <ai_address>1</ai_address>
        <connected_device>4000</connected_device>
      </i>
    <FxChannelModule>
      <i id="0">
        <name>ВИХІД 'ТРИВОГА'</name>
        <channel_type>OUTPUT</channel_type>
        <input_type>GENERAL</input_type>
        <thermal_class>A1</thermal_class>
        <input_work_mode>UNUSED</input_work_mode>
        <input_activation_level>N_O</input_activation_level>
        <out_type>NOTIF</out_type>
        <out_work_mode>UNCONTROLLED</out_work_mode>
        <out_auto_activation>false</out_auto_activation>
        <owner_device_id>4000</owner_device_id>
        <zone_id>1000</zone_id>
        <out_id>0</out_id>
        <input_id>128</input_id>
        <ext_device_channel_type>INPUT_ZONE_1</ext_device_channel_type>
        <smoke_sensitivity>MEDIUM</smoke_sensitivity>
        <out_delay>0</out_delay>
      </i>
      <i id="30">
        <name>ВИХІД [OD2]</name>
        <channel_type>OUTPUT</channel_type>
        <input_type>GENERAL</input_type>
        <thermal_class>A1</thermal_class>
        <input_work_mode>UNUSED</input_work_mode>
        <input_activation_level>N_O</input_activation_level>
        <out_type>GENERAL</out_type>
        <out_work_mode>BLOCKED</out_work_mode>
        <out_auto_activation>false</out_auto_activation>
        <owner_device_id>2</owner_device_id>
        <zone_id>1000</zone_id>
        <out_id>250</out_id>
        <input_id>128</input_id>
        <ext_device_channel_type>INPUT_ZONE_1</ext_device_channel_type>
        <smoke_sensitivity>MEDIUM</smoke_sensitivity>
        <out_delay>0</out_delay>
      </i>
```

Рисунок 4.13 – Основні елементи збереженої конфігурації

На рисунку 4.13 наведено приклади декількох елементів у файлі збереженої конфігурації. Параметри усіх елементів структуровано та описано правильно. На рисунку 4.14 наведено детальний опис одного із головних внутрішніх каналів пристрою, тип каналу «Вихід» та він має усі належні йому параметри з опціями.

```
<i id="8">
  <name>ДАТЧИК ТЕМПЕРАТУРИ</name>
  <channel_type>INTERNAL</channel_type>
  <input_type>GENERAL</input_type>
  <thermal_class>A1</thermal_class>
  <input_work_mode>PARAMETRIC_ZONE</input_work_mode>
  <input_activation_level>N_C</input_activation_level>
  <out_type>GENERAL</out_type>
  <out_work_mode>UNUSED</out_work_mode>
  <out_auto_activation>false</out_auto_activation>
  <owner_device_id>4000</owner_device_id>
  <zone_id>1000</zone_id>
  <out_id>250</out_id>
  <input_id>128</input_id>
  <ext_device_channel_type>INPUT_ZONE_1</ext_device_channel_type>
  <smoke_sensitivity>MEDIUM</smoke_sensitivity>
  <out_delay>0</out_delay>
</i>
```

Рисунок 4.14 – Опис каналу у файлі конфігурації

4.5 Висновки

У цьому розділі було розглянуто та обрано методи та засоби для тестування розробленого мобільного додатку для конфігурування приймально-контрольного пожежного приладу. Протестовано роботу основних та головних модулів додатку, а саме: модуль пошуку та встановлення зв’язку з HID-пристроями; модуль обміну та побудови пакетів протоколу; модуль створення загальної конфігурації системи.

Детально перевірено формування та передавання пакетів UPD за розробленим алгоритмом передавання по інтерфейсу USD-HID.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Результатом магістерської кваліфікаційної роботи «Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу» є мобільний Android-додаток для конфігурування пожежного приймально-контрольного приладу.

Для проведення технологічного аудиту залучено трьох незалежних експертів: к.т.н., доц. каф. КН Арсенюк І.Р. , к.т.н. доц. кафедри ПЗ Катєльніков Д.І., к.т.н. доц. кафедри ПЗ Черноволик Г.О.

Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 5.1.

Таблиця 5.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Крите-р.	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер.	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів	Необхідно отримання великої кількості дозвільних документів, що вимагає значних коштів та часу	Процедура отримання дозвільних документів у вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.2.

Таблиця 5.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Іванчук Я.В. проф. кафедри КН	Катєльніков Д.І. доц. кафедри ПЗ	Черноволик Г.О. доц. кафедри ПЗ
	Бали, виставлені експертами:		
1	3	3	4
Ринкові переваги (недоліки):			
2	3	3	2
3	3	4	3
4	3	3	4
5	3	4	3
Ринкові перспективи			
6	3	2	4
7	3	3	3
Практична здійсненність			
8	4	3	4
9	3	4	3
10	4	4	4
11	4	4	3
12	4	3	3
Сума балів	$СБ_1$=40	$СБ_2$=40	$СБ_3$=40
Середньоарифметична сума балів $\overline{СБ}$	40		

За даними таблиці 5.3 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 40 балів, що відповідає рівню «вище середнього».

В першому розділі роботи були розглянуті переваги та недоліки таких аналогів, як: tLoader, tLoader Prime, oLoaderII, aLoader.

В якості аналога для розробки було обрано aLoader.

Основними недоліками аналога є: версія програми доступна тільки для ПК з ОС Windows або MacOS. Також до недоліків можна віднести те, що в аналозі не реалізований сканер штрих- та QR-кодів.

У таблиці 5.4 наведені основні технічні показники аналога і нового програмного продукту.

Виходячи з результатів отриманих в таблиці 5.4 слід зауважити те, що розроблений програмний продукт випереджає аналог майже по всіх критеріях, що підтверджує його конкурентоспроможність на ринку програмного забезпечення.

Розробка є модифікацією існуючих розробок даної категорії з якісно новим функціональним наповненням. Розробка технічно готова на 100%. Проведене тестування програми довело повну працездатність даного програмного продукту. Розроблено інструкцію користувача.

Таблиця 5.4 - Основні технічні показники аналога і нового програмного продукту

Параметри	Аналог, %	Розробка, %	Примітка
Функціональність	60	85	Переваги у розробки
Мобільність	50	100	Переваги у розробки
Сумісність	70	100	Переваги у розробки
Зручність використання	80	100	Переваги у розробки
Дружність інтерфейсу	80	100	Переваги у розробки

Запропонована програмна реалізація будуть корисним для компаній, особливо за умов, що їх діяльність пов’язана галуззю пожежної безпеки.

На ринку праці наявні фахівці відповідної кваліфікації для обслуговування та підтримки програмного продукту, регламентні обмеження відсутні і немає необхідності отримання дозвільних документів.

Комерціалізація розробки – інвестором проекту є компанія яка виготовляє прилади пожежної безпеки, розроблений додаток є суміжним програмним продуктом із окремими приймально-контрольними пожежними приладами.

Дохід від продукту може бути реалізований у залученні рекламодавців у розміщенні таргетованої реклами.

5.2 Прогнозування витрат на виконання науково-дослідної та конструкторсько–технологічної роботи

Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи може складатися з таких етапів:

1. Розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи.
2. Розрахунок загальних витрат на виконання даної роботи.
3. Прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

4. Розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи.

Виконаємо розрахунок витрат, які безпосередньо стосуються виконавця даного розділу роботи, приймаючи до уваги те, що для розробки програми було залучено одного розробника.

Основна заробітна плата розробника (дослідника) $З_о$:

$$З_о = \frac{М}{Т_р} \cdot t, [грн], \quad (5.1)$$

де М - місячний посадовий оклад конкретного розробника, М = 17000 грн;

$Т_р$ – кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 63 днів.

$$З_о = \frac{17000,00}{21} \cdot 63 = 51000,00 \text{ (грн)}.$$

Результати розрахунків зведемо до таблиці 5.5.

Таблиця 5.5 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Розробник	17000,00	809,50	63	51000,00
Керівник проекту	19000,00	904,70	3	2714,30
Всього				53714,30

Додаткова заробітна плата розраховується як 12 % від суми основної заробітної плати за формулою:

$$З_д = 0,12 \cdot З_о \text{ [грн]}. \quad (5.2)$$

$$З_д = 0,12 \cdot 53714,30 = 6445,70(грн).$$

Нарахування (ЄСВ) на заробітну плату становлять 22%:

$$H_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \text{ [грн]}, \quad (5.3)$$

де Зо – основна заробітна плата розробників, грн;

Зд – додаткова заробітна плата всіх розробників, грн;

β – ставка єдиного соціального внеску на загальнообов'язкове державне соціальне страхування, %.

$$Н_{зп} = (53714,30+6445,70) \cdot 0,22 = 13235,20 \text{ (грн)}.$$

Амортизація обладнання та приміщення, яке використовувалось для проведення розробки, розраховується за формулою:

$$A = \frac{Ц \cdot H_a}{100} \cdot \frac{T}{12} \text{ [грн]}, \quad (5.4)$$

де Ц – балансова вартість обладнання, грн.;

$Н_а$ – річна норма амортизаційних відрахувань;

Т – термін використання під час розробки, місяців.

Норма амортизації розраховується за формулою:

$$H_a = \frac{B_п - B_л}{B_п \cdot T_{кв}} \cdot 100 \text{ [грн]}, \quad (5.5)$$

де $В_п$ і $В_л$ – відповідно первісна та ліквідаційна вартість основних фондів;

$Т_{кв}$ – строк корисного використання, роки.

Норма амортизаційних витрат становитиме:

$$H_a = \frac{20000 - 2000}{20000 \cdot 5} \cdot 100 = 18\%$$

Розрахуємо амортизаційні витрати на ноутбук, балансова вартість якого становить 30000 грн, а термін використання – 3 місяці:

$$\text{А} = \frac{30000 \cdot 18}{100} \cdot \frac{3}{12} = 1350 \text{ (грн).}$$

Зроблені розрахунки наведено у таблиці 4.5.

Таблиця 4.5 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, р	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн
Ноутбук	30000,00	5	3	1350,00
Офісне приміщення	380000,00	20	3	5700,00
Всього:				7050,00

Інформацію про витрати на матеріали, що використані при розробці внесено до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали, що були використані для розробки продукту.

Найменування матеріалу	Одиниці виміру	Ціна за одиницю, грн	Витрачено, шт	Вартість витрачених матеріалів, грн
Папір	уп.	100	1	100,00
Ручка	шт.	15	2	30,00
Олівець	шт.	10	1	10,00
Загальна сума витрат за статтею				140,00

Розрахунок витрат на силову електроенергію ($В_е$) здійснюється за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, [грн], \quad (5.6)$$

де В – вартість 1 кВт-год. електроенергії, становить 1,68 грн./кВт.

П – установлена потужність комп'ютера, кВт;

Ф – фактична кількість годин роботи обладнання.

$К_п$ – коефіцієнт використання потужності, становить 0,3.

$$В_е = 1{,}68 \cdot 0{,}5 \cdot 504 \cdot 0{,}3 = 127{,}1 \text{ (грн)}.$$

Інші витрати $В_{ін}$ охоплюють: витрати на управління організацією, Інтернет, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, Інтернет послуги. Інші витрати $І_в$ можна прийняти як 150% від суми основної заробітної плати розробника:

$$В_{ін} = 150\% \cdot (З_р) \text{ [грн]}. \quad (5.7)$$

$$В_{ін} = 1{,}5 \cdot 53714{,}30 = 80571{,}45 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи – В:

В=53714,30+6445,70+13235,20+7050,00+140+127,1+80571,45=161283,75(грн).

Розраховуємо загальні витрати на виконання даної роботи $В_{заг}$ за формулою:

$$B_{заг} = \frac{B}{\alpha} \text{ [грн]}, \quad (5.8)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях (α= 1.)

$$В_{заг.} = \frac{161283.75}{1} = 161283.75 \text{ (грн)}.$$

Визначаємо загальні витрати на виконання та впровадження результатів виконаної наукової роботи (ЗВ) за формулою:

$$ЗВ = \frac{B_{заг}}{\beta} \text{ [грн]}, \quad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Так, якщо розробка знаходиться:

- на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;
- на стадії технічного проектування, то $\beta \approx 0{,}2$;
- на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;
- на стадії розробки технологій, то $\beta \approx 0{,}4$;
- на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;
- на стадії розробки промислового зразка, $\beta \approx 0{,}7$;
- на стадії впровадження, то $\beta \approx 0{,}9$.

Отже, підставимо дані в формулу й отримаємо результат:

$$ЗВ = \frac{161283.75}{0,7} = 230404,30(грн).$$

Витрати на виконання наукової роботи та впровадження її результатів становитиме 230404,30 грн.

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства.

Виконання даної наукової роботи та впровадження її результатів складає приблизно 1 рік. Позитивні результати від впровадження розробки очікуються вже в перші місяці після впровадження.

Проведемо детальніше прогнозування позитивних результатів та кількісне їх оцінювання по роках.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \cdot \Delta N)_i \ [грн], \quad (5.10)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Припустимо, що внаслідок впровадження результатів наукової розробки чистий прибуток підприємства збільшиться на 60,00 грн, а кількість одиниць реалізованих послуг збільшиться: протягом першого року – на 1000 од., протягом другого року – ще на 2000 од., протягом третього року – ще на 3000 од.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 1 шт., а прибуток, що його отримувало підприємство на одиницю послуги до впровадження результатів наукової розробки – 50,00 грн.

Потрібно спрогнозувати збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_1$ протягом першого року складе:

$$\Delta П_1=60\cdot 1+(60+50)\cdot 1500=165060{,}00 \text{ (грн)}.$$

Обчислимо збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року:

$$\Delta П_2=60\cdot 1+(60+50)\cdot (1500+2000)=385060{,}00 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_3=60\cdot 1+(60+50)\cdot (1500+2000+3000)=715060{,}00 \text{ (грн)}.$$

Розрахунки показують, що комерційний ефект від впровадження розробки виражається у щорічному збільшенні чистого прибутку підприємства.

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Розрахований комерційний ефект від можливого впровадження розробок ще не означає, що ця розробка реально буде впроваджена. Якщо збільшення прогнозованого прибутку від впровадження результатів наукової розробки є вигідним для підприємства (організації), то це ще не означає, що інвестор погодиться фінансувати дану розробку.

Інвестор погодиться вкладати кошти у реалізацію даної наукової розробки тільки за певних умов.

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності. Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

Розрахунок ефективності вкладених інвестицій передбачає:

1-й крок. Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, тобто ЗВ = PV = 230404,30 (грн).

2-й крок. Розраховуємо очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження.

3-й крок. Будуємо вісь часу, на якій відображаємо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів (рис. 5.1).

Платежі показуємо у ті терміни, коли вони здійснюються.

Результати вкладених у наукову розробку інвестицій почнуть виявлятись протягом трьох років.

У першому році підприємство отримає збільшення чистого прибутку на 165060,00 грн відносно базового року, у другому році – збільшення чистого

прибутку на 385060,00 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 715060,00 грн (відносно базового року).

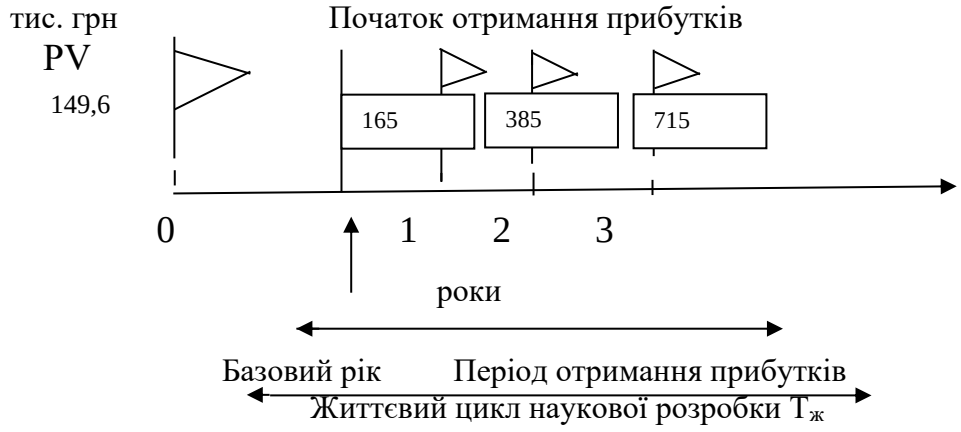


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розраховуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$. за формулою:

$$Е_{абс} = (ПП - PV), [грн], \quad (5.11)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, [\text{ грн}], \quad (5.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 0.

Розрахуємо суму чистого прибутку:

$$\text{ПП} = \frac{165060}{(1+0{,}1)^1} + \frac{385060}{(1+0{,}1)^2} + \frac{715060}{(1+0{,}1)^3} = 1005521{,}10(\text{грн}).$$

Розраховуємо абсолютної ефективності вкладених інвестицій $E_{\text{абс}}$:

$$E_{\text{абс}} = 1005521{,}10 - 230404{,}30 \ = 775116{,}8(\text{грн.})$$

Оскільки $E_{\text{абс}} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, а вкладання коштів у даний проект є доцільним.

Розраховуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1 \qquad (5.13)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1+\frac{775116{,}8}{230404{,}30}} - 1 = \sqrt[3]{4{,}36} - 1 = (1{,}63 - 1) = 0{,}63 \text{ або } 63\%$$

Порівнюємо відносну ефективність $E_в$ з мінімальною (бар'єрною) ставкою

дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,2;

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1), але може бути і значно більше, у нашому випадку f = 0,05.

$$\tau = 0{,}2 + 0{,}05 = 0{,}25$$

Оскільки $E_в$ = 63% > $\tau_{мін}$ = 0,25 = 20%, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку, адже він отримає значно більші прибутки, ніж якщо просто покладе свої гроші на депозит у комерційному банку.

Визначаємо термін окупності вкладених інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} [\text{року}]. \tag{5.15}$$

$$\text{Т}_{\text{ок}} = \frac{1}{0{,}63} = 1{,}58 \text{ (року)}.$$

Термін окупності інвестицій $T_{ок}$ =1,58 < 3…5 років і свідчить, що фінансування даної наукової розробки є доцільним.

5.5 Висновок

В даному розділі було виконано оцінювання комерційного потенціалу розробки. Проведено технологічний аудит з залученням трьох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки є вище середнього. Аналіз комерційного потенціалу розробки показав, що нова розробка за своїми характеристиками випереджає аналоги, що підтверджує її

перспективність. Вона має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку складають 230404,30 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 775116,8 грн свідчить про отримання прибутку інвестором від комерціалізації розробки.

Відносна ефективність вкладених в наукову розробку інвестицій складає 63%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 25%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки. Термін окупності вкладених у реалізацію проекту інвестицій становить 1,58 року, що також свідчить про доцільність фінансування нової розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лесик О.В. Збірник матеріалів міжнародної науково-практичної Інтернет-конференції 9-10 листопада 2021 р: Суми/Віннця НІКО/ВНТУ 2021, 98-99 с.

2. Human Interface Devices (HID) Information. [Електронний ресурс] URL: https://www.usb.org/hid

3. Мобильный доступ HID Mobile Access. Смартфон как идентификатор [Електронний ресурс] URL: https://idcards.ru/materials/technology/skud/mobile-access.php

4. Драйвер прерываний кнопок HID. [Електронний ресурс] URL: https://forum.ixbt.com/topic.cgi?id=4:139638

5. tLoader. [Електронний ресурс] URL: https://tiras.ua/tloader

6. tLoader PRIME. [Електронний ресурс] URL: https://tiras.ua/tloaderprime

7. oLoader II. [Електронний ресурс] URL: https://tiras.ua/oloader-ll

8. aLoader. [Електронний ресурс] URL: https://tiras.ua/aloader

9. Подключаем USB-устройства. [Електронний ресурс] URL: http://developer.alexanderklimov.ru/android/usbmanager.php

10. What is User Datagram Protocol (UDP/IP)?. [Електронний ресурс] URL: https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/

11. Протокол UDP. [Електронний ресурс] URL: http://book.itep.ru/4/44/udp_442.htm

12. Програмне забезпечення та його класифікація. [Електронний ресурс] URL: https://kppk.com.ua/ELLIB/ebook/Gorbenko/IKT/3/3.htm

13. STM32 — USB Custom HID. [Електронний ресурс] URL: https://adelectronics.ru/2020/10/21/stm32-usb-custom-hid/

14. STM32CubeMx. USB Custom HID. Прием и передача данных. [Електронний ресурс] URL: https://microtechnics.ru/stm32cube-usb-custom-hid-priem-i-peredacha-dannyx/

15. HID Application Programming Interface (API). [Електронний ресурс] URL:

https://docs.microsoft.com/en-us/windows-hardware/drivers/hid/introduction-to-hid-concepts

16. USB Transactions. [Електронний ресурс] URL: https://microchipdeveloper.com/usb:transactions

17. USB Human Interface Devices. [Електронний ресурс] URL: https://wiki.osdev.org/USB_Human_Interface_Devices

18. Reliable UDP (RUDP): The Next Big Streaming Protocol?. [Електронний ресурс] URL: https://www.streamingmediaglobal.com/Articles/ReadArticle.aspx?ArticleID=86388

19. UDP: User Datagram Protocol. [Електронний ресурс] URL: https://www.samlit.net/lessons/informatika/tcp_ip/tcp11.html

20. User Datagram Protocol. [Електронний ресурс] URL: https://www.sciencedirect.com/topics/computer-science/user-datagram-protocol

21. 21. C++ vs Java vs Python a comparison. [Електронний ресурс] URL: https://www.javatips.net/blog/c-vs-java-vs-python-a-comparison

22. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. - 689 с.

23. Python IDE. [Електронний ресурс] URL: https://tproger.ru/translations/python-ide/

24. IntelliJ IDEA overview. [Електронний ресурс] URL: https://www.jetbrains.com/help/idea/discover-intellij-idea.html

25. SDK Platform. [Електронний ресурс] URL: https://developer.android.com/studio/releases/platforms

26. Тестирование. Фундаментальная теория. [Електронний ресурс] URL: https://dou.ua/forums/topic/13389/

ДОДАТКИ

Додаток А: Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О.Н.

«01» жовтня 2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу

«Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу»
за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

к.т.н., доцент кафедри ПЗ, Катєльніков Д.І.

«_____»_____2021 р.

Виконав:

студент гр. 2ПІ-20м, Лесик О.В,

«_____»_____2021 р.

Вінниця ВНТУ 2021

1.1 Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу».

Галузь застосування – системи пожежної безпеки.

1.2 Підстава для проведення робіт

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 214 ректора по ВНТУ про закріплення тем МКР.

1.3 Мета та призначення роботи

Метою роботи є розробка програмного додатку для операційної системи Android, який дозволяє конфігурувати приймально-контрольний пожежний прилад через USB HID.

Основними задачами дослідження є:

- провести аналіз методів керування периферійними пристроями за протоколом USB Custom HID з Android пристроїв;
- розробити бітовий протокол обміну даними з приймально-контрольним пожежним приладом.
- розробити програмні засоби та систему конвертації інформації на основі запропонованих методів;
- провести тестування розроблених програмних засобів керування периферійними пристроями.

1.4 Технічні вимоги:

- операційна система Android 5.0 (Lollipop) та вище;
- обсяг оперативного запам'ятовуючого пристрою 2 Гб і більше;

1.5 Перелік технічної документації, що пред’являється по закінченню робіт:

- технічне завдання та обґрунтування;
- лістинги програми.

1.6 Стадії і етапи розробки

Завдання на проектування видане 28 вересня 2020 року. Проектування та дослідження повинно бути завершеним до 1 грудня 2020 року.

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	28.10.2020 р. - 01.10.2020 р.	Вик.
2	Розробка загальної моделі та структури системи	02.10.2020 р. – 10.10.2020 р.	Вик.
3	Підбір та інтеграція програмних засобів для реалізації розроблених модулі системи	11.10.2020 р. - 20.10.2020 р.	Вик.
4	Розробка модулю взаємодії між Android-HID	21.10.2020 р. - 5.11.2020 р.	Вик.
5	Розробка модулів побудови та збереження конфігурації системи	6.11.2020 р. - 15.11.2020 р.	Вик.
6	Тестування роботи модулів та інтерфейсу додатку для конфігурування	16.11.2020 р. - 20.11.20202 р.	Вик.
7	Економічна частина	21.11.2020 р. - 28.11.2020 р.	Вик.
8	Оформлення матеріалів до захисту МКР	29.11.2020 р. - 01.12.2020 р.	Вик.

1.7 Порядок контролю і приймання

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Завдання отримав О.В. Лесик, студент гр. 2ПІ-20м

Науковий керівник Д. І. Катєльніков, к.т.н., доц. кафедри ПЗ

Додаток Б. Протокол перевірки

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу.

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 20м

Науковий керівник: к.т.н. доц. Катєльніков Д.І.

Unicheck	
Оригінальність	**90%**
Схожість	10%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методу та програмного забезпечення мобільного додатку для конфігурування приймально-контрольного пожежного приладу».

Автор _____ Лесик Олександр Валентинович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

Додаток В. Лістинг модуля з’єднання

Лістинг модуля ідентифікації HID-пристроїв

```
import org.hid4java.*;
import org.hid4java.event.HidServicesEvent;

import java.util.concurrent.TimeUnit;

import static java.util.concurrent.TimeUnit.NANOSECONDS;

public class UsbHidDeviceExample implements HidServicesListener {

    private static final Integer VENDOR_ID = 0x3eb;
    private static final Integer PRODUCT_ID = 0xffffaa55;
    private static final int PACKET_LENGTH = 64;
    public static final String SERIAL_NUMBER = null;

    public static void main(String[] args) throws HidException {

        UsbHidDeviceExample example = new UsbHidDeviceExample();
        example.executeExample();

    }

    public void executeExample() throws HidException {

        // Configure to use custom specification
        HidServicesSpecification hidServicesSpecification = new HidServicesSpecification();
        hidServicesSpecification.setAutoShutdown(true);
        hidServicesSpecification.setScanInterval(500);
        hidServicesSpecification.setPauseInterval(5000);

hidServicesSpecification.setScanMode(ScanMode.SCAN_AT_FIXED_INTERVAL_WITH_PAUSE_AFTER_WRITE
);

        // Get HID services using custom specification
        HidServices hidServices = HidManager.getHidServices(hidServicesSpecification);
        hidServices.addHidServicesListener(this);

        // Start the services
        System.out.println("Starting HID services.");
        hidServices.start();

        System.out.println("Enumerating attached devices...");

        // Provide a list of attached devices
        for (HidDevice hidDevice : hidServices.getAttachedHidDevices()) {
            System.out.println(hidDevice);
            System.out.println(hidDevice.getProductId());
        }

        // Open the device device by Vendor ID and Product ID with wildcard serial number
        HidDevice hidDevice = hidServices.getHidDevice(VENDOR_ID, PRODUCT_ID,
SERIAL_NUMBER);
        if (hidDevice != null) {
            // Consider overriding dropReportIdZero on Windows
            // if you see "The parameter is incorrect"
            // HidApi.dropReportIdZero = true;

            // Device is already attached and successfully opened so send message
            //sendMessage(hidDevice);
            byte[] arr = new byte[256];
            int read = hidDevice.read(arr);
            System.out.println(arr);
        }

        System.out.printf("Waiting 30s to demonstrate attach/detach handling. Watch for
slow response after write if configured.%n");
```

```
        // Stop the main thread to demonstrate attach and detach events
        sleepUninterruptibly(30, TimeUnit.SECONDS);

        // Shut down and rely on auto-shutdown hook to clear HidApi resources
        hidServices.shutdown();

    }

    public void hidDeviceAttached(HidServicesEvent event) {

        System.out.println("Device attached: " + event);

        // Add serial number when more than one device with the same
        // vendor ID and product ID will be present at the same time
        if (event.getHidDevice().isVidPidSerial(VENDOR_ID, PRODUCT_ID, null)) {
            sendMessage(event.getHidDevice());
        }

    }

    public void hidDeviceDetached(HidServicesEvent event) {

        System.err.println("Device detached: " + event);

    }

    public void hidFailure(HidServicesEvent event) {

        System.err.println("HID failure: " + event);

    }

    private void sendMessage(HidDevice hidDevice) {

        // Ensure device is open after an attach/detach event
        if (!hidDevice.isOpen()) {
            hidDevice.open();
        }

        // Send the Initialise message
        byte[] message = new byte[PACKET_LENGTH];
        message[0] = 0x3f; // USB: Payload 63 bytes
        message[1] = 0x23; // Device: '#'
        message[2] = 0x23; // Device: '#'
        message[3] = 0x00; // INITIALISE

        int val = hidDevice.write(message, PACKET_LENGTH, (byte) 0x00);
        if (val >= 0) {
            System.out.println("> [" + val + "]");
        } else {
            System.err.println(hidDevice.getLastErrorMessage());
        }

        // Prepare to read a single data packet
        boolean moreData = true;
        while (moreData) {
            byte data[] = new byte[PACKET_LENGTH];
            // This method will now block for 500ms or until data is read
            val = hidDevice.read(data, 500);
            switch (val) {
                case -1:
                    System.err.println(hidDevice.getLastErrorMessage());
                    break;
                case 0:
                    moreData = false;
                    break;
                default:
                    System.out.print("< [");
                    for (byte b : data) {
                        System.out.printf(" %02x", b);
```

```
                    }
                    System.out.println("]");
                    break;
                }
            }
        }

        public static void sleepUninterruptibly(long sleepFor, TimeUnit unit) {
            boolean interrupted = false;
            try {
                long remainingNanos = unit.toNanos(sleepFor);
                long end = System.nanoTime() + remainingNanos;
                while (true) {
                    try {
                        // TimeUnit.sleep() treats negative timeouts just like zero.
                        NANOSECONDS.sleep(remainingNanos);
                        return;
                    } catch (InterruptedException e) {
                        interrupted = true;
                        remainingNanos = end - System.nanoTime();
                    }
                }
            } finally {
                if (interrupted) {
                    Thread.currentThread().interrupt();
                }
            }
        }

    }


    public class OtgDeviceFacade {
        public static int TRANSFER_TIMEOUT;
        private static final int USB_CSW_LENGTH = 13;
        private static final int USB_CSW_OFF_SIGNATURE = 0;
        private static final int USB_CSW_OFF_STATUS = 12;
        private static final int USB_CSW_OFF_TAG = 4;
        private static final int USB_CSW_SIGNATURE = 1396855637;
        private static final int USB_CSW_STATUS_SUCCESS = 0;
        public static final byte USB_DIRECTION_TO_DEVICE = 0;
        public static final byte USB_DIRECTION_TO_HOST = -128;
        private CommandBlockWrapper commandBlockWrapper;
        private CommandStatusWrapper commandStatusWrapper;
        private UsbEndpoint inputEndpoint;
        private UsbEndpoint outputEndpoint;
        private byte[] receiveBuffer;
        private RequestSense requestSense;
        private ScsiRead scsiRead;
        private ScsiReadCapacity scsiReadCapacity;
        private ScsiWrite scsiWrite;
        private final UsbDeviceConnection usbDeviceConnection;
        private UsbInterface usbInterface;

        public OtgDeviceFacade(UsbInterface var1, byte var2, UsbDeviceConnection var3,
    UsbDevice var4) {
            this.commandBlockWrapper = new CommandBlockWrapper(var2, 0);
            this.commandStatusWrapper = new CommandStatusWrapper();
            this.requestSense = new RequestSense();
            this.scsiReadCapacity = new ScsiReadCapacity();
            this.scsiRead = new ScsiRead();
            this.scsiWrite = new ScsiWrite();
            this.receiveBuffer = new byte[8192];
            this.usbDeviceConnection = var3;
            this.usbInterface = var1;
            this.initEnpoints(var1);
            this.initCommandBuffers();
        }

        private void initCommandBuffers() {
```

```
        this.commandBlockWrapper.initCommandBuffers();
        this.commandStatusWrapper.initCommandBuffers();
        this.requestSense.initCommandBuffers();
        this.scsiReadCapacity.initCommandBuffers();
        this.scsiRead.initCommandBuffers();
        this.scsiWrite.initCommandBuffers();
    }

    private void initEnpoints(UsbInterface var1) {
        var1.getInterfaceSubclass();
        UsbEndpoint var2 = var1.getEndpoint(0);
        UsbEndpoint var4 = var1.getEndpoint(1);
        UsbEndpoint var3;
        if (var2.getDirection() == 128) {
            var3 = var2;
            var2 = var4;
        } else {
            var3 = var4;
        }

        this.inputEndpoint = var3;
        this.outputEndpoint = var2;
    }

    public void close() {
        UsbDeviceConnection var1 = this.usbDeviceConnection;
        if (var1 != null) {
            if (!var1.releaseInterface(this.usbInterface)) {
                Log.e("USB", "could not release interface!");
            }

            this.usbDeviceConnection.close();
            Log.e("USB", "closed-----------------v");
        }
    }

    int getSectorSize() {
        return this.scsiReadCapacity.getSectorSize();
    }

    int getSectors() {
        return this.scsiReadCapacity.getSectors();
    }

    public UsbDeviceConnection getUsbDeviceConnection() {
        return this.usbDeviceConnection;
    }

    public UsbInterface getUsbInterface() {
        return this.usbInterface;
    }

    byte[] read(int param1, int param2) {
        // $FF: Couldn't be decompiled
    }

    public void readCapacity() {
        try {
            this.scsiReadCapacity.readCapacity(this.commandBlockWrapper,
this.commandStatusWrapper, this.usbDeviceConnection, this.inputEndpoint,
this.outputEndpoint);
        } catch (UsbCommanException var2) {
            var2.printStackTrace();
        }

    }

    void write(int param1, int param2, byte[] param3) {
        // $FF: Couldn't be decompiled
```

Додаток Г. Лістинг модуля побудови

Лістинг модуля побудови та опрацювання даних в конфігурації

```
public enum ConfigType implements Serializable {
    private static final ConfigType[] $VALUES;
    CID_18K,
    CID_18K_INTEGRAL,
    ORION_16,
    ORION_16i,
    ORION_4,
    ORION_4i,
    ORION_8,
    ORION_8_PLUS,
    ORION_8i,
    ORION_L,
    ORION_Li,
    ORION_M,
    ORION_Mi,
    ORION_S,
    ORION_Si,
    ORION_XS,
    ORION_XSi;

    public final int deviceId;
    public final int modelListPosition;
    public final String name;
    public final String secondName;
    private final String updateFileName;

    static {
        ConfigType var0 = new ConfigType("CID_18K", 0, "CID-18k-GPRS", "18kCID", 0, 183,
"B71.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var2 != 1) {
                    if (var2 != 2) {
                        if (var2 != 3) {
                            if (var2 != 4) {
                                if (var2 == 5) {
                                    return new CID18kConfig();
                                } else {
                                    throw new UnsupportedConfigVersionException(this,
var2);
                                }
                            } else {
                                return new
ua.tiras.loader.core.data.cid18k.v4.configs.CID18kConfig();
                            }
                        } else {
                            return new
ua.tiras.loader.core.data.cid18k.v3.configs.CID18kConfig();
                        }
                    } else {
                        return new
ua.tiras.loader.core.data.cid18k.v2.configs.CID18kConfig();
                    }
                } else {
                    return new ua.tiras.loader.core.data.cid18k.configs.CID18kConfig();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                return (Integer)_EL.getOrDefault(CID18kValues.getCountsMap(), var3, 0);
            }
        };
        CID_18K = var0;
        ConfigType var1 = new ConfigType("CID_18K_INTEGRAL", 1, "18kCIDi", "18kCID-I_1", 1,
183, "B71.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
```

```
UnsupportedConfigVersionException, SavingException {
                if (var2 != 4) {
                    if (var2 == 5) {
                        return new CID18kIntegralConfig();
                    } else {
                        throw new UnsupportedConfigVersionException(this, var2);
                    }
                } else {
                    return new
ua.tiras.loader.core.data.cid18k_integral.v4.config.CID18kIntegralConfig();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                return (Integer)_EL.getOrDefault(CID18kValues.getCountsMap(), var3, 0);
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        CID_18K_INTEGRAL = var1;
        ConfigType var2 = new ConfigType("ORION_4", 2, "Orion NOVA 4", "Or 4/1", 3, 157,
"9D.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var2 != 1) {
                    if (var2 != 2) {
                        if (var2 != 3) {
                            if (var2 != 4) {
                                if (var2 == 5) {
                                    return new Orion4Config();
                                } else {
                                    throw new UnsupportedConfigVersionException(this,
var2);
                                }
                            } else {
                                return new
ua.tiras.loader.core.data.nova4.v4.configs.Orion4Config();
                            }
                        } else {
                            return new
ua.tiras.loader.core.data.nova4.v3.configs.Orion4Config();
                        }
                    } else {
                        return new
ua.tiras.loader.core.data.nova4.v2.configs.Orion4Config();
                    }
                } else {
                    return new ua.tiras.loader.core.data.nova4.configs.Orion4Config();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                if (var2 != 3 && var2 != 4) {
                    return var2 != 5 ?
(Integer)_EL.getOrDefault(Orion4Values.getCountsMap(), var3, 0) :
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova4.v5.configs.Orion4Values.INSTANCE.
getCountsMap(), var3, 0);
                } else {
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova4.v3.configs.Orion4Values.getCounts
Map(), var3, 0);
                }
            }

            public String getNameForUpdateFile(int var1) {
                return "9D1.HEX";
            }
```

```
            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN1;
            }
        };
        ORION_4 = var2;
        ConfigType var3 = new ConfigType("ORION_8", 3, "Orion NOVA 8", "Or 8/1", 4, 158,
"9E.HEX") {
            public MainConfig createConfig(int var1, int var2) throws SavingException,
UnsupportedConfigVersionException {
                if (var2 != 1) {
                    if (var2 != 2) {
                        if (var2 != 3) {
                            if (var2 != 4) {
                                if (var2 == 5) {
                                    return new Orion8Config();
                                } else {
                                    throw new UnsupportedConfigVersionException(this,
var2);
                                }
                            } else {
                                return new
ua.tiras.loader.core.data.nova8.v4.configs.Orion8Config();
                            }
                        } else {
                            return new
ua.tiras.loader.core.data.nova8.v3.configs.Orion8Config();
                        }
                    } else {
                        return new
ua.tiras.loader.core.data.nova8.v2.configs.Orion8Config();
                    }
                } else {
                    return new ua.tiras.loader.core.data.nova8.configs.Orion8Config();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                if (var2 != 3 && var2 != 4) {
                    return var2 != 5 ?
(Integer)_EL.getOrDefault(Orion8Values.getCountsMap(), var3, 0) :
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova8.v5.configs.Orion8Values.INSTANCE.
getCountsMap(), var3, 0);
                } else {
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova8.v3.configs.Orion8Values.getCounts
Map(), var3, 0);
                }
            }

            public String getNameForUpdateFile(int var1) {
                return "9E1.HEX";
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN1;
            }
        };
        ORION_8 = var3;
        ConfigType var4 = new ConfigType("ORION_16", 4, "Orion NOVA 16", "Or 16/1", 5, 182,
"B6.HEX") {
            public MainConfig createConfig(int var1, int var2) throws SavingException,
UnsupportedConfigVersionException {
                if (var2 != 1) {
                    if (var2 != 2) {
                        if (var2 != 3) {
                            if (var2 != 4) {
                                if (var2 == 5) {
                                    return new Orion16Config();
                                } else {
                                    throw new UnsupportedConfigVersionException(this,
```

```
var2);
                                }
                            } else {
                                return new
ua.tiras.loader.core.data.nova16.v4.Orion16Config();
                            }
                        } else {
                            return new
ua.tiras.loader.core.data.nova16.v3.configs.Orion16Config();
                        }
                    } else {
                        return new
ua.tiras.loader.core.data.nova16.v2.configs.Orion16Config();
                    }
                } else {
                    return new ua.tiras.loader.core.data.nova16.configs.Orion16Config();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                if (var2 != 3 && var2 != 4) {
                    return var2 != 5 ?
(Integer)_EL.getOrDefault(Orion16Values.getCountsMap(), var3, 0) :
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova16.v5.configs.Orion16Values.INSTANC
E.getCountsMap(), var3, 0);
                } else {
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova16.v3.configs.Orion16Values.getCoun
tsMap(), var3, 0);
                }
            }

            public String getNameForUpdateFile(int var1) {
                return "B61.HEX";
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN1;
            }
        };
        ORION_16 = var4;
        ConfigType var5 = new ConfigType("ORION_4i", 5, "Orion NOVA 4i", "Or 4/1", 6, 186,
"BA.HEX") {
            public MainConfig createConfig(int var1, int var2) throws SavingException,
UnsupportedConfigVersionException {
                if (var2 != 3) {
                    if (var2 != 4) {
                        if (var2 == 5) {
                            return new Orion4iConfig();
                        } else {
                            throw new UnsupportedConfigVersionException(this, var2);
                        }
                    } else {
                        return new
ua.tiras.loader.core.data.nova_iSeries.v4.configs.Orion4iConfig();
                    }
                } else {
                    return new
ua.tiras.loader.core.data.nova_iSeries.configs.Orion4iConfig();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                Integer var4 = 0;
                return var2 != 5 ? (Integer)_EL.getOrDefault(Orion4Values.getCountsMap(),
var3, var4) :
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova4.v5.configs.Orion4Values.INSTANCE.
getCountsMap(), var3, var4);
            }
```

```
            public String getNameForUpdateFile(int var1) {
                return "BA1.HEX";
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN1;
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        ORION_4i = var5;
        ConfigType var6 = new ConfigType("ORION_8i", 6, "Orion NOVA 8i", "Or 8/1", 7, 187,
"BB.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var2 != 3) {
                    if (var2 != 4) {
                        if (var2 == 5) {
                            return new Orion8iConfig();
                        } else {
                            throw new UnsupportedConfigVersionException(this, var2);
                        }
                    } else {
                        return new
ua.tiras.loader.core.data.nova_iSeries.v4.configs.Orion8iConfig();
                    }
                } else {
                    return new
ua.tiras.loader.core.data.nova_iSeries.configs.Orion8iConfig();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                Integer var4 = 0;
                return var2 != 5 ? (Integer)_EL.getOrDefault(Orion8Values.getCountsMap(),
var3, var4) :
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova8.v5.configs.Orion8Values.INSTANCE.
getCountsMap(), var3, var4);
            }

            public String getNameForUpdateFile(int var1) {
                return "BB1.HEX";
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN1;
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        ORION_8i = var6;
        ConfigType var7 = new ConfigType("ORION_16i", 7, "Orion NOVA 16i", "Or 16/1", 8,
188, "BC.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var2 != 3) {
                    if (var2 != 4) {
                        if (var2 == 5) {
                            return new Orion16iConfig();
                        } else {
                            throw new UnsupportedConfigVersionException(this, var2);
                        }
                    } else {
                        return new
ua.tiras.loader.core.data.nova_iSeries.v4.configs.Orion16iConfig();
                    }
```

```
                } else {
                    return new
ua.tiras.loader.core.data.nova_iSeries.configs.Orion16iConfig();
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                Integer var4 = 0;
                return var2 != 5 ? (Integer)_EL.getOrDefault(Orion16Values.getCountsMap(),
var3, var4) :
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova16.v5.configs.Orion16Values.INSTANC
E.getCountsMap(), var3, var4);
            }

            public String getNameForUpdateFile(int var1) {
                return "BC1.HEX";
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN1;
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        ORION_16i = var7;
        ConfigType var8 = new ConfigType("ORION_8_PLUS", 8,
StrHelper.getString(string.orion8_plus_conf_name), "Or8plus-1", 9, 142, "8E1.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var2 == 1) {
                    return new Orion8PlusConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var2);
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                return (Integer)_EL.getOrDefault(Orion8PlusValues.getCountsMap(), var3, 0);
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN1;
            }
        };
        ORION_8_PLUS = var8;
        ConfigType var9 = new ConfigType("ORION_XS", 9, "Orion NOVA XS", "OrXS", 11, 192,
"C0.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 0 && var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 0:
                    case 1:
                        return new OrionXSConfig();
                    case 2:
                        return new
ua.tiras.loader.core.data.nova_xs.v2.configs.OrionXSConfig();
                    case 3:
                        return new
ua.tiras.loader.core.data.nova_xs.v3.configs.OrionXSConfig();
                    case 4:
                        return new
ua.tiras.loader.core.data.nova_xs.v4.configs.OrionXSConfig();
                    case 5:
```

```
                        return new
ua.tiras.loader.core.data.nova_xs.v5.config.OrionXSConfig();
                    case 6:
                        return new
ua.tiras.loader.core.data.nova_xs.v6.config.OrionXSConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_xs.v7.config.OrionXSConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_xs.v8.configs.OrionXSConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v5.configs.OrionXSConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v6.configs.OrionXSConfig();
                } else if (var2 == 7) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v7.configs.OrionXSConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v8.configs.OrionXSConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                label32: {
                    if (var1 != 1) {
                        if (var1 != 2) {
                            break label32;
                        }
                    } else {
                        switch(var2) {
                        case 0:
                        case 1:
                        case 2:
                        case 3:
                        case 4:
                        case 5:
                        case 6:
                        case 7:
                        case 8:
                            return (Integer)_EL.getOrDefault(OrionXSValues.getCountsMap(),
var3, 0);
                        }
                    }

                    if (var2 != 5 && var2 != 6) {
                        if (var2 != 7 && var2 != 8) {
                            break label32;
                        }

                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_xs.h2.v7.configs.OrionXSValues.get
CountsMap(), var3, 0);
                    }

                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_xs.h2.v5.configs.OrionXSValues.get
CountsMap(), var3, 0);
                }

                StringBuilder var4 = new StringBuilder();
                var4.append("No such version: ");
```

```
                var4.append(var1);
                var4.append(".");
                var4.append(var2);
                throw new IllegalArgumentException(var4.toString());
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C0");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }
        };
        ORION_XS = var9;
        ConfigType var10 = new ConfigType("ORION_S", 10, "Orion NOVA S", "OrS", 12, 193,
"C1.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 0:
                    case 1:
                        return new OrionSConfig();
                    case 2:
                        return new
ua.tiras.loader.core.data.nova_s.v2.configs.OrionSConfig();
                    case 3:
                        return new
ua.tiras.loader.core.data.nova_s.v3.configs.OrionSConfig();
                    case 4:
                        return new
ua.tiras.loader.core.data.nova_s.v4.configs.OrionSConfig();
                    case 5:
                        return new
ua.tiras.loader.core.data.nova_s.v5.config.OrionSConfig();
                    case 6:
                        return new
ua.tiras.loader.core.data.nova_s.v6.configs.OrionSConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_s.v7.configs.OrionSConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_s.v8.configs.OrionSConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v5.configs.OrionSConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v6.configs.OrionSConfig();
                } else if (var2 == 7) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v7.configs.OrionSConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v8.configs.OrionSConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
```

```
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                Integer var4 = 0;
                switch(var2) {
                case 0:
                case 1:
                    return (Integer)_EL.getOrDefault(OrionSValues.getCountsMap(), var3,
var4);
                case 2:
                case 3:
                case 4:
                case 5:
                case 6:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_s.v2.configs.OrionSValues.getCount
sMap(), var3, var4);
                case 7:
                case 8:
                    if (var1 == 1) {
                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_s.v7.configs.OrionSValues.getCount
sMap(), var3, var4);
                    } else if (var1 == 2) {
                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_s.h2.v7.configs.OrionSValues.getCo
untsMap(), var3, var4);
                    }
                default:
                    StringBuilder var5 = new StringBuilder();
                    var5.append("No such version: ");
                    var5.append(var2);
                    throw new IllegalArgumentException(var5.toString());
                }
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C1");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }
        };
        ORION_S = var10;
        ConfigType var11 = new ConfigType("ORION_M", 11, "Orion NOVA M", "OrM", 13, 194,
"C2.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 0:
                    case 1:
                        return new OrionMConfig();
                    case 2:
                        return new
ua.tiras.loader.core.data.nova_m.v2.configs.OrionMConfig();
                    case 3:
                        return new
ua.tiras.loader.core.data.nova_m.v3.configs.OrionMConfig();
                    case 4:
```

```
                        return new
ua.tiras.loader.core.data.nova_m.v4.configs.OrionMConfig();
                    case 5:
                        return new
ua.tiras.loader.core.data.nova_m.v5.config.OrionMConfig();
                    case 6:
                        return new
ua.tiras.loader.core.data.nova_m.v6.configs.OrionMConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_m.v7.configs.OrionMConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_m.v8.configs.OrionMConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v5.configs.OrionMConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v6.configs.OrionMConfig();
                } else if (var2 == 7) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v7.configs.OrionMConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v8.configs.OrionMConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                label37: {
                    if (var1 != 1) {
                        if (var1 != 2) {
                            break label37;
                        }
                    } else {
                        switch(var2) {
                        case 0:
                        case 1:
                            return (Integer)_EL.getOrDefault(OrionMValues.getCountsMap(),
var3, 0);
                        case 2:
                        case 3:
                        case 4:
                        case 5:
                        case 6:
                            return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.v2.configs.OrionMValues.getCount
sMap(), var3, 0);
                        case 7:
                        case 8:
                            return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.v7.configs.OrionMValues.getCount
sMap(), var3, 0);
                        }
                    }

                    if (var2 == 5 || var2 == 6) {
                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.h2.v5.configs.OrionMValues.getCo
untsMap(), var3, 0);
                    }

                    if (var2 == 7 || var2 == 8) {
                        return
```

```
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.h2.v7.configs.OrionMValues.getCo
untsMap(), var3, 0);
                    }
                }

                StringBuilder var4 = new StringBuilder();
                var4.append("No such version: ");
                var4.append(var1);
                var4.append(".");
                var4.append(var2);
                throw new IllegalArgumentException(var4.toString());
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C2");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }
        };
        ORION_M = var11;
        ConfigType var12 = new ConfigType("ORION_L", 12, "Orion NOVA L", "OrL", 14, 195,
"C3.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 0:
                    case 1:
                        return new OrionLConfig();
                    case 2:
                        return new
ua.tiras.loader.core.data.nova_l.v2.configs.OrionLConfig();
                    case 3:
                        return new
ua.tiras.loader.core.data.nova_l.v3.configs.OrionLConfig();
                    case 4:
                        return new
ua.tiras.loader.core.data.nova_l.v4.configs.OrionLConfig();
                    case 5:
                        return new
ua.tiras.loader.core.data.nova_l.v5.config.OrionLConfig();
                    case 6:
                        return new
ua.tiras.loader.core.data.nova_l.v6.configs.OrionLConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_l.v7.configs.OrionLConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_l.v8.configs.OrionLConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_l.h2.v5.configs.OrionLConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_l.h2.v6.configs.OrionLConfig();
                } else if (var2 == 7) {
```

```
                    return new
ua.tiras.loader.core.data.nova_l.h2.v7.configs.OrionLConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_l.h2.v8.configs.OrionLConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                Integer var4 = 0;
                switch(var2) {
                case 0:
                case 1:
                    return (Integer)_EL.getOrDefault(OrionLValues.getCountsMap(), var3,
var4);
                case 2:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_l.v2.configs.OrionLValues.getCount
sMap(), var3, var4);
                case 3:
                case 4:
                case 5:
                case 6:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_l.v3.configs.OrionLValues.getCount
sMap(), var3, var4);
                case 7:
                case 8:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_l.v7.configs.OrionLValues.getCount
sMap(), var3, var4);
                default:
                    StringBuilder var5 = new StringBuilder();
                    var5.append("No such version: ");
                    var5.append(var2);
                    throw new IllegalArgumentException(var5.toString());
                }
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C3");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }
        };
        ORION_L = var12;
        ConfigType var13 = new ConfigType("ORION_XSi", 13, "Orion NOVA XS(i)", "OrXSi", 15,
197, "C5.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 3:
                        return new OrionXSiConfig();
                    case 4:
                        return new
ua.tiras.loader.core.data.nova_xs.v4.integral.OrionXSiConfig();
                    case 5:
```

```
                        return new
ua.tiras.loader.core.data.nova_xs.v5.integral.OrionXSiConfig();
                    case 6:
                        return new
ua.tiras.loader.core.data.nova_xs.v6.integral.OrionXSiConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_xs.v7.integral.OrionXSiConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_xs.v8.integral.OrionXSiConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v5.integral.OrionXSiConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v6.integral.OrionXSiConfig();
                } else if (var2 == 7) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v7.integral.OrionXSiConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_xs.h2.v8.integral.OrionXSiConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                label32: {
                    if (var1 != 1) {
                        if (var1 != 2) {
                            break label32;
                        }
                    } else {
                        switch(var2) {
                        case 0:
                        case 1:
                        case 2:
                        case 3:
                        case 4:
                        case 5:
                        case 6:
                        case 7:
                        case 8:
                            return (Integer)_EL.getOrDefault(OrionXSValues.getCountsMap(),
var3, 0);
                        }
                    }

                    if (var2 != 5 && var2 != 6) {
                        if (var2 != 7 && var2 != 8) {
                            break label32;
                        }

                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_xs.h2.v7.configs.OrionXSValues.get
CountsMap(), var3, 0);
                    }

                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_xs.h2.v5.configs.OrionXSValues.get
CountsMap(), var3, 0);
                }

                StringBuilder var4 = new StringBuilder();
                var4.append("No such version: ");
```

```
                var4.append(var1);
                var4.append(".");
                var4.append(var2);
                throw new IllegalArgumentException(var4.toString());
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C5");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        ORION_XSi = var13;
        ConfigType var14 = new ConfigType("ORION_Si", 14, "Orion NOVA S(i)", "OrSi", 16,
198, "C6.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 3:
                        return new OrionSiConfig();
                    case 4:
                        return new
ua.tiras.loader.core.data.nova_s.v4.integral.OrionSiConfig();
                    case 5:
                        return new
ua.tiras.loader.core.data.nova_s.v5.integral.OrionSiConfig();
                    case 6:
                        return new
ua.tiras.loader.core.data.nova_s.v6.integral.OrionSiConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_s.v7.integral.OrionSiConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_s.v8.integral.OrionSiConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v5.integral.OrionSiConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v6.integral.OrionSiConfig();
                } else if (var2 == 7) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v7.integral.OrionSiConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_s.h2.v8.integral.OrionSiConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
                }
            }
```

```
            public int getConfigCount(int var1, int var2, Counts var3) {
                Integer var4 = 0;
                switch(var2) {
                case 0:
                case 1:
                    return (Integer)_EL.getOrDefault(OrionSValues.getCountsMap(), var3,
var4);
                case 2:
                case 3:
                case 4:
                case 5:
                case 6:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_s.v2.configs.OrionSValues.getCount
sMap(), var3, var4);
                case 7:
                case 8:
                    if (var1 == 1) {
                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_s.v7.configs.OrionSValues.getCount
sMap(), var3, var4);
                    } else if (var1 == 2) {
                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_s.h2.v7.configs.OrionSValues.getCo
untsMap(), var3, var4);
                    }
                default:
                    StringBuilder var5 = new StringBuilder();
                    var5.append("No such version: ");
                    var5.append(var2);
                    throw new IllegalArgumentException(var5.toString());
                }
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C6");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        ORION_Si = var14;
        ConfigType var15 = new ConfigType("ORION_Mi", 15, "Orion NOVA M(i)", "OrMi", 17,
199, "C7.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 3:
                        return new OrionMiConfig();
                    case 4:
                        return new
ua.tiras.loader.core.data.nova_m.v4.integral.OrionMiConfig();
                    case 5:
                        return new
ua.tiras.loader.core.data.nova_m.v5.integral.OrionMiConfig();
                    case 6:
```

```
                        return new
ua.tiras.loader.core.data.nova_m.v6.integral.OrionMiConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_m.v7.integral.OrionMiConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_m.v8.integral.OrionMiConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v5.integral.OrionMiConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v6.integral.OrionMiConfig();
                } else if (var2 == 7) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v7.integral.OrionMiConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_m.h2.v8.integral.OrionMiConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
                }
            }

            public int getConfigCount(int var1, int var2, Counts var3) {
                label37: {
                    if (var1 != 1) {
                        if (var1 != 2) {
                            break label37;
                        }
                    } else {
                        switch(var2) {
                        case 0:
                        case 1:
                            return (Integer)_EL.getOrDefault(OrionMValues.getCountsMap(),
var3, 0);
                        case 2:
                        case 3:
                        case 4:
                        case 5:
                        case 6:
                            return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.v2.configs.OrionMValues.getCount
sMap(), var3, 0);
                        case 7:
                        case 8:
                            return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.v7.configs.OrionMValues.getCount
sMap(), var3, 0);
                        }
                    }

                    if (var2 == 5 || var2 == 6) {
                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.h2.v5.configs.OrionMValues.getCo
untsMap(), var3, 0);
                    }

                    if (var2 == 7 || var2 == 8) {
                        return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_m.h2.v7.configs.OrionMValues.getCo
untsMap(), var3, 0);
                    }
                }

                StringBuilder var4 = new StringBuilder();
```

```
                var4.append("No such version: ");
                var4.append(var1);
                var4.append(".");
                var4.append(var2);
                throw new IllegalArgumentException(var4.toString());
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C7");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        ORION_Mi = var15;
        ConfigType var16 = new ConfigType("ORION_Li", 16, "Orion NOVA L(i)", "OrLi", 18,
200, "C8.HEX") {
            public MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
                if (var1 != 1) {
                    if (var1 != 2) {
                        throw new UnsupportedConfigVersionException(this, var1, var2);
                    }
                } else {
                    switch(var2) {
                    case 3:
                        return new OrionLiConfig();
                    case 4:
                        return new
ua.tiras.loader.core.data.nova_l.v4.integral.OrionLiConfig();
                    case 5:
                        return new
ua.tiras.loader.core.data.nova_l.v5.integral.OrionLiConfig();
                    case 6:
                        return new
ua.tiras.loader.core.data.nova_l.v6.integral.OrionLiConfig();
                    case 7:
                        return new
ua.tiras.loader.core.data.nova_l.v7.integral.OrionLiConfig();
                    case 8:
                        return new
ua.tiras.loader.core.data.nova_l.v8.integral.OrionLiConfig();
                    }
                }

                if (var2 == 5) {
                    return new
ua.tiras.loader.core.data.nova_l.h2.v5.integral.OrionLiConfig();
                } else if (var2 == 6) {
                    return new
ua.tiras.loader.core.data.nova_l.h2.v6.integral.OrionLiConfig();
                } else if (var2 == 7) {
                    return new
ua.tiras.loader.core.data.nova_l.h2.v7.integral.OrionLiConfig();
                } else if (var2 == 8) {
                    return new
ua.tiras.loader.core.data.nova_l.h2.v8.integral.OrionLiConfig();
                } else {
                    throw new UnsupportedConfigVersionException(this, var1, var2);
                }
            }
```

```
            public int getConfigCount(int var1, int var2, Counts var3) {
                Integer var4 = 0;
                switch(var2) {
                case 0:
                case 1:
                    return (Integer)_EL.getOrDefault(OrionLValues.getCountsMap(), var3,
var4);
                case 2:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_l.v2.configs.OrionLValues.getCount
sMap(), var3, var4);
                case 3:
                case 4:
                case 5:
                case 6:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_l.v3.configs.OrionLValues.getCount
sMap(), var3, var4);
                case 7:
                case 8:
                    return
(Integer)_EL.getOrDefault(ua.tiras.loader.core.data.nova_l.v7.configs.OrionLValues.getCount
sMap(), var3, var4);
                default:
                    StringBuilder var5 = new StringBuilder();
                    var5.append("No such version: ");
                    var5.append(var2);
                    throw new IllegalArgumentException(var5.toString());
                }
            }

            public String getNameForUpdateFile(int var1) {
                StringBuilder var2 = new StringBuilder();
                var2.append("C8");
                var2.append(var1);
                var2.append(".HEX");
                return var2.toString();
            }

            public ConfigType.Generation getNovaGeneration() {
                return ConfigType.Generation.GEN2;
            }

            public boolean isIntegralModification() {
                return true;
            }
        };
        ORION_Li = var16;
        $VALUES = new ConfigType[]{var0, var1, var2, var3, var4, var5, var6, var7, var8,
var9, var10, var11, var12, var13, var14, var15, var16};
    }

    private ConfigType(String var3, String var4, int var5, int var6, String var7) {
        this.name = var3;
        this.secondName = var4;
        this.deviceId = var5;
        this.modelListPosition = var6;
        this.updateFileName = var7;
    }

    public static MainConfig createInstance(String var0, int var1) throws
UnsupportedConfigVersionException, SavingException {
        return getConfigTypeByName(var0).createConfig(var1);
    }

    public static MainConfig createInstance(String var0, int var1, int var2) throws
UnsupportedConfigVersionException, SavingException {
        return getConfigTypeByName(var0).createConfig(var1, var2);
    }
```

```
    public static ConfigType getConfigTypeByName(String var0) {
        ConfigType var2;
        for(int var1 = 0; var1 < values().length; ++var1) {
            var2 = values()[var1];
            if (var2.name.equals(var0)) {
                return var2;
            }
        }

        var2 = CID_18K;
        if (var0.equals(var2.secondName)) {
            return var2;
        } else {
            StringBuilder var3 = new StringBuilder();
            var3.append("There's no device with name '");
            var3.append(var0);
            var3.append("'");
            throw new IllegalArgumentException(var3.toString());
        }
    }

    @Deprecated
    public MainConfig createConfig(int var1) throws UnsupportedConfigVersionException,
SavingException {
        return this.createConfig(1, var1);
    }

    public abstract MainConfig createConfig(int var1, int var2) throws
UnsupportedConfigVersionException, SavingException;

    public abstract int getConfigCount(int var1, int var2, Counts var3);

    @Deprecated
    public int getConfigCount(int var1, Counts var2) {
        return this.getConfigCount(1, var1, var2);
    }

    public String getDeviceName() {
        return this.name;
    }

    public String getModelCodeHEX() {
        int var1 = this.updateFileName.indexOf(".");
        return this.updateFileName.substring(0, var1);
    }

    public String getNameForUpdateFile(int var1) {
        return this.updateFileName;
    }

    public ConfigType.Generation getNovaGeneration() {
        return null;
    }

    public boolean isIntegralModification() {
        return false;
    }

    public static enum Generation {
        private static final ConfigType.Generation[] $VALUES;
        GEN1,
        GEN2;

        static {
            ConfigType.Generation var0 = new ConfigType.Generation("GEN1", 0);
            GEN1 = var0;
            ConfigType.Generation var1 = new ConfigType.Generation("GEN2", 1);
            GEN2 = var1;
            $VALUES = new ConfigType.Generation[]{var0, var1};
```

Додаток Д. Ілюстративні матеріали

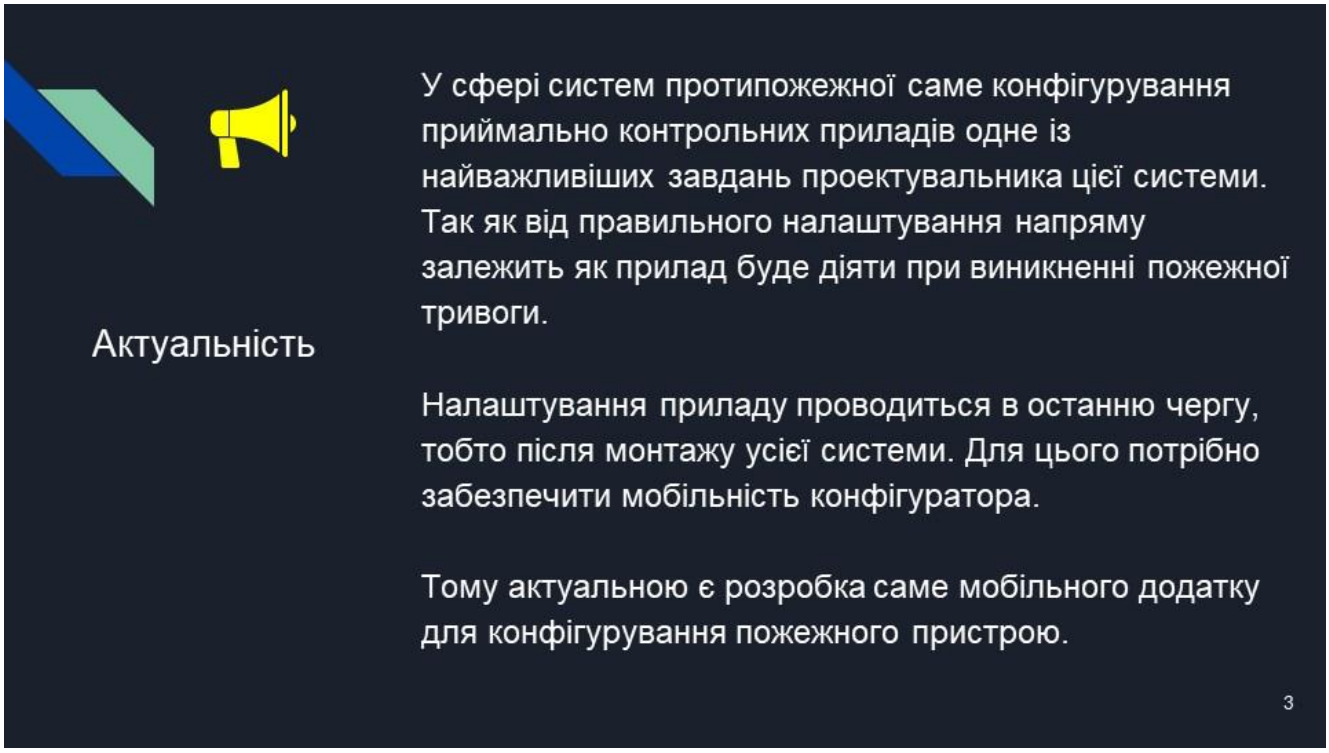
ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ КОНФІГУРУВАННЯ ПРИЙМАЛЬНО-КОНТРОЛЬНОГО ПОЖЕЖНОГО ПРИЛАДУ

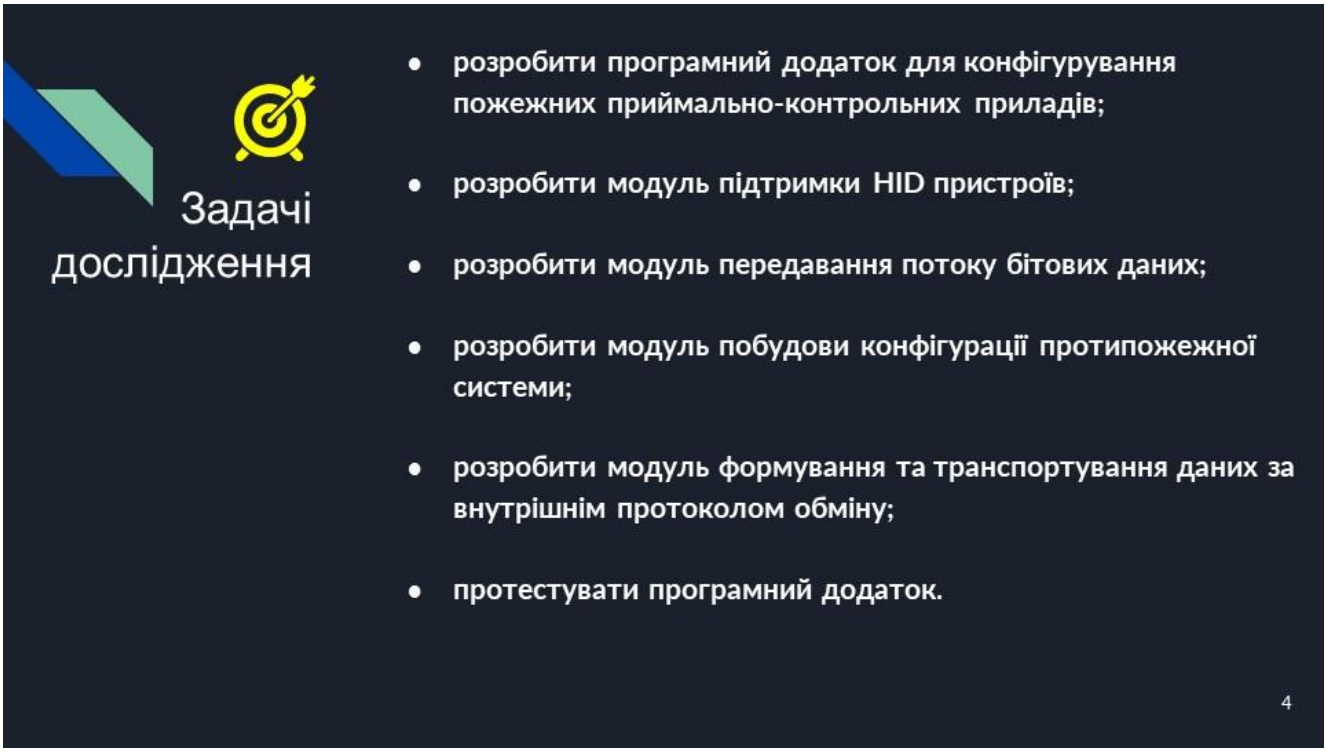
Плакат 1 – Тема, автор, науковий керівник магіської кваліфікаційної роботи

Плакат 2 – Загальна характеристика роботи

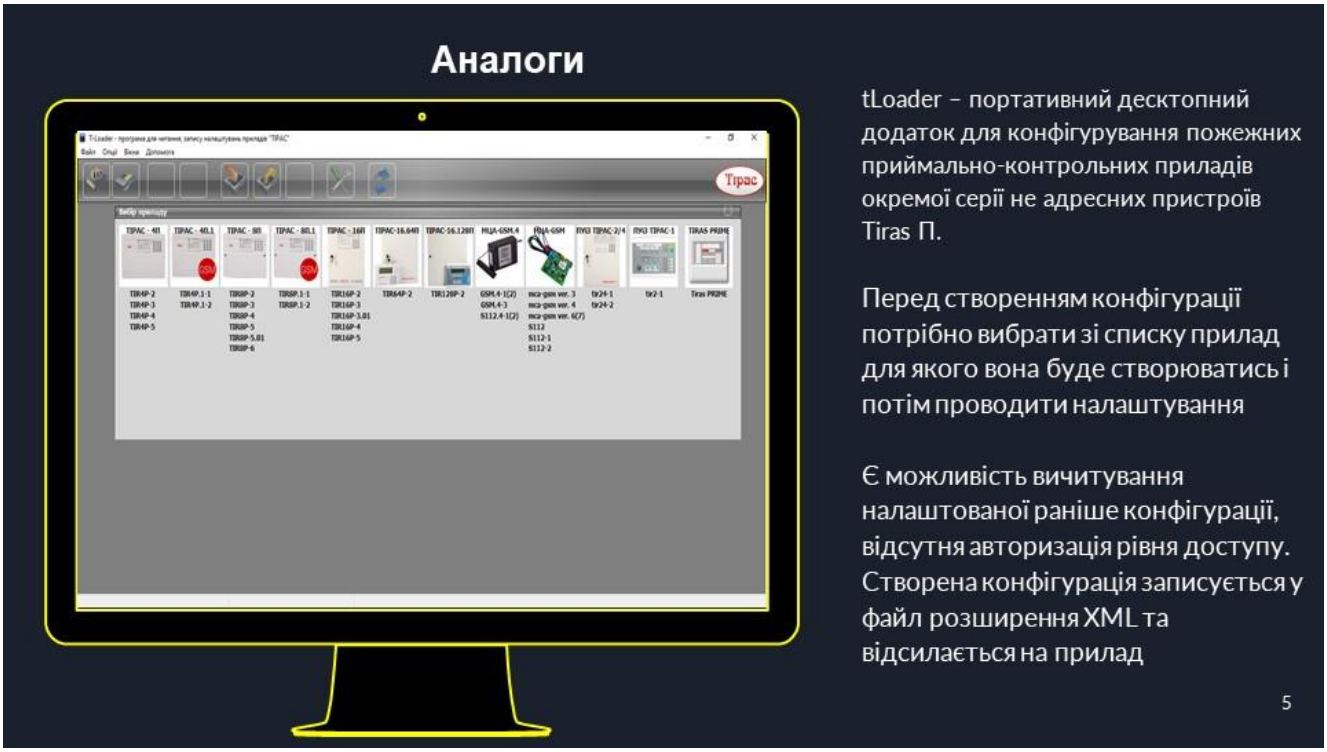
Плакат 3 – Актуальність розробки



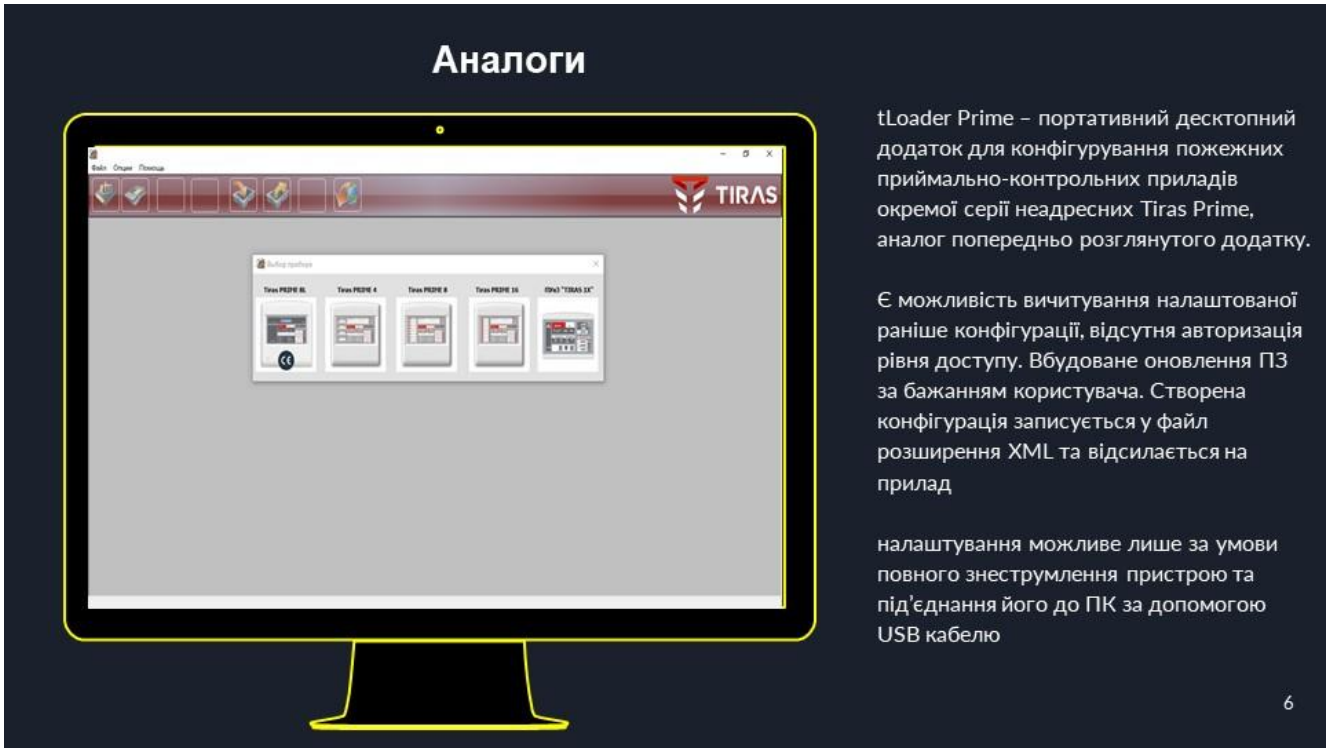
Плакат 4 – Задачі дослідження



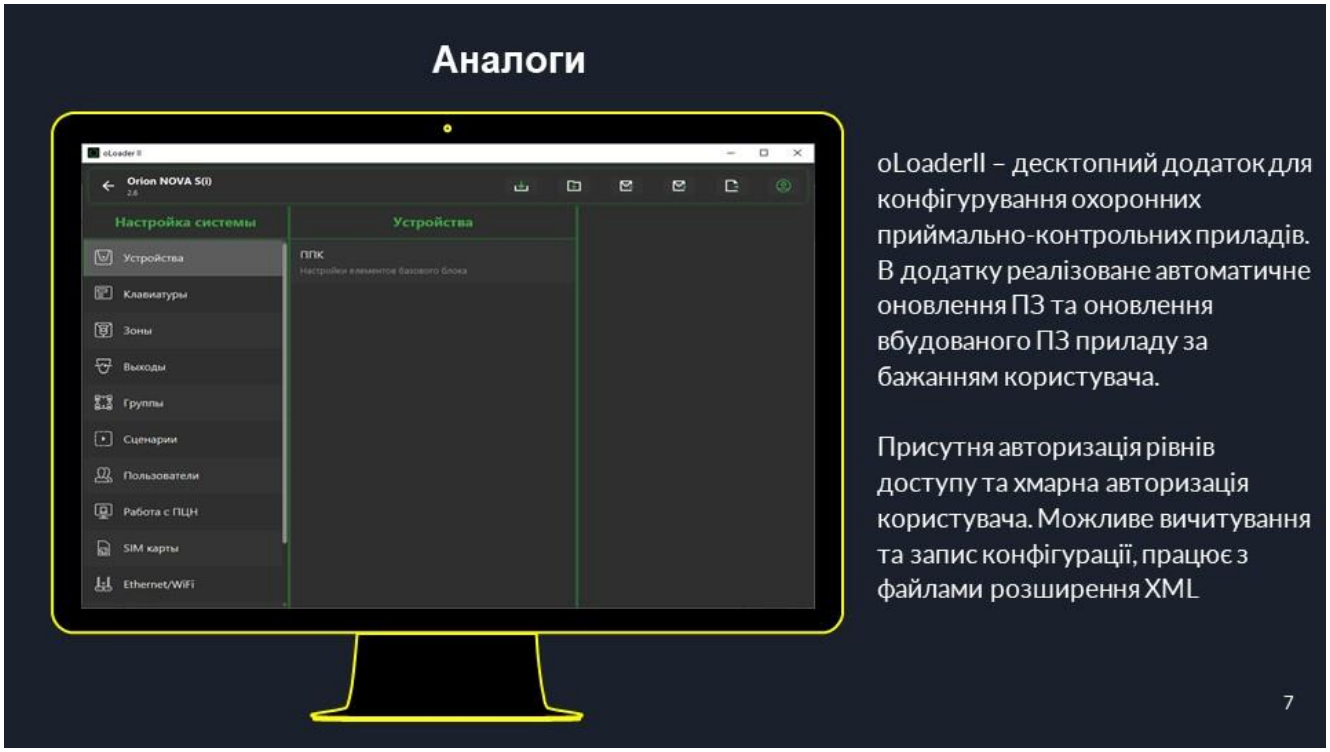
Плакат 5 – Аналоги додатків



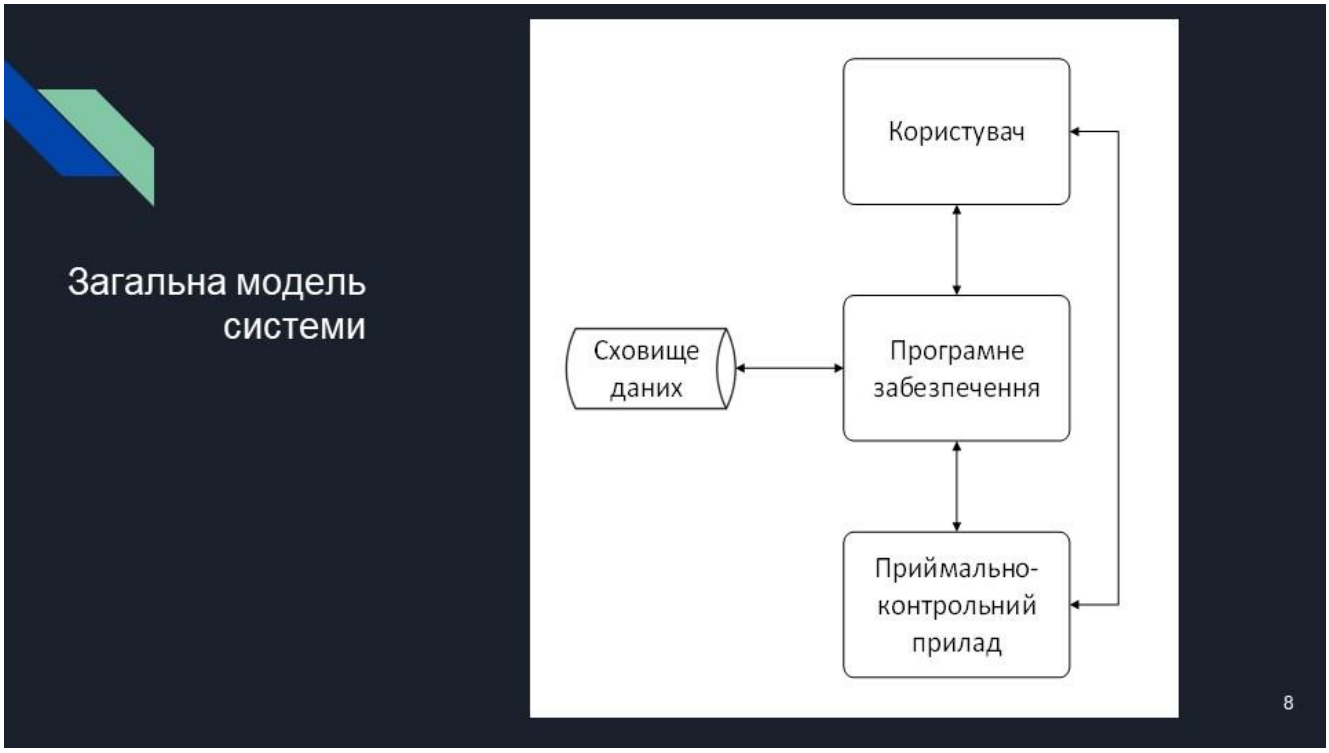
Плакат 6 – Аналоги додатків



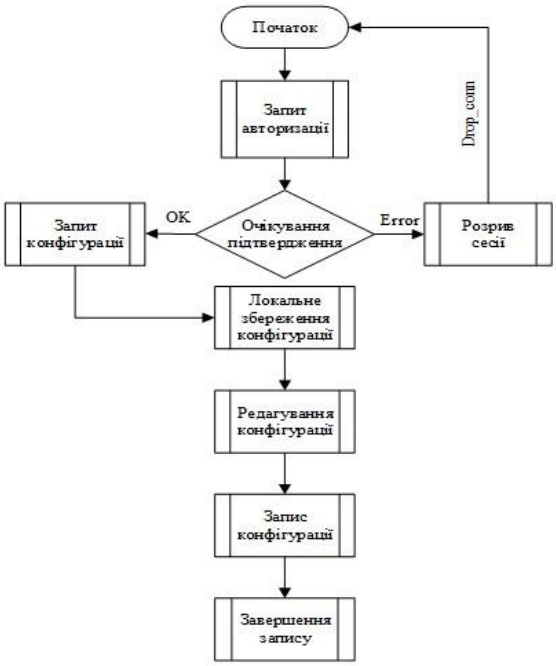
Плакат 7 – Аналоги додатків



Плакат 8 – Загальна модель системи



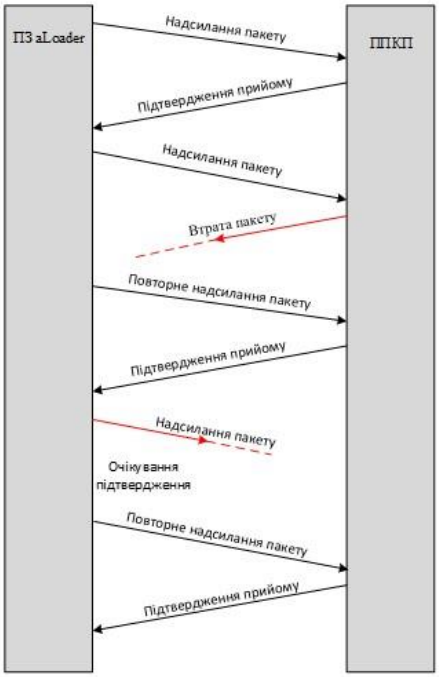
Плакат 9 – Алгоритм роботи додатку



Алгоритм роботи додатку

9

Плакат 10 – Протокол UDP

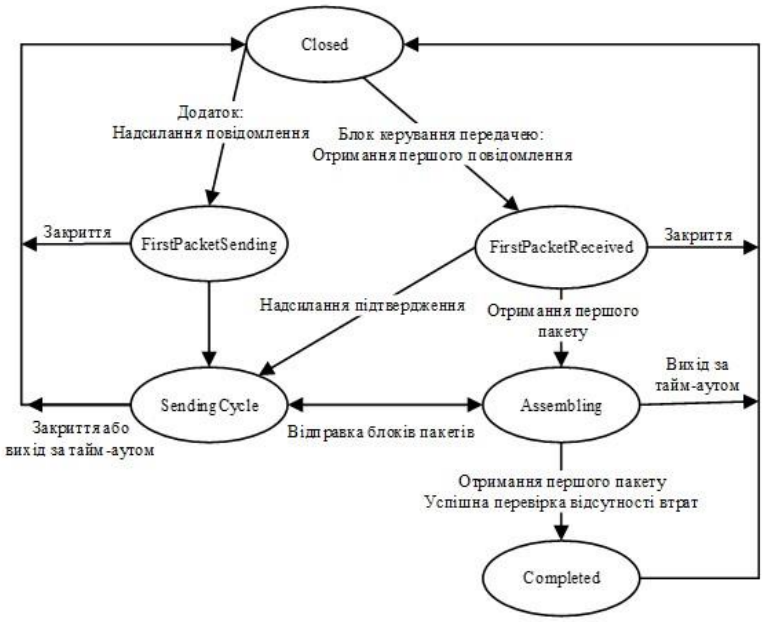


Коли з'єднання встановлено, починається надсилання даних. Дані передаються блоками пакетів. Кожен блок, крім останнього, містить фіксовану кількість пакетів. Воно дорівнює розміру вікна прийому/передачі. Останній блок даних може мати менше пакетів.

Після відправлення кожного блоку, сторона-відправник очікує на підтвердження про доставку, або запиту на повторну доставку втрачених пакетів, залишаючи відкритим вікно прийому/передачі для отримання відповідей. Після отримання підтвердження про доставку блоку, вікно прийому/передачі зсувається і відправляється наступний блок даних.

10

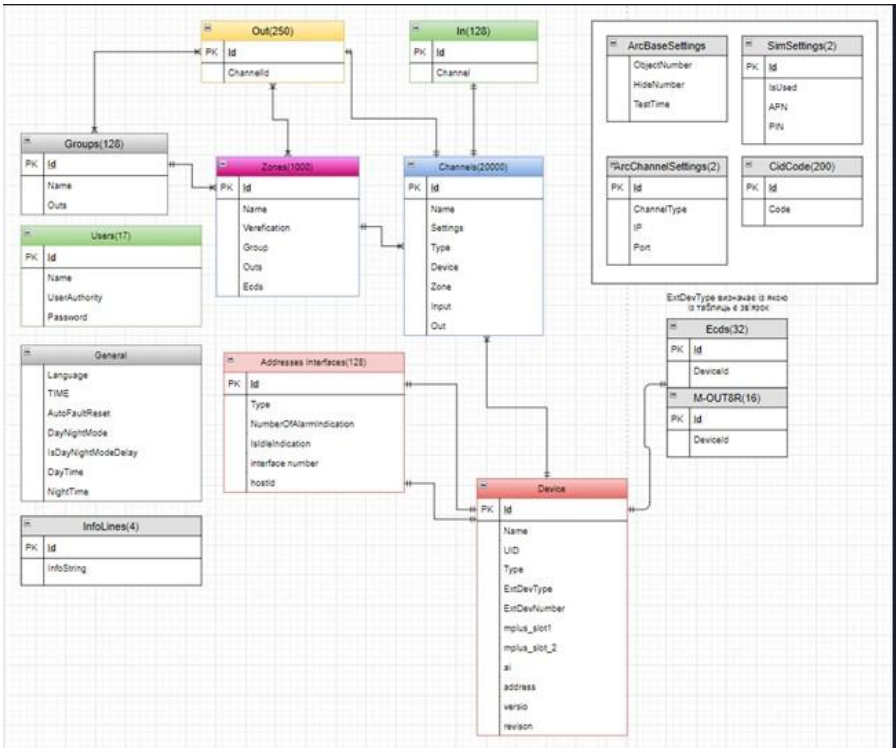
Плакат 11 – Діаграма станів



Діаграма станів Reliable UDP

11

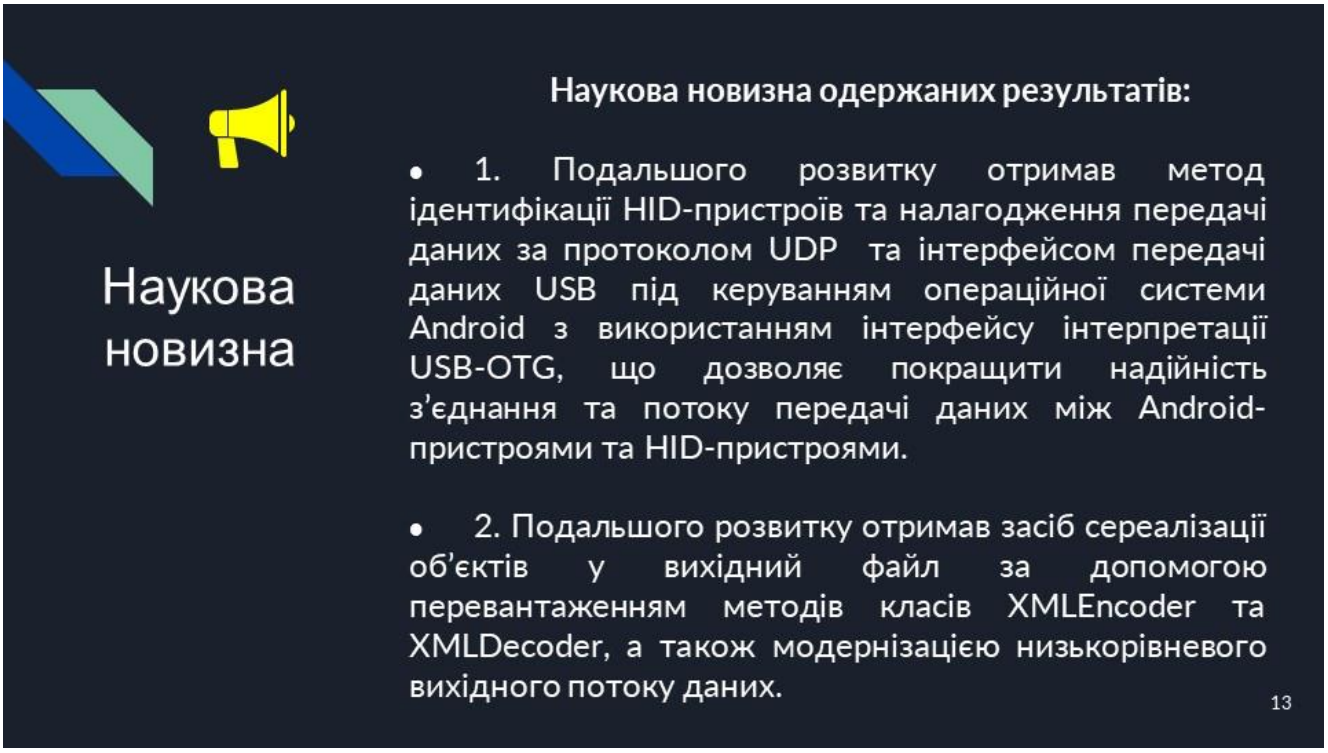
Плакат 12 – Діаграма класів



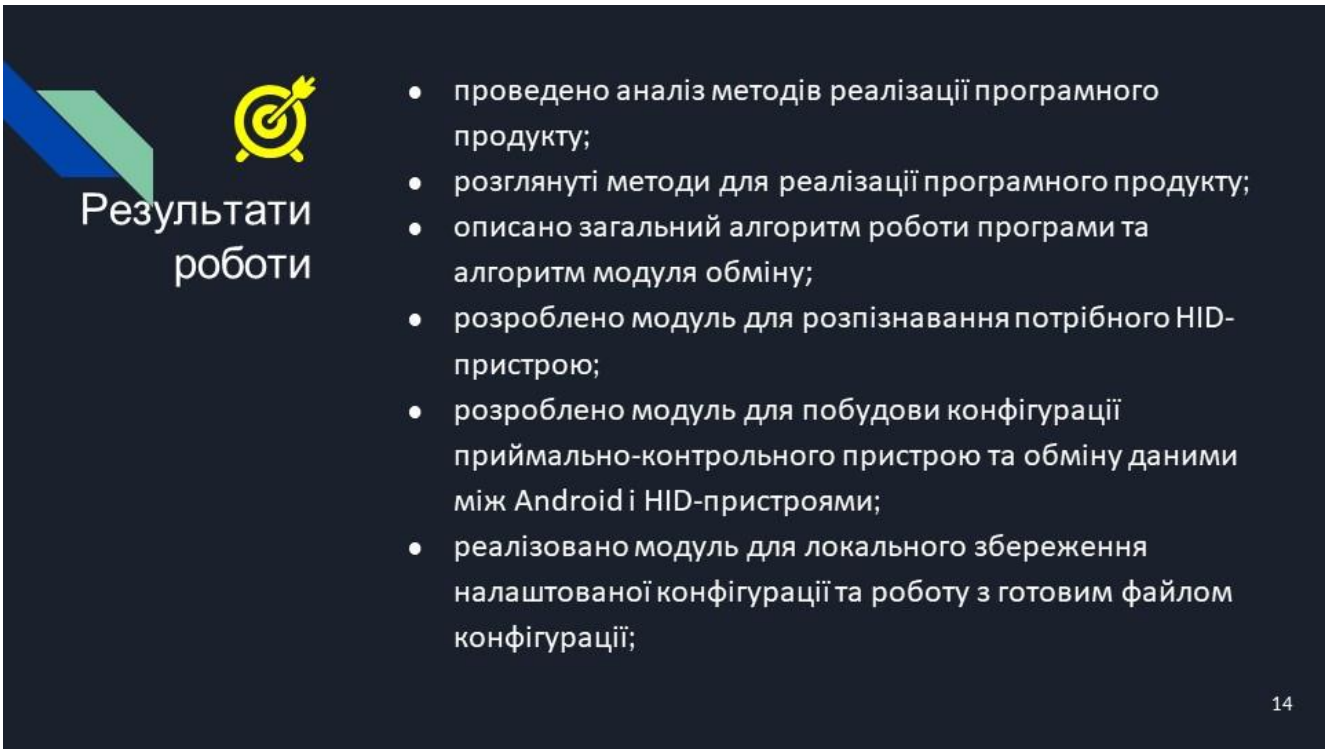
Діаграма класів модуля побудови конфігурації

12

Плакат 13 – Наукова новизна



Плакат 14 – Результати роботи



Плакат 15 – Публікації

Плакат 16 – Подяка