**Державний університет телекомунікацій**

Інститут захисту інформації

Факультет **Інформаційної безпеки**

Кафедра **Безпеки Інформаційних технологій**

Реферат

Тема: **«Програмний комплекс для порівняльного аналізу цілісності передавання даних у безпроводових каналах зв'язку 2,4-2,5 ГГц»**

Виконав:

Студент групи БСД - 41

Гребенюк Олександр Володимирович

Київ – 2014

# Вступ

Швидкий розвиток безпроводових технологій призвів до заповнення робочих частот і збільшення взаємного впливу безпроводових мереж одна на одну. Такий негативний влив призводить до погіршення умов передавання даних, тобто погіршення цілісності обміну інформацією в таких мережах. Виявлення вільних частот в діапазоні дозволяє динамічно переналаштовувати мережу, покращуючи цілісність даних (як одного з елементів забезпечення безпеки).

Аналізатори спектра дозволяють проводити сканування частотного діапазону в режимі реального часу, таким чином, відповідають вимогам для інструментів для забезпечення цілісності даних. Для вимірювання інтенсивності електромагнітного поля в діапазоні 2,4–2,5 ГГц використовуються промислові аналізатори спектру на мікросхемах Chipcon CC2500 і CC2511-F32, Cypress CYRF6934, CYRF6935 і CYRF6936, які дозволяють отримати більш-менш адекватне уявлення про стан ефіру. Але різні пристрої працюють на різних операційних системах і в різних програмних середовищах з різною частотою оновлення отриманої і графічної інформації, тому порівняння таких пристроїв становить значну проблему.

Таким чином, в даній роботі зроблено спробу створити одне програмне забезпечення, в якому можна порівнювати аналізатори спектрів різних версій від різних виробників. Хоча для розробки вибрана мова програмування Java, проте через особливості драйверів пристроїв не на всіх операційних системах працюють усі пристрої. Найбільш сумісною виявилася ОС Linux (дистрибутиви Ubunto, Mint і Gentoo), в якій підтримуються всі доступні автору аналізатори спектру.

# Абревіатури

|  |  |
| --- | --- |
| API | Application Programming Interface |
| CSS | Cascading Style Sheets |
| COM | Communication Port, Serial Port |
| CVS | Concurrent Versioning System |
| GNU | GNU’s Not UNIX |
| GNU GPL | GNU General Public License |
| HID | Human Interface Device |
| IRP | I/O Request Packet |
| JDK | Java Development Kit |
| JNI | Java Native Interface |
| JSON | JavaScript Object Notation |
| JVM | Java Virtual Machine |
| NIO | Non-blocking I/O |
| PID | Product Identifier |
| RSSI | Received Signal Strength Indication |
| UI | User Interface |
| USB | Universal Serial Bus |
| VID | Vendor Identifier |
| XML | Extensible Markup Language |
|  |  |
| ОС | Операційна система |

1. Огляд Бібліотек та програмного забезпечення
   1. Середовище розробки

Для розробки проекту було використано IDE IntelliJ IDEA Community Edition (див. рисунок 1.1). IntelliJ IDEA — комерційне інтегроване середовище розробки для Java від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії Community Edition і повнофункціональної комерційної версії Ultimate Edition, для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community-версії поширюються рамках ліцензії Apache 2.0. Бінарні складання підготовлені для Linux, Mac OS X і Windows.

Community версія середовища IntelliJ IDEA підтримує інструменти для проведення тестування TestNG і JUnit, системи контролю версій CVS, Subversion, Mercurial, Git і GitHub, засоби складання Maven і Ant, мови програмування Java, Java ME, Scala, Clojure, Groovy і Dart. До складу входить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, XML-редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань [1].



* + - 1. Процес налагодження програми в IntelliJ IDEA
  1. Конструювання графічного інтерфейсу

Для конструювання графічного інтерфейсу технологія JavaFX використовує формат розмітки FXML — це декларативна мова на основі XML, яка створена корпорацією Oracle для визначення інтерфейсу користувача JavaFX 2.0. Oracle надає спеціальну програму для роботи з FXML — JavaFX Scene Builder 2.0. Приклад роботи програми JavaFX Scene Builder можна побачити на рисунок 1.2.



* + - 1. Конструювання файлу MainWindow.fxml у JavaFX Scene Builder
  1. Використані бібліотеки

В даному проекті використовуються бібліотеки сторонніх розробників. Для завантаження бібліотек було використано Maven Central Repository.

Бібліотеки для роботи з пристроями:

* JavaHIDAPI;
* jSSC;
* usb4java.

Бібліотека для роботи з інтерфейсом користувача:

* ControlsFX.

Бібліотека для роботи с форматом JSON:

* Google Gson.

Бібліотеки, які розширюють стандартні можливості Java та підвищують якість коду в цілому:

* Apache Commons Collections;
* Apache Commons Lang;
* Reflections.
  + 1. Apache Commons Collections

Java Collections Framework був важливим доповненням в JDK 1.2. Він додав, багато потужних структур даних, які прискорюють розробку найбільш значущих Java-додатків. Commons-Collections розвинули класи JDK шляхом надання нових інтерфейсів, реалізацій і утиліт. Надається під Apache License.

В даному проекті використовується клас бібліотеки BidiMap, який представляє собою мапу, ключ якої може буди використаний як значення, а значення як ключ.

* + 1. Apache Commons Lang

Стандартні Java-бібліотеки не в змозі забезпечити достатньо методів для маніпулювання основними класами. Apache Commons Lang надає ці додаткові методи.

Lang надає безліч допоміжних утиліт для java.lang API, зокрема методів маніпуляції String, основні чисельні методи, reflection, concurrency, створення і серіалізация властивостей системи. Крім того, він містить основні вдосконалення java.util.Date. Надається під Apache License.

В даному проекті ця бібліотека використовується для більш зручного визначення ОС, на якій запущена програма (клас org.apache.commons.lang3.  
SystemUtils).

* + 1. ControlsFX

ControlsFX є проект з відкритим кодом для JavaFX, яка покликана забезпечити дійсно високоякісний UI та інші інструменти, які доповнюють JavaFX. Бібліотека має JavaDoc документацію високої якості. Надається під BSD 3-Clause License.

В даному проекті бібліотека використовується як зручний засіб відображення діалогових вікон.

* + 1. Google Gson

Gson це бібліотека для перетворення об’єктів Java у формат JSON. Вона також може бути використана для перетворення рядка JSON до еквівалентного об’єкта Java. Gson може працювати з довільними Java об’єктами, включаючи вже існуючі об’єкти, на які не мають сирцевого коду. Надається під Apache License 2.0.

Переваги Gson:

* забезпечує прості методи toJson і fromJson для перетворення Java об’єктів в JSON і навпаки;
* вже існуючі об’єкти можуть бути перетворені в та з JSON;
* розширена підтримка Java Generics;
* користувацькі подання для об’єктів.

В даному проекті використовується для підтримки функціонування функцій налаштувань програми та зберігання та програвання збережених треків.

* + 1. JavaHIDAPI

Java HID API є JNI, що дозволяє використовувати бібліотеку HIDAPI з Java коду. HIDAPI є мультиплатформною бібліотекою, яка дозволяє додатку взаємодіяти з USB та Bluetooth пристроями HID-класу під Windows, Linux і Mac OS X.

Після створення екземпляра HIDManager можна використовувати деякі з його методів. Метод listDevices повертає список активних в даний момент часу HID. Кожен пристрій представлено екземпляром класу HIDDeviceInfo, який містить інформацію про пристрої. Щоб відкрити пристрій потрібно викликати метод Open.

HIDManager також надає кілька зручних методів для швидкого пошуку і відкриття пристрою або шляху (openByPath) або через vendor id / product id / serial number (openById).

Кожний відкритий пристрій представлено класом HIDDevice. Якщо пристрій відкрито кілька разів, HIDDevice буде однаковий, але безпека потоків не гарантується.

HIDAPI може бути використаний в рамках однієї з трьох ліцензій:

* GNU Public License, версія 3.0;
* BSD-стиль ліцензії;
* оригінал ліцензії HIDAPI.
  + 1. jSSC

jSSC (Java Simple Serial Connector) — бібліотека для роботи з COM портами з Java. jSSC підтримує Win32 (Win98—Win8), Win64, Linux (x86, x86–64, ARM), Solaris (x86, x86–64), Mac OS X 10.5 і вище (x86, x86–64, PPC, PPC64). Надається під GNU Lesser GPL.

В даному проекті ця бібліотека використовується як основний засіб для взаємодії з COM-пристроями (Pololu Wixel, TI ez430-RF2500, Ubiquiti AirView2).

* + 1. Reflections

В інформатиці відбиття або означає процес, під час якого програма може відстежувати і модифікувати власну структуру і поведінку під час виконання. Парадигма програмування, покладена в основу відображення, називається рефлексивним програмуванням. Це один з видів метапрограмування.

У більшості сучасних комп’ютерних архітектур програмні інструкції (код) зберігаються як дані. Різниця між кодом і даними в тому, що виконуючи код, комп’ютери обробляють дані. Тобто інструкції виконуються, а дані обробляються так, як написано цими інструкціями. Однак програми, написані за допомогою деяких мов, здатні обробляти власні інструкції як дані і виконувати, таким чином, рефлексивні модифікації. Такі самомодифікуючі програми в основному створюються за допомогою високорівневих мов програмування, що використовують віртуальні машини (наприклад, Smalltalk, скриптові мови).

Reflections сканує директорію класів, індексує метадані та дозволяє получати доступ до них під час виконання програми. Надається під Other Open Source License.

Використовуючи Reflections ви можете запросити такі метадані:

* підтипи певного типу;
* типи / constructos / методи / поля з анотацією;
* отримати всі погодження ресурсів відповідні регулярному виразу;
* отримати всі методи з конкретною сигнатурою, параметрами та типом повернення.

В даному проекті ця бібліотека використовується для знаходження всіх підкласів класу Device, тим самим генеруючи список всіх підтримуваних профілів пристроїв.

* + 1. usb4java

usb4java це бібліотека Java для доступу до USB-пристроїв. Вона заснована на native libusb 1.0 і використовує Java NIO буфери для обміну даними між libusb і Java. usb4java також підтримує стандарт javax-USB (JSR-80) через розширення usb4java-javax.

Підтримує платформи Linux (x86 32/64 біт, ARM 32 біт), OS X (x86 32/64 біт) і Windows (тільки x86 32/64 біт). Але інші платформи можуть працювати так само добре (якщо є Java 6 і підтримуються libusb) шляхом компіляції бібліотеки JNI вручну. Надається під LGPL.

В даному проекті бібліотека використовується тільки у якості допоміжної.

1. Реалізація Протоколів і інтерфейсу
   1. Загальна схема роботи програми

Загальна схема роботи програми (див. рисунок 2.1) зображена у вигляді діаграми послідовності.

При підключені користувачем пристрою до системи потік класу DeviceConnectionListener знаходить та ідентифікує його. Далі створюється новий потік для роботи з підключеним пристроєм.

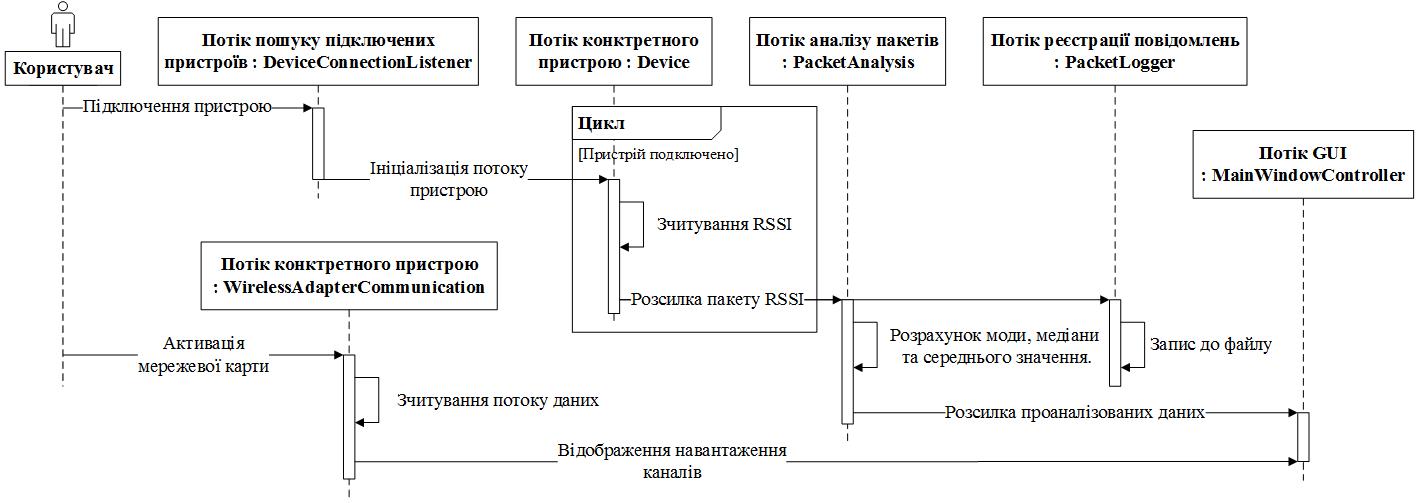
Потік для роботи з пристроєм працює у циклі, в якому відбувається спроба зчитати данні. При успішному зчитуванні генерується подія, яка розповсюджує прийняті пристроєм значення RSSI у виді пакету.

PacketAnalysis аналізує отримані дані та генерує подію, яка перехоплюється системою графічного відображення даних для їх візуалізації.

PacketLogger зберігає пакети для можливості повторного використання.

Користувач також може використати вмонтовану мережеву карту для визначення навантаження конкретного каналу. При цьому активується клас WirelessAdapterCommunication, який зчитує данні з мережевої карти та генерує подію, яка перехоплюється системою графічного відображення даних для їх візуалізації.

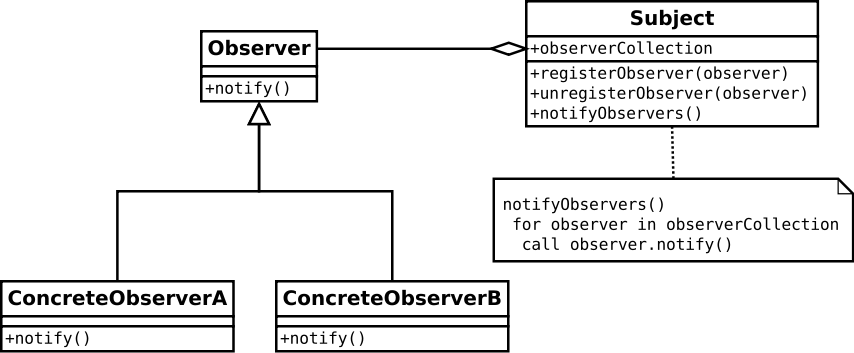
Програма розроблялася модульною, так як з самого початку не було відомо чітких вимог до проекту. Модульність дозволяє зробити прозорішими тексти програми, прискорити написання і тестування, а також дає можливість легко супроводжувати її, так як для внесення змін потрібно працювати з одним модулем не чіпаючи інші [2–4].



* + - 1. Загальна схема роботи програми
  1. Шаблони проектування
     1. Шаблон Observer

Спостерігач, Observer (див. рисунок 2.2) — поведінковий шаблон проектування. Також відомий як «підлеглі» (Dependents), «видавець-передплатник» (Publisher-Subscriber).

Призначення — визначає залежність типу «один до багатьох» між об’єктами таким чином, що при зміні стану одного об’єкта всіх залежних від нього сповіщають про цю подію.



* + - 1. Схема шаблону проектування Observer

При реалізації шаблону Observer зазвичай використовуються наступні класи:

* Observable — інтерфейс, що визначає методи для додавання, видалення та оповіщення спостерігачів;
* Observer — інтерфейс, за допомогою якого спостерігач отримує сповіщення;
* ConcreteObservable — конкретний клас, який реалізує інтерфейс Observable;
* ConcreteObserver — конкретний клас, який реалізує інтерфейс Observer.

Шаблон Observer застосовується в тих випадках, коли система володіє такими властивостями:

* існує, як мінімум, один об’єкт, що розсилає повідомлення;
* є не менше одного одержувача повідомлень, причому їхня кількість і склад можуть змінюватися під час роботи програми.

Цей шаблон часто застосовують в ситуаціях, в яких відправника повідомлень не цікавить, що роблять одержувачі з наданою їм інформацією.

В даній програмі шаблон Observer застосовується для комунікації між модулями програми. Це дозволяє явно відокремити їх один від одного, роблячи їх незалежними.

* + 1. Шаблон Singleton

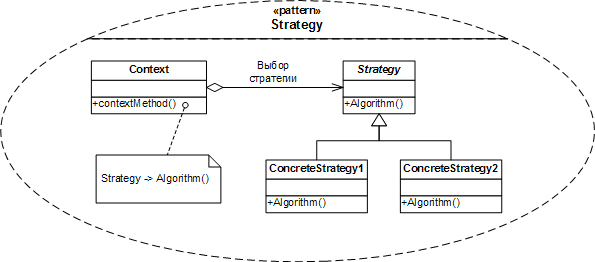
Одинак, Singleton — шаблон проектування, відноситься до класу твірних шаблонів. Гарантує, що клас матиме тільки один екземпляр, і забезпечує глобальну точку доступу до цього екземпляра.

Для деяких класів важливо, щоб існував тільки один екземпляр. Рішення полягає в тому, щоб сам клас контролював свою «унікальність», забороняючи створення нових екземплярів, та сам забезпечував єдину точку доступу. Це є призначенням шаблону Одинак.

В даному проекті шаблон Singleton використовується для наступних класів: допоміжного, реєстрації повідомлень, аналізу пакетів, меню налаштувань, пошуку підключених пристроїв.

* + 1. Шаблон Strategy

Стратегія, Strategy (див. рисунок 2.3) — шаблон проектування, належить до класу шаблонів поведінки. Відомий ще під іншою назвою — "Policy". Його суть полягає у тому, щоб створити декілька схем поведінки для одного об’єкту та винести в окремий клас. Шаблон Strategy дозволяє міняти вибраний алгоритм незалежно від об’єктів-клієнтів, які його використовують.



* + - 1. Схема шаблону проектування Strategy

Завдання шаблону — за типом клієнта (або за типом оброблюваних даних) вибрати відповідний алгоритм, який слід застосувати.

Учасники:

* Strategy — визначає, як будуть використовуватися різні алгоритми;
* ConcreteStrategy — реалізують ці різні алгоритми;
* Context — використовує конкретні класи ConcreteStrategy за допомогою посилання на конкретний тип абстрактного класу Strategy.

В даному проекті шаблон Strategy використовується для вибору конкретного пристрою зі стеку доступних, виходячи з його параметрів [5].

# Висновок

В роботі представлені результати розробки універсального засобу для роботи з аналізаторами спектру бюджетного сегменту від різних виробників. Проаналізовані протоколи обміну між пристроями і програмним забезпеченням, яке з ними поставляється, вибрані найкращі ідеї для реалізації відображення результатів сканування. Вбудована можливість «гарячого» приєднання/від’єднання пристроїв, підключення кількох однакових пристроїв, робота з мережевою карткою.

Дані від різних пристроїв зберігаються в універсальній структурі даних, для яких існує можливість повторного «програвання» і аналізу результатів.

В якості середовища розробки вибране Intellij IDEA і JavaFX Scene Builder, для кроспратформеності додатка використовується мова програмування Java, але в результаті не вийшло на ОС Windows задіяти деякі пристрої (через закритий сирцевий код драйверів) і мережеву картку.

Процес розробки програмного забезпечення відповідає основним принципам шаблону проектування Scrum.

# Список літератури

1. Макконнелл, С. Совершенный код. Мастер-класс // Стив Макконнелл / Пер. с англ. — М.: Русская редакция, 2010. — 896 с.
2. Bloch, J. Effective Java // Joshua Bloch / 2nd edition. — Sun Microsystems, 2008. — 369 p.
3. Мартин, Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста // Роберт Мартин / Пер. с англ. — СПб: Питер, 2010. — 464 с.
4. Сьерра, К. Изучаем Java // Кэти Сьерра, Берт Бейтс / Изд. 2-е / Пер. с англ. — Эксмо, 2005. — 708 p.
5. Гамма, Э. Приемы объектно-ориентированного проектирования: Паттерны проектирования // Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. — СПб.: Питер, 2010. — 368 с.