ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky

Mobilní aplikace pro správu UHF RFID čtečky

leden 2016

Diplomant: Bc. Tomáš Zitta
Vedoucí práce: Ing. Marek Neruda Ph.D.

Čestné prohlášení

Prohlašuji, že jsem zadanou diplomovou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé diplomové práce nebo její části se souhlasem katedry.

Datum: 11. 1. 2015

……..……

podpis diplomanta

Poděkování

Chtěl poděkovat své rodině a vedoucímu práce za poskytnutou podporu během tvorby této práce.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra telekomunikační techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Zitta Tomáš**

Studijní program: Komunikace, multimédia a elektronika
Obor: Sítě elektronických komunikací

Název tématu: **Mobilní aplikace pro správu UHF RFID čtečky**

Pokyny pro vypracování:

Navrhněte a zrealizujte mobilní aplikaci pro správu UHF RFID čtečky pro OS Android. Uveďte možnosti přístupu k UHF RFID čtečce pomocí externího zařízení. Proveďte rešerši funkcí, které aplikace pro správu UHF RFID čteček dnes umožňují a tyto základní funkce implementujte do mobilní aplikace. Zvažte další funkce pro správu UHF RFID čtečky.

Seznam odborné literatury:

[1] Programming Android: Java Programming for the New Generation of Mobile Devices. Sebastopol: O'Reilly Media, 2011. ISBN 978-1-4493-8969-7.
[2] UJBÁNYAI, Miroslav. Programujeme pro Android. Praha: Grada Publishing a.s., 2011. ISBN 978-80-247-3995-3.
[3] Android developers. Android [online]. 2014 [cit. 2014-11-13]. Dostupné z: http://developer.android.com/index.html

Vedoucí: Ing. Marek Neruda, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

prof. Ing. Boris Šimák, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 13. 11. 2014

Anotace

Tato diplomová práce se zabývá návrhem a realizací aplikace pro správu UHF (Ultra High Frequency) RFID (Radio Frequency Identification) čtečky. V aplikaci je použit protokol LLRP (Low Level Reader Protocol). První část práce se zabývá úvodem do tématu. Druhá část práce se zabývá technologií RFID a LLRP. Třetí část se zabývá operačním systémem Android. Čtvrtá část se zabývá rešerší funkcí aplikací UHF RFID čteček. Poslední část práce se zabývá návrhem, realizací a testováním aplikace.

Summary

This master thesis deals with the design and implementation of application for management of UHF (Ultra High Frequency) RFID (Radio Frequency Identification) reader. The application uses LLRP (Low Level Reader Protocol). The first part of the work deals with introduction to the topic. The second part deals with the RFID technology and LLRP. The third part deals with the Android operating system. The fourth part deals with search functions of applications of UHF RFID readers .The last part of the work focuses on the design, implementation and testing of the application.

Obsah

1 Úvod

Cílem této práce je navrhnout a vytvořit mobilní aplikaci pro správu UHF RFID čtečky pro OS Android.

Hlavní výhoda tohoto technického řešení spočívá v mobilitě, kdy správu UHF RFID čtečky lze provádět ze snadno přenosných zařízení, jako je mobilní telefon nebo tablet. Tato zařízení nejsou připojena kabelem do počítačové sítě a je využit bezdrátový přenos dat.

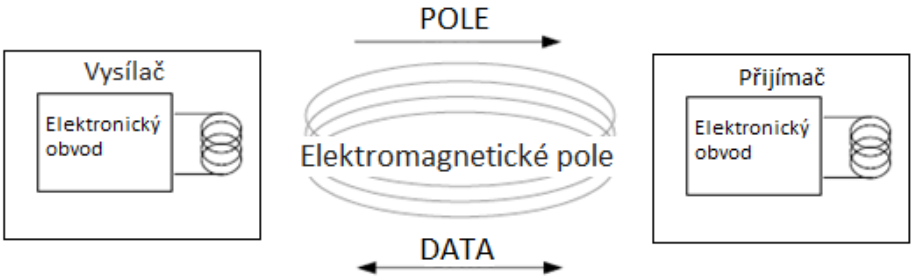
V úvodních kapitolách je probrána problematika technologie RFID a operačního systému Android.

V praktické části je rozebírána rešerše funkcí UHF RFID čteček, návrh a realizace middleware a testování aplikace.

V závěru je provedeno zhodnocení dosažených výsledků a možnost budoucího vývoje middleware.

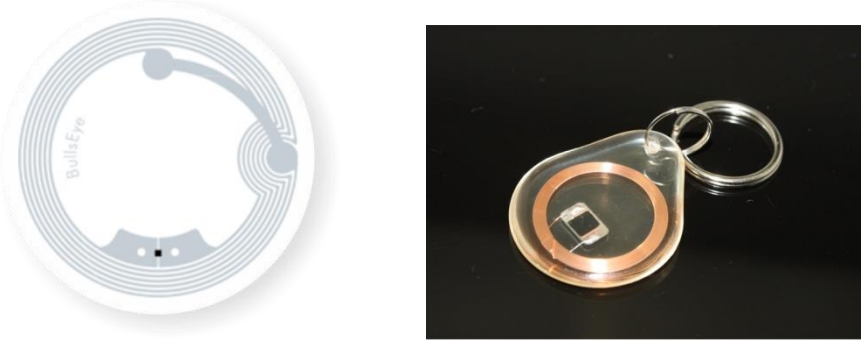
2 RFID

RFID (Radio Frequency Identification) je technologie pro bezdrátový a bezkontaktní přenos informací. RFID pracuje ve 3 pásmech a to LF (30 až 300 kHz), HF (3 až 30 MHz) a UHF (300 MHz až 3 GHz). RFID slouží k zaznamenání, uchování a poskytování informací o objektech v reálném čase. Datový řetězec tvoří čtečka a transpondér. Princip je založen na elektromagnetické indukci, kdy elektromagnetické pole dvou antén je použito pro přenos energie a tím informací mezi vysílačem a přijímačem.[62] RFID existuje v mnoha standardech. Při párování se nejprve přenesou informace o použité přenosové technologii, poté dotazovací zařízení zahájí komunikaci prostřednictvím dohodnutého standardu [55].



Obr. 2.1 Princip funkce [51]

Tagům se dříve říkalo transpondéry, ale v poslední době se rozšířilo toto označení. Proto je použito v práci označení tag. RFID tagy jsou pasivní, aktivní, pasivní nebo semipasivní obvody, které obsahují data a mohou komunikovat s aktivními RFID čtečkami [62]. Tagy jsou složeny z antény, ke které je připájen integrovaný obvod s pamětí. Anténa s integrovaným obvodem může být nalepena na papír, či zalita v plastu nebo ve skle. Velikost paměti se pohybuje od 1 B do 128 kB v závislosti na standardu. Tagy nejčastěji slouží jen pro čtení, kdy data jsou nahrána už při výrobě, ale tagy mohou být i přepisovatelné a uzamknutelné. Používají se v oblastech, kde je potřeba přenést jen menší množství dat. Dále se používají hybridní tagy, které kromě aktivního integrovaného RFID obvodu samotného obsahují další senzory a případně GPS.



Obr. 2.2 Provedení RFID tagů [52], [53]

2.1 Typy LF, HF a UHF

Pásmo LF (30 až 300 kHz) umožňuje přenos informací na vzdálenost okolo 10cm při nízkých přenosových rychlostech. Systémy z tohoto pásma pracují na kmitočtech 125 khz a 134 kHz. Tyto systémy slouží převážně k identifikaci. Výhoda tohoto pásma je odolnost vůči rušení.

Pásmo HF (3 až 30 MHz) umožňuje přenos informací na vzdálenost okolo 10 až 1 m (v případě NFC jsou to jednotky cm). Nejpoužívanější kmitočet tohoto pásma je 13,56 MHz, na tomto kmitočtu pracuje i technologie NFC, která je s některými standardy RFID kompatibilní. Nejběžnější použití těchto systémů je k platbám a věrnostním programům.

Pásmo UHF (300 MHz až 3 GHz) umožňuje přenos informací až na 12 m pro pasivní tagy. Toto pásmo dosahuje nejvyšších přenosových rychlostí, ale je nejcitlivější k rušení. Toto pásmo umožňuje výrobu levnějších tagů, než v případě zbylých dvou pásem a proto je využíváno v inventarizačních systémech.

2.3 Rozhraní LLRP

Rozhraní LLRP je určeno k výměně informací mezí čtečkou a klientem. Pracuje na aplikační vrstvě RM/OSI modelu.

2.3.1 Knihovna LLRP toolkit

. K impementaci protokolu LLRP existuje knihovna LLRP toolkit.

3 Operační systém Android

3.1 Úvod

Android je operační systém pro mobilní zařízení, je vyvíjen hlavně společností Google a OHA (Open Handset Alliance) [2]. Jako základ systému je použito jádro Linux. Verze Linux kernelu se liší podle verze Androidu a zařízení. Android ovšem na rozdíl od klasických GNU/Linux distribucí postrádá doplňky GNU (GNU's Not Unix) [1]. Android je open source, což znamená, že má otevřený zdrojový kód, který je k dispozici pod licencí Apache 2.0 [3] a při dodržení podmínek je k dispozici zdarma [4]. Tento zdrojový kód lze zkompilovat a vytvořit si firmware pro vlastní zařízení. Zdrojový kód lze upravovat a přispět do vývoje Androidu. Vývojáři mohou umisťovat hotové aplikace na server Google Play [5], kde mohou být nabízené ke stažení zdarma nebo za poplatek.

3.2 Historie

Operační systém Android získala společnost Google akvizicí firmy Android Inc. v srpnu v roce 2005. Společnost Android Inc. byla založena v roce 2003 a její zakladatelé byly Andy Rubin, Rich Miner, Nick Sears a Chris White. Android byl představen 5. listopadu 2007 při vzniku sdružení OHA, o týden později bylo vydáno tzv. early-look SDK (Software Development Kit) [1]. OHA je sdružení 84 firem (v době vzniku jich bylo více než 30) s jediným cílem, a to vyvíjet otevřené standardy pro mobilní zařízení. Ve sdružení jsou softwarové firmy (např. Ebay a Google), mobilní operátoři (např. China Mobile a T-Mobile), výrobci mobilních telefonů (např. HTC, Sony a Samsung), výrobci a dodavatelé polovodičových součástek (např. Intel, Qualcomm a Nvidia), reklamní společnosti a další [2]. Finální SDK 1.0 bylo vydáno 23. září 2008. Mezi veřejnost se Android dostal 22. října 2008, kdy byl uveden ve Spojených Státech Amerických s telefonem T-Mobile G1 známým také pod názvem HTC Dream, v ten den také došlo ke zveřejnění zdrojových kódů systému [6]. Android byl uveden ve verzi 1.0.

Od té doby Android prošel mnoha úpravami a stal se z něho nejrozšířenější operační systém pro mobilní zařízení na světě. Jeho tržní podíl ve druhém čtvrtletí roku 2015 činil 82,8% [9].

V současné době je Android ve verzi 6.0.1 Marshmallow. Android ve verzi 4.0 Ice Cream Sandwich a novější verze jsou natolik odlišné od verze 1.0, že se skoro jedná o dva odlišné operační systémy. Verze OS Android 5.0 kompletně změnila uživatelské rozhraní oproti verzi 4.0. Jednotlivé verze jsou popsány v tab. 3.1. Změny uživatelského rozhraní jsou zachyceny na obr. 3.1. Android je užíván v chytrých telefonech, tabletech, netboocích, ale už i v digitálních fotoaparátech, pro které byl zpočátku vyvíjen. Dále jsou vyvíjeny verze zaměřené na nositelnou elektroniku, televize a automobily.

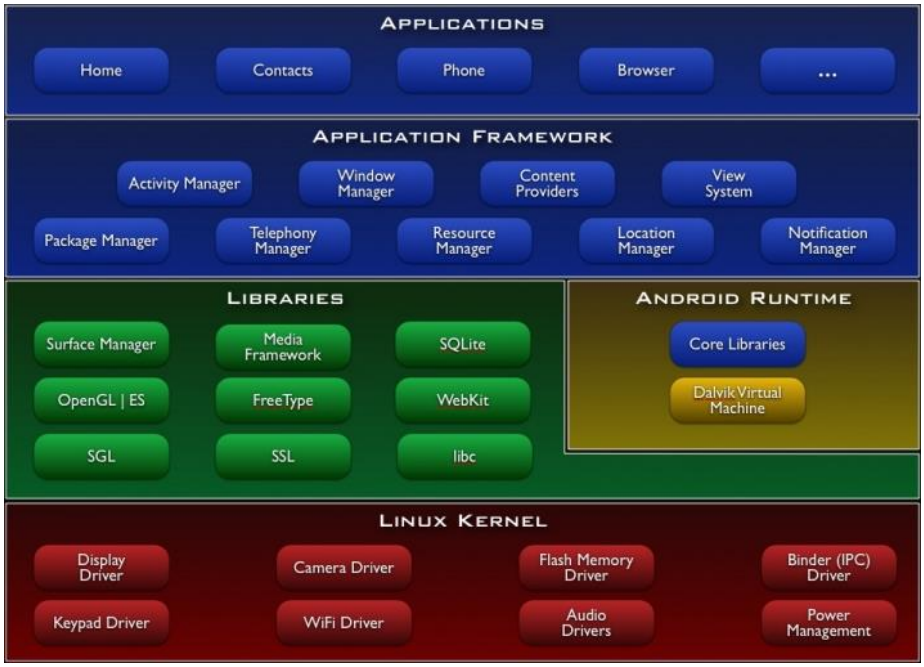
Tabulka 3.1 Seznam a popis verzí Androidu [1], [10], [13], [13], [18], [19], [20]

Verze	Slovní označení	Datum vydání	Klíčové změny
1.0	Base	23. září 2008	Vydání pro veřejnost
1.1	Base 1.1	9. únor 2009	Minoritní update
1.5	Cupcake	30. duben 2009	Podpora akcelerometru, implementace softwarové klávesnice, podpora Bluetooth profilu pro přenos bezdrátové hudby ve stereo kvalitě
1.6	Donut	15. září 2009	Podpora rozlišení displeje 800x480 pixelů, podpora VPN (Virtual Private Network)
2.0 2.1	Eclair	26. říjen 2009	Změna uživatelského rozhraní, podpora Microsoft Exchange, Bluetooth 2.1, více možností rozlišení displeje
2.2	Froyo	20. květen 2010	Možnost instalace Aplikací na paměťovou kartu a vytvoření Wi-Fi hotspotu, podpora Adobe Flash 10.1 a OpenGL ES 2.0
2.3 2.4	Ginger-bread	6. prosinec 2010	Vylepšení UI, podpora NFC, protokolu SIP a kodeku WebM a HTML5
3.0 3.1 3.2	Honey-comb	22. únor 2011	Tabletové UI, Není pro telefony, není potřeba fyzických tlačítek, funkce USB (Universal Serial Bus) Host
4.0.x	Ice Cream Sandwich	19. říjen 2011	Nové UI sjednocené pro telefony a tablety podpora Wi-Fi direct a Android Beam pro přenos obsahu mezi zařízeními
4.1.x	Jelly Bean	27. červen 2012	Zlepšena plynulost vykreslování, vylepšené notifikace
4.2.x	Jelly Bean MR1	29. říjen 2012	Podpora více uživatelských účtů, nová notifikační lišta
4.3.x	Jelly Bean MR2	24. červenec 2013	Podpora BT (Bluetooth) Low Energy, profilů s omezeným oprávněním, DRM (Digital Rights Management) pro média, OpenGL ES 3.0
4.4.x	Kit Kat	31. říjen 2013	Podpora emulace NFC karet pro platby, skupinové snímáním senzorů, detekce kroků a krokoměr, fullscreen režim, Chromium WebView
5.0.x	Lollipop	3. listopad 2014	Nové uživatelské rozhraní pojmenované „Material Design“, snížení spotřeby el. Energie - Project Volta, nové běhové prostředí ART, podpora 64 bitových architektur, OpenGL ES 3.1, H.265, nových senzorů
5.1.x	Lollipop MR1	10. březen 2015	Podpora více SIM karet rozšířená podpora služeb operátorů
6.0.x	Marshmal low	5. říjen 2015	Podpora uživatelského udílení oprávnění aplikacím, podpora senzorů otisků prstů, záloha nastavení aplikací, lepší správa napájení

Obr 3.1 Srovnání základních obrazovek Androidu 1.0 [7] a 4.2 [8] a 5.0 a 6.0

3.3 Architektura OS

Architektura Androidu je rozdělena do pěti vrstev [13]. Jádro systému je tvořeno OS Linux, nad ním jsou nativní knihovny a vedle nich pracuje běhové prostředí. Do verze 5.0 se používalo „Runtime“ prostředí DVM (Dalvik Virtual Machine), od verze 5.0 se používá běhové prostředí ART (Android Runtime). Ve třetí vrstvě se nachází „Application Framework“ a až nad ní jsou samotné aplikace [1].



Obrázek 3.2: Architektura Androidu [16]

3.3.1 Linux Kernel

Jádro Linux je použito pro správu paměti, procesů, způsob zabezpečení a práci se sítí [1]. Ovladače se nachází v jádře, které je modulární [14]. Jedním z důvodů, proč bylo jako jádro zvolen Linux, je jeho svobodný kód. Některé části jádra se chovají odlišně, než kdyby běžely na klasickém PC. Správa paměti nepoužívá SWAP a při nedostatku paměti jsou aplikace na pozadí ukončovány. Procesy také neběží úplně odděleně, protože jejich paměť je částečně sdílena pomocí komponenty „Shared Memory Driver“. Sdílení je

prováděno z důvodů malé operační paměti zařízení a sdílí se např. knihovny a jednotlivé komponenty Manager. Správa řízení napájení musí uvažovat omezený zdroj energie. Nad jádrem je vrstva HAL (Hardware Abstraction Layer), která abstrahuje hardware pro aplikace [6].

3.3.2 Knihovny

V Tab. 3.2 jsou uvedeny nativní knihovny, které jsou napsány v jazycích C a C++. Aplikace ke knihovnám mají přístup přes Aplication Framework. Některé knihovny jako např. libc jsou podobné knihovnám z GNU / Linux operačních systémů, ale jsou ořezané o funkce a komponenty, které nejsou potřeba na mobilní platformě a proto byly přepsány od začátku [6].

Tabulka 3.2 Seznam základních knihoven [1], [6], [16]

FreeType	renderování fontů
Libc	knihovna jazyka C
LibWebCore	engine pro prohlížeč internetu
Media	podpora pro grafické formáty a audio a video kodeky
OpenGL ES	3D akcelerace grafiky
SGL	2D grafický engine
SQLite	databázový engine s sql rozhraním

3.3.3 Android Runtime

Android Runtime obsahuje komponentu ART (Android Runtime) od verze Android 5.0 nebo DVM se základními knihovnami pro starší verze Androidu [13]. Aplikace pro platformu Android jsou psány v jazyce Java, ale Java SE (Standard Edition) není plně pokryta. Knihovny uživatelského rozhraní klasických počítačů (Swing a Abstract Window Toolkit) jsou nahrazeny knihovnami rozhraní Androidu a jsou přidány knihovny HTTP (Hypertext Transfer Protocol) Apache, knihovny pro práci s Bluetooth a Wi-Fi a další [1], [6]. Pro běh aplikací je potřeba virtuální stroj.

Standardní JVM (Java Virtual Machine) není vhodný, protože mobilní zařízení obsahují nevýkonný procesor, malou operační paměť a akumulátory s omezenou kapacitou. Dále byl požadavek na běh více instancí VM najednou na rozdíl od PC, kde většinou běží jedno JVM a v něm běží více aplikací a dalším důvodem bylo, že JVM není open source. Tyto důvody vedly k vývoji vlastnímu VM [6]. DVM a jeho knihovny jsou založené na open source java knihovnách Apache Harmony. Dalvik vyvinul Dan Bornstein ze společnosti Google. Tato komponenta je pojmenovaná po městě Dalvík na Islandu, protože odtamtud pocházeli předci autora [6]. Dalvik má vlastní formát kompilovaných tříd DEX (Dalvik EXecutable) z důvodů šetření paměti. Dalvik používá JIT (Just In Time) od verze Androidu 2.2[13]. Před touto verzí Dalvik kompilaci neprováděl. Oproti standardnímu JVM Dalvik používá více nativního kódu[6]. Aplikace jsou kompilovány při běhu a zabírají více místa v operační paměti a déle se spouští, než kdyby byly zkompilované při instalaci.

Problém s rychlostí běhu řeší nové běhové prostředí ART. ART nahradilo prostředí Dalvik od verze Androidu 5.0. ART je oproti Dalviku typu AoT (Ahead of Time). To znamená, že aplikace jsou při instalaci zkompilovány z bajt kódu virtuálního stroje DEX do nativního kódu OAT procesoru telefonu pomocí nástroje dex2oat. Tyto soubory OAT jsou mapovány k paměti a tím umožňují snadnější práci s více procesy současně (multitasking) a správu operační paměti a není potřeba vyrovnávací paměť „cache“ pro JIT kompilaci. A tedy aplikace by měly být rychlejší a spotřebovávat méně systémových prostředků, takže telefony by měly mít delší výdrž na jedno nabití baterií. Nevýhoda zkompilovaných aplikací je, že zabírají v zařízení více místa, at to až o 10 až 20%. ART podporuje kromě architektury ARM také architektury x86, MIPS a jejich 64 bitové varianty. Běhové prostředí ART je možné použít díky nárůstu výkonu zařízení [11 - 12].

3.3.4 Aplikační rámec

Aplikační rámec „Application Framework“ pomáhá vývojářům vytvářet aplikace. Poskytuje knihovny a komponenty „Manager“ pro usnadnění tvorby aplikace. Komponenty „Manager“ poskytují např. telefonní služby, lokační služby a řízení komponent uživatelského rozhraní. Aplikacím také zpřístupňuje nativní knihovny [17].

Tabulka 3.3 Seznam základních komponent aplikačního rámce [1], [13], [15]

View System	tlačítka, textová pole, seznamy, atd.
Resource Manager	zajišťuje zdroje mimo kód aplikace, tzn. texty, layouty, obrázky
Activity Manager	řídí životní cyklus aktivit
Notification Manager	notifikace ve stavové liště

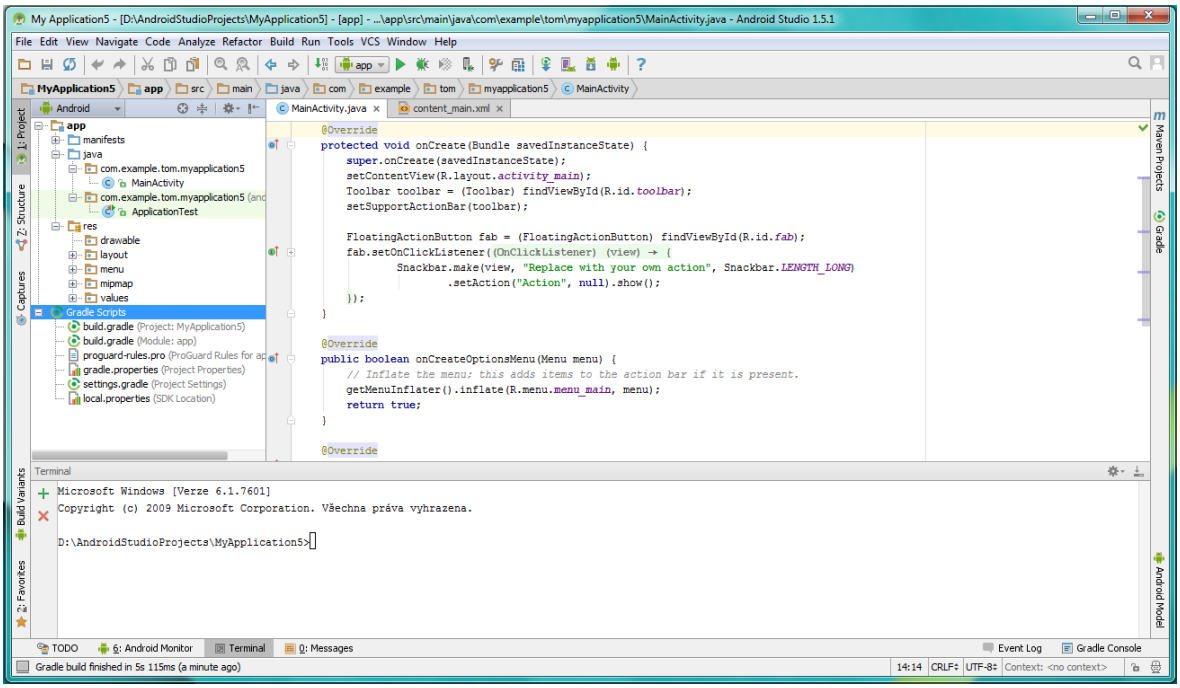
3.3.5 Aplikace

Android je dodáván se sadou předinstalovaných aplikací od správce kontaktů až po Google mapy. Aplikace si jsou rovnocenné, takže lze nahradit jakoukoliv předinstalovanou aplikaci jinou aplikací [17].

3.4 Popis vývojového rozhraní

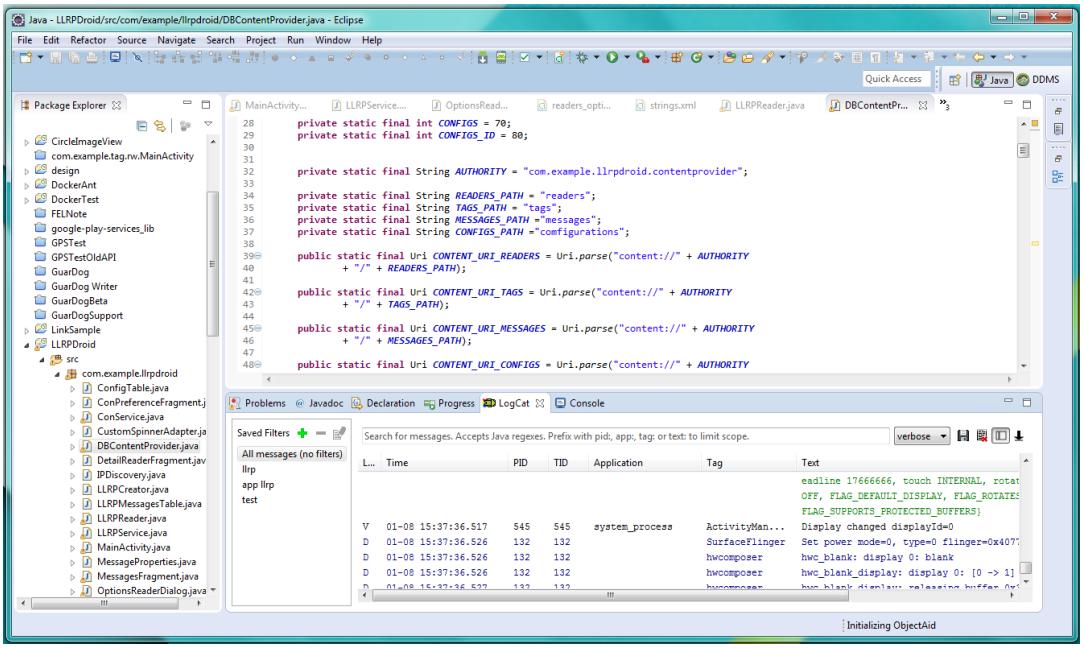
Na platformu Android se aplikace vytváří prostřednictvím křížového vývoje, z toho plyne, že samotný vývoj probíhá na jiné platformě. Vývoj může probíhat na operačních systémech Windows, GNU/Linux distribucích a OS X [32]. Nejdřív je nutné stáhnout a nainstalovat JDK (Java Development Kit).

Pro vývoj je oficiálně podporováno IDE (Integrated Development Environment) Android Studio, které jako základ používá IDE IntelliJ IDEA. Android Studio vylepšuje především tvorbu Layouts. Při jejich tvorbě se zobrazuje ukázka vzhledu v reálném čase. Dále byla přidána funkce ukázkového zobrazení na obrazovkách různých velikostí v jednom okně. Android Studio zpřehledňuje práci s řetězci, kdy jsou prezentovány ve zdrojovém kódu jako přímo napsané, ale při najetí na ně kurzorem se zobrazí cesta k jeho zdroji. Celkově byla rozšířena nápověda pro dokončování kódu [46].



Obrázek 3.3: Vývojové prostředí Android Studio

Aplikace lze ještě vyvíjet i na starším vývojovém prostředí Eclipse. Avšak doplněk pro vývoj pro Android ADT (Android Development Tools) již není podporován [30].



Obr. 3.4 Vývojové prostředí Eclipse

Prostřednictvím programu „SDK Manager“ je možné stáhnout jednotlivé SDK balíčky k starším verzím Androidu a další knihovny. Pomocí AVD (Android Virtual Device) doplňku lze vytvářet virtuální zařízení. Pro některé aplikace není nutné při vývoji použít skutečné zařízení [1]. K využití skutečných zařízení je potřeba pro ně stáhnout příslušné ADB (Android Debug Bridge) ovladače. Vývojové prostředí komunikuje se skutečným a virtuálním zařízením prostřednictvím nástroje ADB využívající architektury klient-server [31].

Praktická část

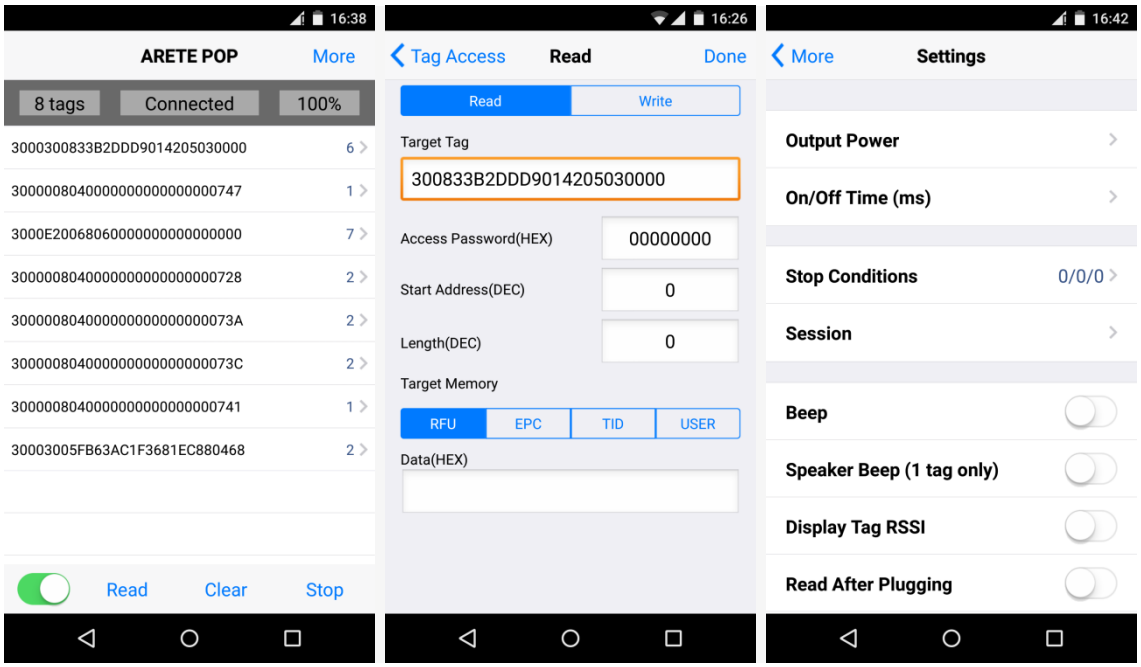
4 Rešerše funkcí aplikací UHF RFID čteček

4.1 ARETE POP

RFID čtečka s označením ARETE POP [41] je specifická tím, že se připojuje k zařízením s OS Android 2.3.3 a vyšším nebo iOS 6.0 a vyšším, prostřednictvím konektoru JACK 3,5 mm. Dále obsahuje baterii o kapacitě 360 mAh a kruhově polarizovanou anténu. Pracovní kmitočet se pohybuje od 865 do 926 MHz dle geografického určení. Maximální výstupní výkon čtečky je 25 dBm [41]. Cena čtečky se pohybuje okolo 275 £ bez daně (cca 10 000 Kč) [42].

Pro zprovoznění čtečky je nutné stáhnout aplikaci ARETE POP do mobilního zařízení z Google play, případně z App Store.

Aplikace čtečky je kompatibilní s protokolem EPCglobal Class 1 Gen 2 [41]. Čtečka umožňuje čtení a změnu EPC (Electronic Product Code) tagu, použití příkazů LOCK a KILL pro zamknutí a znefunkčnění tagu. Čtečka umožňuje zobrazit a změnit obsah uživatelské pamětí a RFU (Reserved for Future Use) paměti, paměť TID (Tag ID) lze obvykle jen zobrazit. Další podporované funkce jsou export dat, zobrazení počtu načtení jednotlivých ID tagu, nastavení výstupního výkonu čtečky a zobrazení RSSI (Received Signal Strength Indication) tagu [41].



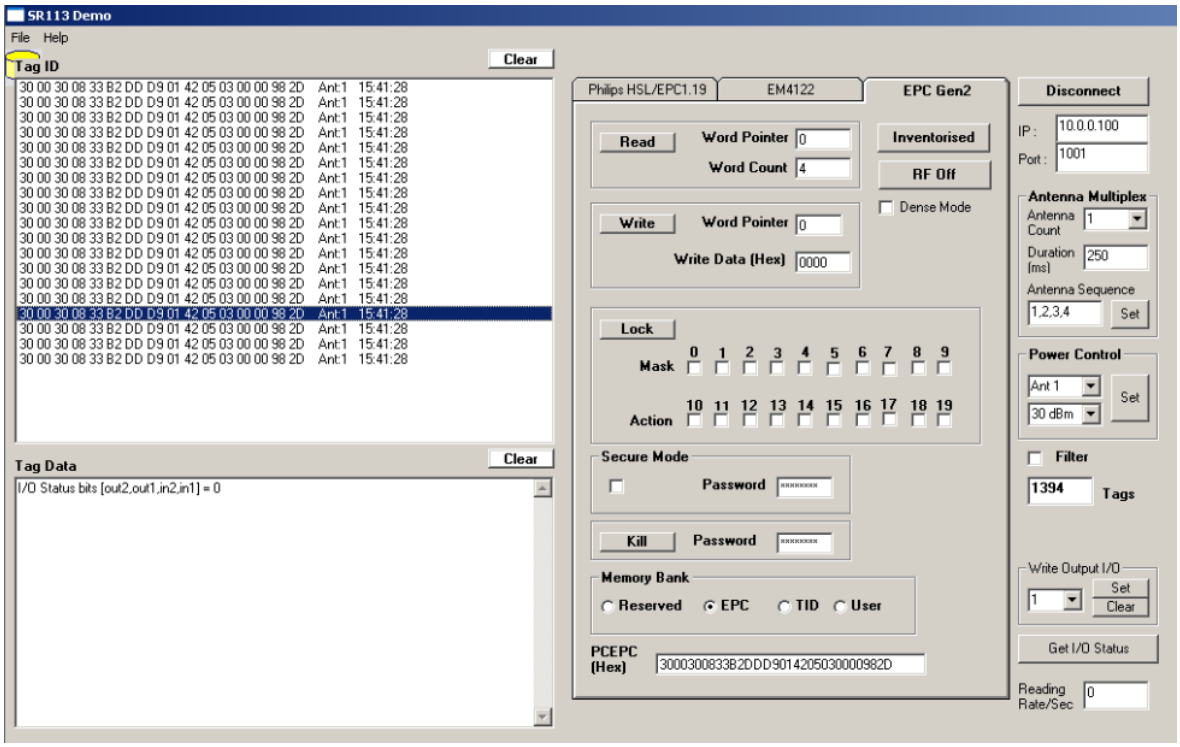
Obrázek 4.1: Uživatelské prostředí aplikace pro ovládání čtečky [45].

4.2 Elatec SR113

RFID čtečka s označením Elatec SR113 je vybavena rozhraními Ethernet, RS232, RS485 a I/O porty. Dále umožňuje připojit až 4 antény prostřednictvím 4 SMA konektorů. Pracovní kmitočet se pohybuje od 860 do 960 MHz dle geografického určení. Maximální výstupní výkon čtečky je 30 dBm. Čtečka je kompatibilní s protokoly ISO18000-6B, EPCglobal Class 1 Gen 2 a EM Microelectronics-Marin SA [43]. Čtečka se v současné době již neprodává.

Pro připojení ke čtečce je potřeba nainstalovat na OS Windows aplikaci SR113 Demo. Dále je nutné čtečku připojit přímo síťovým kabelem k počítači a nastavit staticky adresu sítě 10.0.0.0 s maskou 255.255.255.0. Připojení v aplikaci SR113 Demo je realizováno tlačítkem „conect“. Inventarizace tagů se provádí stisknutím tlačítka „inventorised“ s následným stisknutím tlačítka RF On. Tlačítkem STOP se zastaví čtení. Čtečka po nakonfigurování v aplikaci SR113 podporuje LLRP (Low Level Reader Protocol) 1.0.1.

Schopnosti aplikace čtečky při práci s tagy se liší dle použitého čipu tagu. Pro tagy Philips HSL/EPC1.19 umožňuje čtečka čtení a zápis dat tagu. Pro tagy EM4122 umožňuje pouze čtení. Pro tagy EPC Gen2 umožňuje použití příkazů LOCK a KILL a čtení a změnu EPC tagu. Čtečka umožňuje zobrazit a změnit obsah uživatelské pamětí a RFU paměti, paměť TID lze obvykle jen zobrazit. Dále umožňuje filtraci tagů, zobrazení ID portu antény, na které byl tag načten, nastavení výstupního výkonu čtečky celkově i pro jednotlivé antény a vypnutí jednotlivých portů antén [44].



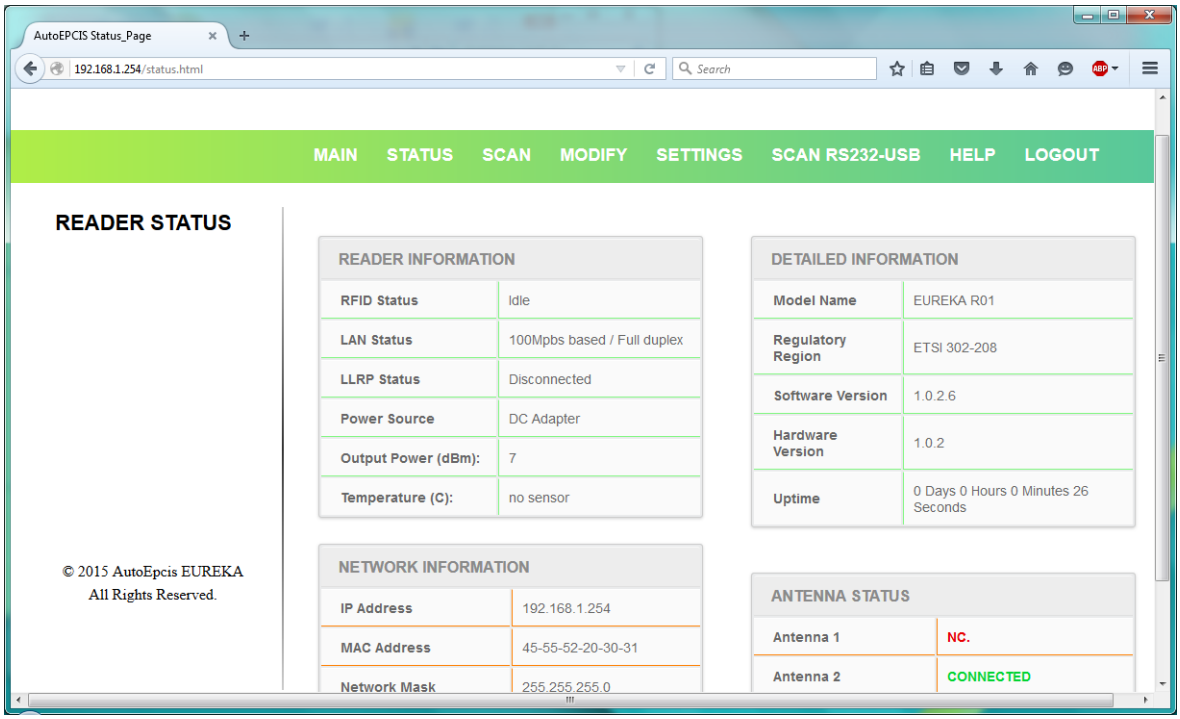
Obrázek 4.2: Uživatelské prostředí aplikace pro ovládání čtečky.

4.3 Proprietární UHF RFID čtečka Eureka

Tato čtečka vznikla v rámci projektu Eureka na Katedře telekomunikační techniky ČVUT v Praze, FEL. Disponuje rozhraním Ethernet a umožňuje připojit až 4 antény prostřednictvím SMA konektorů [54]. Čtečka má webové rozhraní a podporuje protokol LLRP 1.0.1. Maximální výstupní výkon čtečky je 29 dBm. Čtečka je určena pro UHF pásmo 868 MHz a je kompatibilní s protokolem EPCglobal Class 1 Gen 2.

Pro zprovoznění čtečky je třeba ji připojit do počítačové sítě a dále se pomocí přidělené IP adresy buď přihlásit do webového rozhraní, nebo se pomocí middleware připojit prostřednictvím protokolu LLRP, a poté lze provést konfiguraci čtečky.

Webové rozhraní čtečky umožňuje čtení a změnu EPC tagu. Další podporované funkce jsou zobrazení počtu načtení jednotlivých ID tagu, manuální konfigurace sítě, nastavení výstupního výkonu a citlivosti čtečky. Webové rozhraní u tagů zobrazuje RSSI (Received Signal Strength Indication).



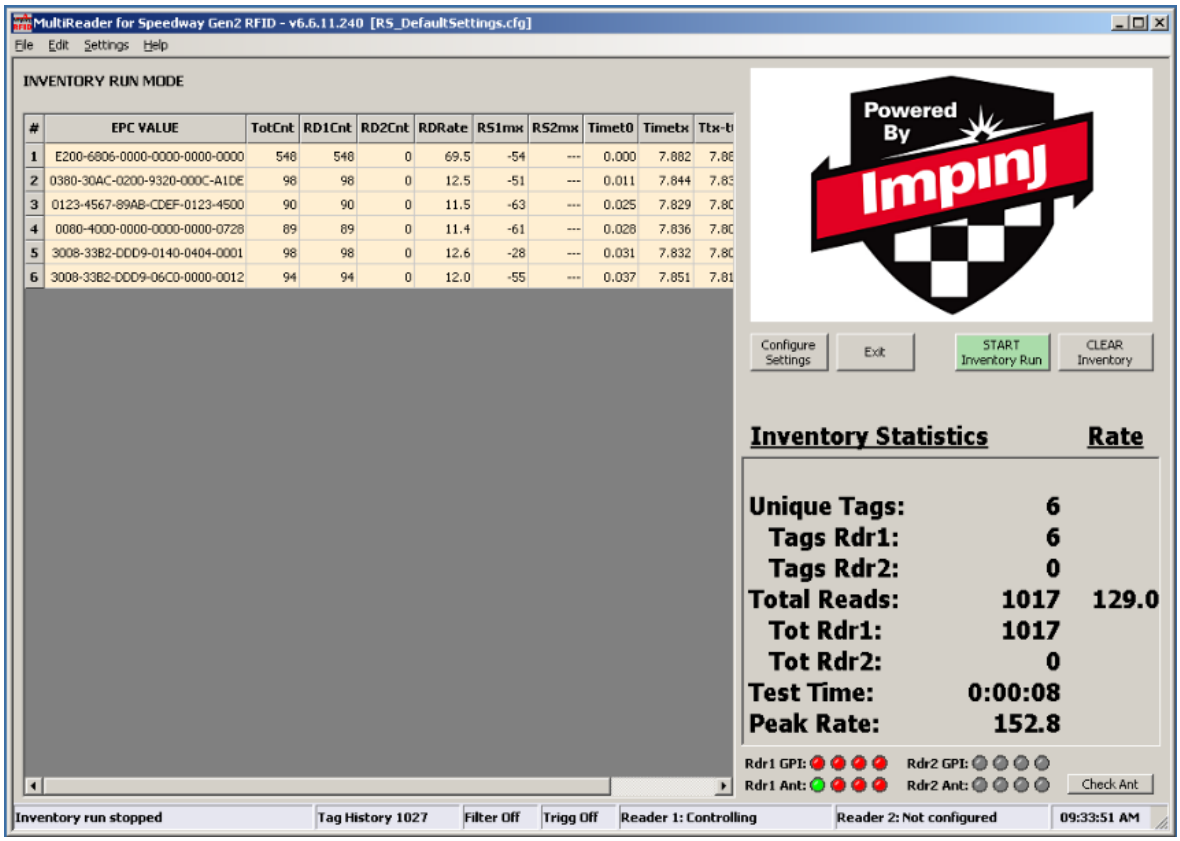
Obrázek 4.3: Uživatelské prostředí aplikace pro ovládání čtečky.

4.4 Impinj SPEEDWAY R420

RFID čtečka společnosti Impinj s označením SPEEDWAY R420 je vybavena rozhraními Ethernet s podporou PoE (Power over Ethernet), Console, USB v režimu Host, USB v režimu Device, GPIO s RS232. Dále umožňuje připojit až 4 antény prostřednictvím 4 SMA konektorů. Pracovní kmitočet se pohybuje od 865 do 956 MHz dle geografického určení [46]. Maximální výstupní výkon čtečky je 32,5 dBm. Čtečka podporuje LLRP 1.0.1 a je kompatibilní s protokolem EPCglobal Class 1 Gen 2 [47]. Cena čtečky je 1585 $ [48].

Pro připojení ke čtečce je vyžadována aplikace Impinj Multireader pro OS Windows. Poté se čtečka připojí do počítačové sítě přes rozhraní Ethernet. Pro připojení je třeba zadat IP adresu a nastavit zeměpisnou oblast kmitočtové regulace [46].

Aplikace čtečky umožňuje čtení a změnu EPC tagu a umožňuje použití příkazů LOCK a KILL. Čtečka umožňuje zobrazit a změnit obsah uživatelské pamětí a RFU paměti, paměť TID lze obvykle jen zobrazit. Dále umožňuje nastavení citlivosti a výstupního výkonu čtečky i zvlášť pro jednotlivé antény. Porty jednotlivých antén lze vypnout a zapnout. V seznamu načtených EPC tagů se zobrazuje čas mezi prvním a posledním načtením tagu, počet načtení jednotlivých EPC tagů, RSSI, čas prvního a posledního načtení tagu. V aplikaci se zobrazuje počet načtených tagů za sekundu a celkový počet načtení EPC tagů podle čteček. Dále lze provést nastavení pracovní frekvence a filtraci tagů, dle EPC. IP adresu čtečky lze zadat manuálně nebo lze automaticky vyhledat dostupné čtečky v síti a zvolit čtečku ze seznamu nalezených [46].



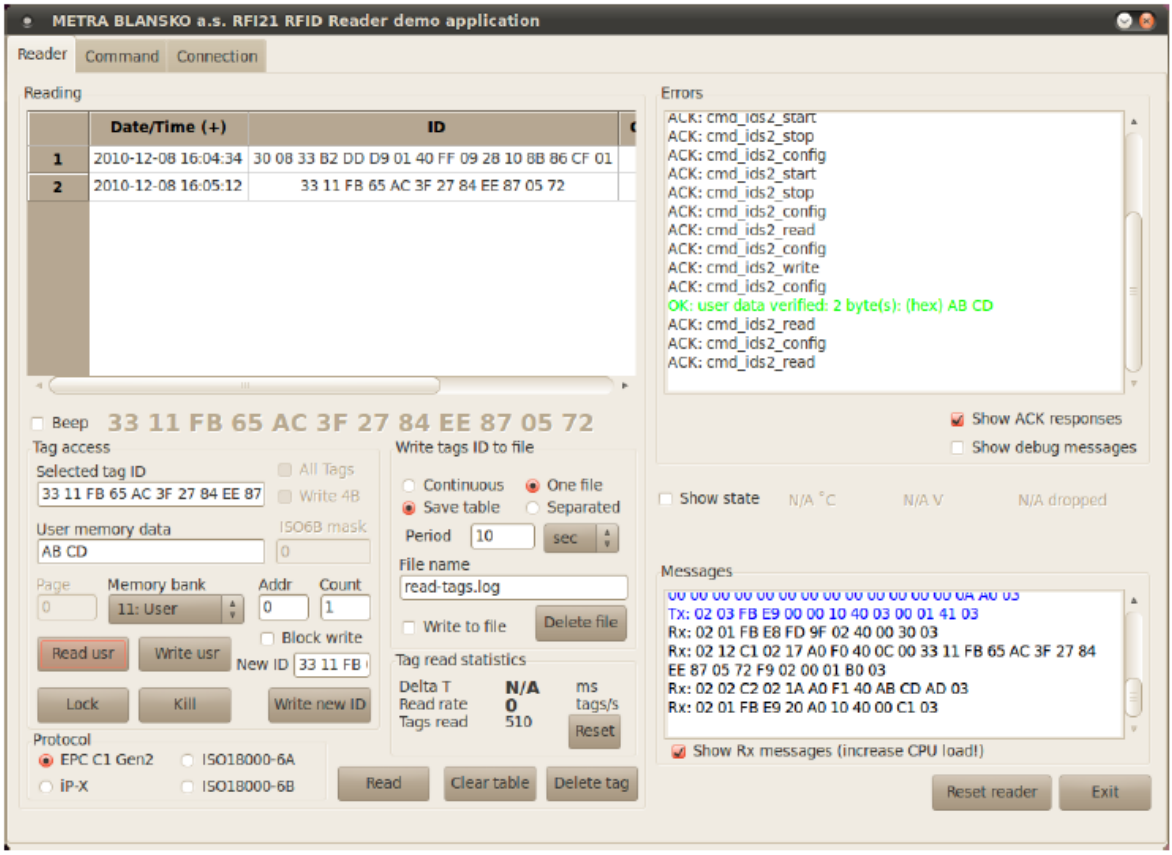
Obrázek 4.4: Uživatelské prostředí aplikace pro ovládání čtečky [49].

4.5 Metra Blansko RFI21.1

RFID čtečka společnosti Metra Blansko s označením RFI21.1 obsahuje rozhraní USB, UART a GPIO. Ke čtečce je možné připojit 1 anténu prostřednictvím SMA konektoru. Pracovní kmitočet se pohybuje v rozsazích 865 - 868 MHz a 902 - 928MHz. Maximální výstupní čtečky se pohybuje od 17 do 25 dBm. Čtečka podporuje standardy EPC Class 1 Gen 2, ISO18000-6 A a ISO18000-6 B [50]. Rozhraní LLRP 1.0.1 je podporováno pouze při použití převodníku mezi proprietárním protokolem [51]. Společnost Metra Blansko již ukončila výrobu těchto čteček [52].

Pro připojení ke čtečce je vyžadována nainstalovaná aplikace „RFI21 RFID Demo Application“ podporující OS Windows od verze XP a GNU/Linux 2.6 a vyšší. Čtečka se připojuje k PC přes rozhraní UART nebo USB [51].

Aplikace čtečky umožňuje čtení a změnu EPC tagu a umožňuje použití příkazů LOCK a KILL. Čtečka umožňuje zobrazit a změnit obsah uživatelské pamětí a RFU paměti, paměť TID lze obvykle jen zobrazit. Dále umožňuje nastavení citlivosti a výstupního výkonu čtečky. V seznamu načtených EPC tagů se zobrazuje počet načtení jednotlivých EPC tagů, RSSI a čas prvního načtení tagu. Dále lze natavit pracovní frekvenci a rychlost čtení [51].



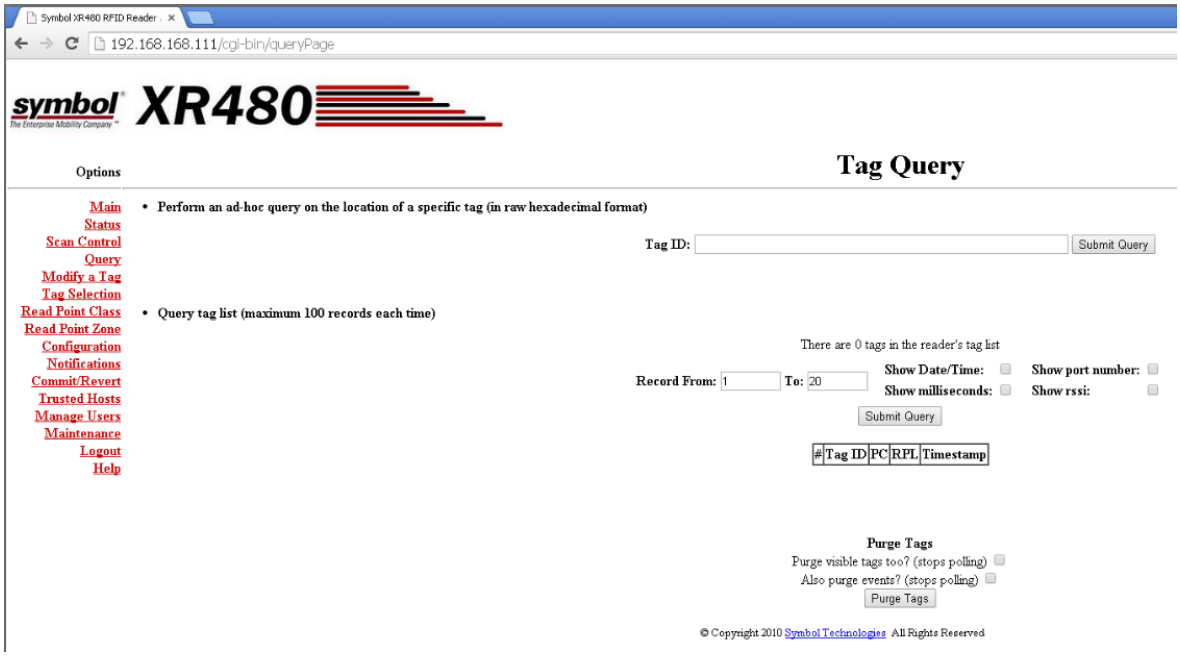
Obrázek 4.5: Uživatelské prostředí aplikace pro ovládání čtečky [51].

4.6 Motorola XR480

RFID čtečka společnosti Motorola s označením XR480 disponuje rozhraním Ethernet, USB v režimu Host, USB v režimu Device, RS232 a GPIO. Evropský model čtečky podporuje kmitočty 865,6 až 867,6 MHz a poskytuje maximální výkon 30 dBm. Čtečka je kompatibilní s protokoly EPCGlobal Class 0, EPCGlobal Class 1, EPCGlobal Class 1 Gen 2. Čtečka má webové rozhraní a podporuje protokol LLRP 1.0.1. [53]. Čtečka umožňuje připojit až 8 antén. Tato čtečka se již nevyrábí [55].

Pro zprovoznění čtečky je zapotřebí ji připojit do počítačové sítě. A pomocí přidělené IP adresy se buď přihlásit do webového rozhraní nebo se pomocí middleware připojit prostřednictvím protokolu LLRP a poté lze provést konfiguraci čtečky [53].

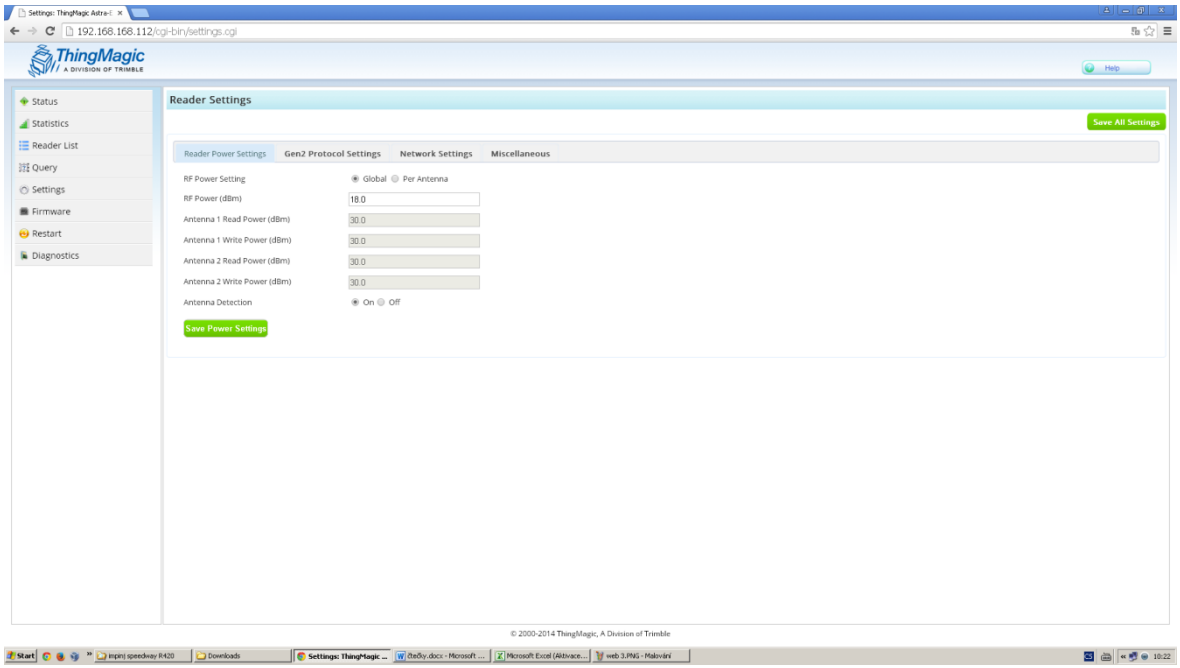
Webové rozhraní čtečky umožňuje čtení a změnu EPC tagu a umožňuje použití příkazů LOCK a KILL. Webové rozhraní umožňuje zobrazit a změnit obsah uživatelské pamětí a RFU paměti, paměť TID lze obvykle jen zobrazit. Dále lze provést nastavení výstupního výkonu a pracovní frekvence, export dat, konfiguraci filtrace tagů, dle EPC. V seznamu načtených EPC tagů se zobrazuje počet načtení jednotlivých EPC tagů, RSSI, čas prvního načtení tagu, ID portu antény, kde byly tagy načteny [53].



Obrázek 4.6: Uživatelské prostředí aplikace pro ovládání čtečky [53].

4.7 Thingmagic ASTRA-EX

RFID čtečka společnosti Thingmagic s označením ASTRA-EX disponuje rozhraními Ethernet s podporou PoE, USB, Console, GPIO. Podpora kmitočtů je v rozsahu 865,6 až 928 MHz dle geografického určení. Čtečka umožňuje připojení 2 antén. Maximální výstupní výkon je 31,5 dBm. Čtečka obsahuje webové rozhraní a podporuje rozhraní LLRP 1.0.1 a protokol EPCglobal Gen2 [56]. Cena čtečky je 995 $ [58].



Obrázek 4.7: Uživatelské prostředí aplikace pro ovládání čtečky.

Pro zprovoznění čtečky je zapotřebí ji připojit do počítačové sítě a pomocí přidělené IP adresy se buď přihlásit do webového rozhraní, nebo se pomocí middleware připojit prostřednictvím protokolu LLRP. Poté lze provést konfiguraci čtečky [57].

Webové rozhraní čtečky umožňuje čtení EPC tagů. Dále lze provést nastavení rychlosti čtení a výstupního výkonu celkově nebo i pro jednotlivé antény zvlášť. V seznamu načtených EPC tagů se zobrazuje počet načtení jednotlivých EPC tagů, RSSI a ID portu antény. IP adresu čtečky lze zadat manuálně nebo lze automaticky vyhledat dostupné čtečky v síti a zvolit čtečku ze seznamu nalezených [58].

Tab. 4.1 Porovnání funkcí aplikací čteček

	IMPINJ SPEED WAY R420	Arete POP	THINGMAGIC ASTRA-EX	Eureka	Motorola XR480	Elatec SR113	Metra Blan-sko RFI21.1
ID tagu EPC	x	x	x	x	x	x	x
Podpora LLRP	x	-	x	x	x	x	x
Počet načtení jednotlivých ID tagu celkově	x	x	x	x	x	-	x
Počet načtení jednotlivých ID tagu podle čteček	x	-	x	-	-	-	-
Nastavení rychlosti čtení	-	-	x	-	-	-	x
Maximální RSSI tagu	x	x	-	x	x	-	x
Čas prvního načtení tagu	x	-	-	-	x	-	x
Čas posledního načtení tagu	x	-	-	-	-	-	-
Nastavení výstupního výkonu čtečky	x	x	x	x	x	x	x
Nastavení výstupního výkonu pro jednotlivé antény	x	-	x	-	-	x	-
Čas mezi prvním a posledním načtením tagu	x	-	-	-	-	-	-
Hledání dostupných čteček a možnost výběru z nalezených	x	-	x	-	-	-	-
Manuální nastavení připojení ke čtečkám	x	x	x	x	x	x	x
Zapnutí a vypnutí portu antény	x	-	-	-	x	x	-
Nastavení pracovní frekvence	x	-	-	-	x	-	x
Přepis ID tagu	x	x	-	x	x	x	x
ID čtečky	-	-	x	-	-	-	-
ID portu antény, kde byly načteny tagy	-	-	x	x	x	x	-
Nastavení citlivosti	x	-	-	x	-	-	x
Konfigurace sítě	x	-	x	x	x	x	-
Flitrace tagů dle EPC	x	-	-	-	x	x	-
Export dat	-	x	-	-	x	-	-
Zápis do paměti tagu	x	x	-	x	x	x	x
Znefunkčnění tagu	x	x	-	-	x	x	x
Zamknutí tagu	x	x	-	-	x	x	x

4.8 Porovnání a závěr

Většina aplikací a webových rozhraní nepodporuje export dat, který je důležitý z hlediska zálohy, případně pro další zpracování. V reálném prostředí se také může stát, že se vyskytnou cizí tagy, které jsou nežádoucí, a proto je vhodná filtrace, která umožňuje nežádoucí tagy zakázat nebo povolit jen úzký výběr vlastních tagů.

Z tabulky je patrné, že žádná aplikace nebo webové prostředí neposkytuje veškeré existující funkce. Nejvíce funkcí nabízí aplikace čtečky Impinj, ale postrádá důležitou funkci export dat. Všechny aplikace podporují nastavení výstupního výkonu čtečky.

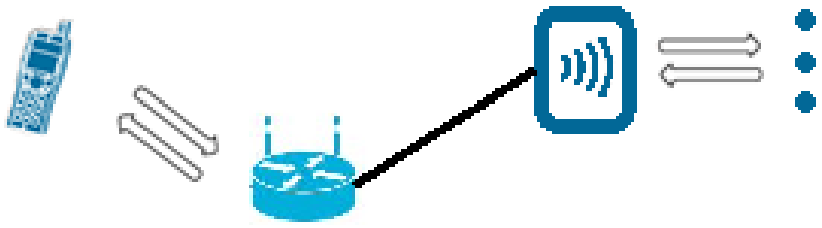
Při vývoji RFID middleware jsou proto implementovány funkce z této tabulky.

5 Realizace mobilního middleware

5.1 Úvod

Aplikace LLRPDroid je určena ke správě RFID čtečky, která podporuje protokol LLRP a je připojena v počítačové síti. V počítačové síti musí být přítomen Wi-Fi přístupový bod, k němuž lze připojit mobilní terminál s OS Android a nainstalovanou aplikací LLRPDroid.

Aplikace umožňuje nastavit výkon vysílače a citlivost přijímače RFID čtečky. Dále zobrazuje průběh komunikace mezi čtečkou a aplikací ve formě LLRP zpráv. Hlavní funkcí aplikace je načítání EPC (Electronic Product Code) tagů do inventáře. Pro načtené tagy aplikace umožňuje jejich uživatelskou modifikaci. Tagy lze prostřednictvím aplikace zamknout, číst a přepsat paměti USER, EPC, RFU a TID. Databázi inventáře lze exportovat na SD kartu případně odeslat emailem.



Obr. 5.1 Schéma propojení terminálu s RFID čtečkou

5.2 Popis zařízení

Vývoj aplikace byl realizován na notebooku Acer ASPIRE V3-571G.

Obr. 5.2 Notebook Acer ASPIRE V3-571G [68]

Tab. 5.3 Specifikace notebooku

typ procesoru	Intel Core i5-3210M 2.5GHz
operační systém	Microsoft Windows 7 Professional x64 CZ
displej	15.6", TN LCD, 1366 x 768 px
paměť	6 GB DDR3 RAM, 750 GB HDD
grafická karta	NVIDIA GeForce GT 640M, 2GB VRAM
přístup k síti	Wi-Fi, Blutooth, Ethernet

K vývoji aplikace byl použit tablet Asus Nexus 7. V průběhu vývoje aplikace došlo postupně k aktualizaci operačního systému v tabletu z verze 4.4.4 až na 5.1.1. Tablet poskytl dostatek výkonu i v raných fázích vývoje, kdy aplikace byla značně neoptimalizovaná.

Obr. 5.4 Tablet Asus Nexus 7 [69], [70], [71]

Tab. 5.2 Specifikace tabletu [72]

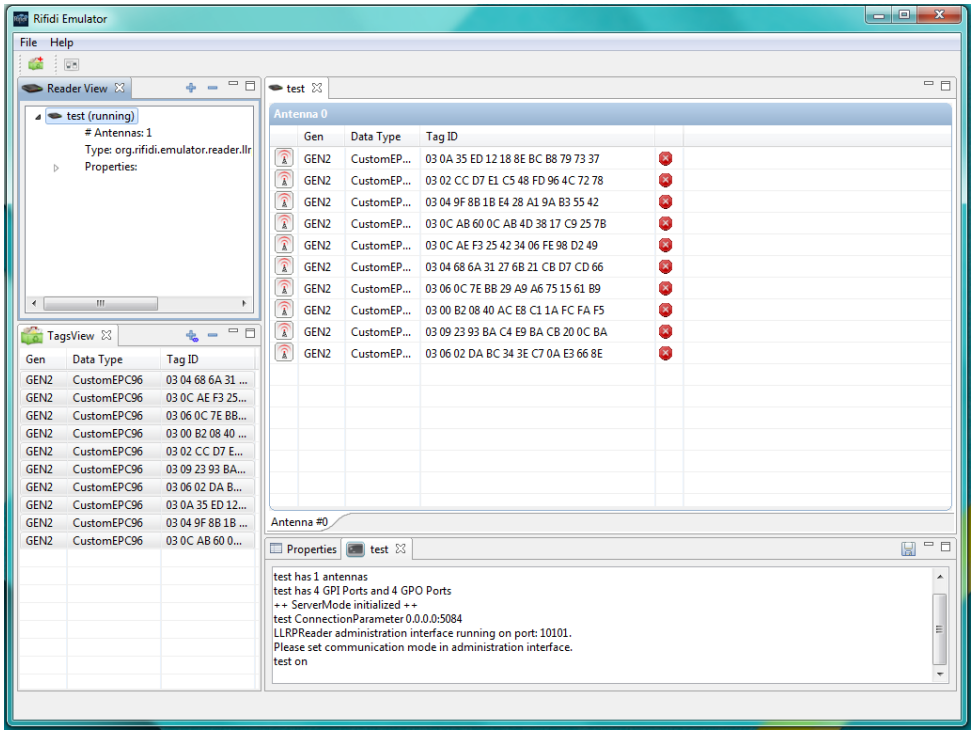
rozměry	199 x120 x 10,5 mm
hmotnost	340 g
displej	7", IPS LCD, 1280 x 800 px
konektivita	microUSB, Wi-Fi 802.11 b/g/n, Bluetooth, GPS
paměť	16 GB, 1 GB RAM
procesor	NvidiaTegra 3 T30 (4 x Cartex A9 1,3 GHz, 40nm)
operační systém	Android 4.2.2 Jelly Bean
baterie	4325 mAh

Pro vývoj pokročilých funkcí RFID čteček byla použita proprietární RFID čtečka vyvinutá při projektu EuReKa na ČVUT FEL viz kapitola 4.3.

5.3 Popis software

K naprogramování aplikace bylo použito Android SDK a IDE Eclipse s ADT doplňkem viz kapitola 3.4. Aplikace se do tabletu nainstalovala při spuštění zdrojového kódu v Eclipse a v tabletu zůstala nainstalovaná i po odpojení tabletu od notebooku.

Pro simulaci RFID čtečky byl použit software RiFiDi. RiFiDi je open-source napsaný v jazyce JAVA. Podporuje verzi protokolu LLRP 1.0.1. Je dostupný pro OS Windows a GNU/Linux s nainstalovaným prostředím JAVA [90]. RiFiDi je middleware, který umožňuje částečně emulovat funkci RFID čteček připojených do sítě a RFID tagy. V okně „properties“ se zobrazují příchozí a odchozí LLRP zprávy ve formátu XML. RiFiDi umožňuje emulovat nejen standardní LLRP, ale i čtečky ThingMagic, Symbol XR400 a další. Dále umožňuje emulovat tagy mnoha datových typů první i druhé generace. Neumožňuje nastavení výkonu a citlivosti emulované RFID čtečky a neumožňuje přístupové operace tagů. Pro tyto funkce bylo nutné použít reálnou RFID čtečku viz kapitola 5.2.



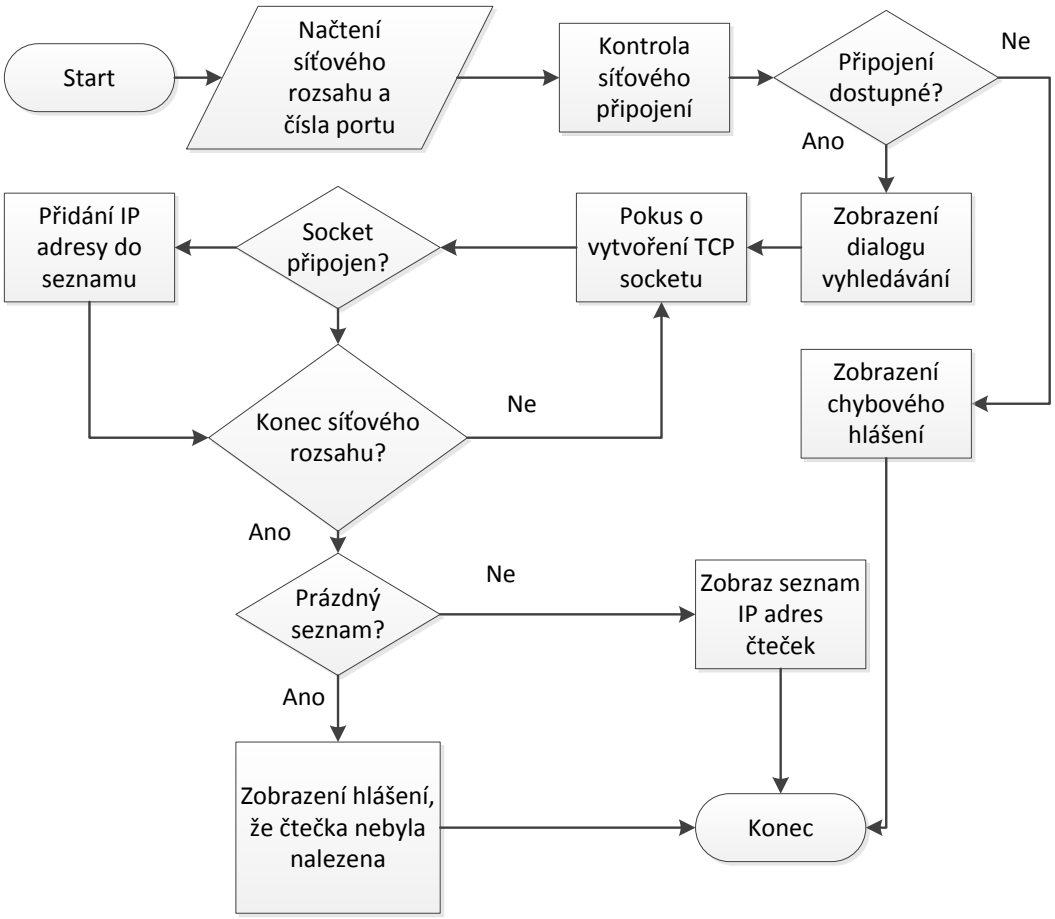
Obr. 5.5 Uživatelský prostředí middleware RiFiDi

5.4 Popis jednotlivých procesů

Hlavní procesy mobilní aplikace LLRPDroid jsou vyhledání čteček v síti, připojení ke čtečce, odpojení čtečky, načítání tagů, ukládání LLRP zpráv, nastavení čtečky, export databáze, čtení a zápis paměti tagu, zamknutí a znefunkčnění tagu.

5.4.1 Proces vyhledání čteček v síti

Cílem tohoto procesu je vyhledat všechny čtečky v síti z předem zadaného síťového rozsahu. Po uživatelově stisknutí tlačítka pro přidání čteček se z nastavení načte síťový rozsah a číslo portu. Poté se zkontroluje dostupnost síťového připojení. Pokud není dostupné, zobrazí se chybové hlášení. Pokud je dostupné, zobrazí se dialog vyhledávání a spustí se cykly pro zadaný síťový rozsah. Pomocí cyklů dochází postupně ke zvyšování čísla IP adresy. Uvnitř cyklu probíhá pokus o vytvoření „socketu“ z IP adresy určené cyklem a portu z nastavení. Pokud je připojen, tak se daná IP adresa přidá do seznamu a „socket“ bude rozpojen. Toto se opakuje do konce rozsahu. Poté dojde k zobrazení dialogu se seznamem nalezených čteček.

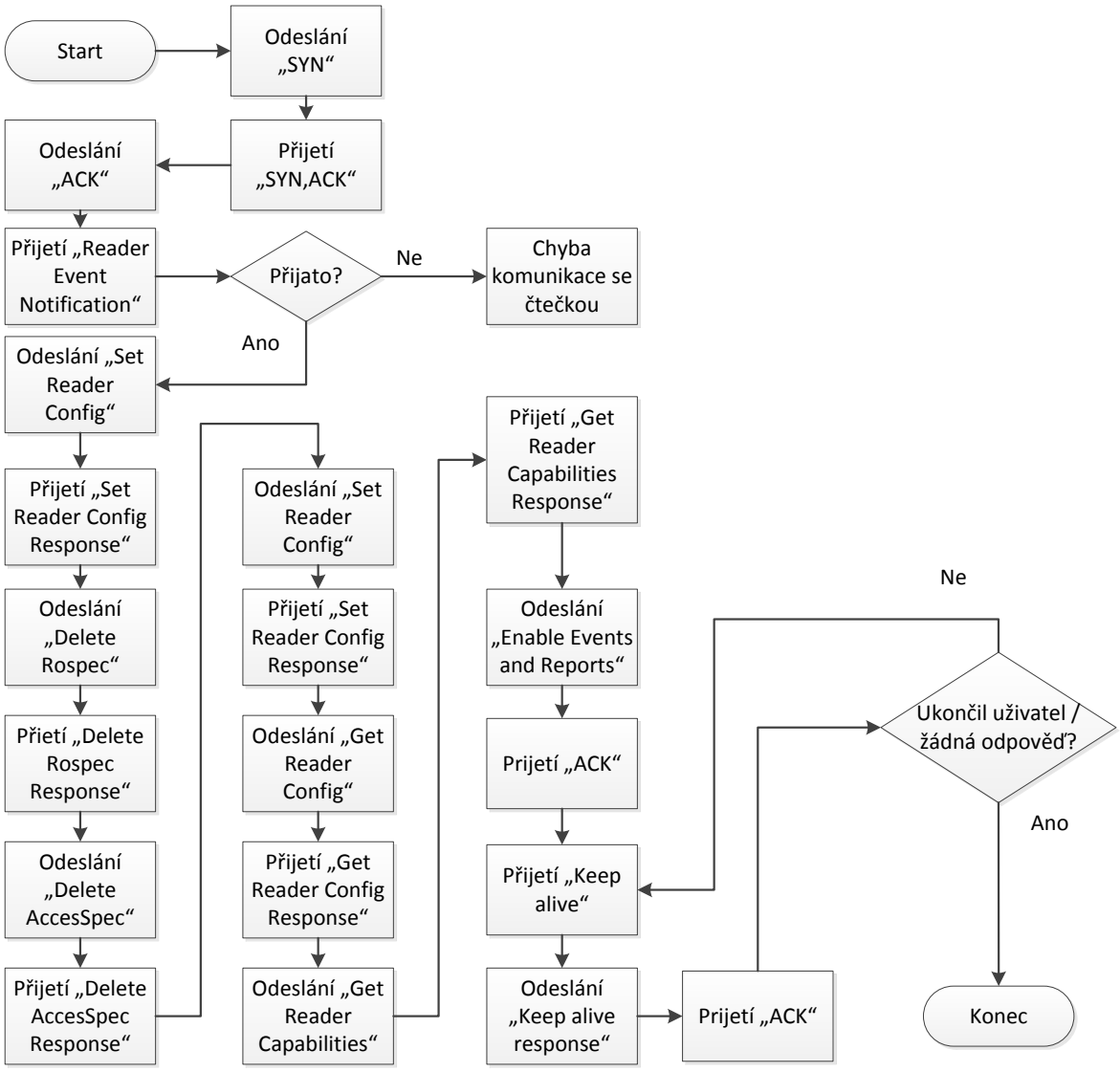


Obr. 5.6 Diagram procesu vyhledávání čteček

```
private Runnable testRunnable(final String host) {
          //Nové vlákno pro scan IP adresy
          return new Runnable() {
              public void run() {
                        //Zadaný port LLRP protokolu
                        int rPort=port;
                        // Vytvoření socketu pro testování IP rozsahu
                        try{
                        Socket con=new Socket ();
                        //Nastavení intervalu pro čekání na odpověď
                        int timeout = 100; //v milisekundách
                        //Připojení socketu na ip adresu definvanou cyklem
                        con.connect(new InetSocketAddress(host,rPort), timeout);
                        //Pokud je socket připojen
                        if (con.isConnected()) {
                         //Přiřazení do pomocné proměnné
                         reachable=true;
                         //Zavření socketu
                                con.close();
                                //Přidání IP adresy do seznamu nalezených čteček
                                list.add(host);
                        //Zachycení vyjímky, kdy socket není připojen
                    } catch (Exception c) {
                    }
              }
          };
      }
```

Obr. 5.7 Ukázka metody testRunnable

5.4.2 Proces připojení ke čtečce



Obr. 5.8 Diagram procesu připojení ke čtečce

Cílem tohoto procesu je úspěšně sestavit LLRP spojení mezi čtečkou a mobilní aplikací.

Po uživatelově výběru ze seznamu nalezených čteček, se uloží IP adresa, jméno čtečky a číslo portu do databázové tabulky RFID čteček. V komponentě LLRPService dojde k vytvoření instance objektu LLRPReader mapovaným na jméno čtečky. Objekt LLRPReader obsluhuje komunikaci mezi aplikací a čtečkou. V tomto objektu se sestaví TCP (Transmission Control Protocol) socket.

Dále čtečka pošle aplikaci zprávu READER_EVENT_NOTIFICATION obsahující parametr popisující události čtečky. Aplikace poté pošle zprávu SET_READER_CONFIG obsahující parametr „EventsAndReports“ s hodnotou „TRUE“ pro to, aby čtečka neposílala reporty a události. Čtečka následně odpoví zprávou SET_READER_CONFIG_RESPONSE obsahující parametr LLRPStatus. Pokud v tomto

parametru je hodnota 0, tak konfigurace proběhla úspěšně, pokud je v něm jiná hodnota, tak některý parametr zprávy SET_READER_CONFIG je chybně nastaven. Následně aplikace čtečce pošle zprávu DELETE_ROSPEC pomocí, které se vymažou všechny ROSpecs (Reader Operation Specifications) na základě hodnoty 0 v ROSpecID.

Čtečka smazání potvrdí zprávou DELETE_ROSPEC_RESPONSE obsahující 0 v parametru LLRPStatus. Aplikace poté pošle zprávu DELETE_ACCESSPEC pomocí, které se vymažou všechny AccessSpecs (Access Specifications) na základě hodnoty 0 v AccessSpecID. Čtečka smazání potvrdí zprávou DELETE_ACCESSPEC_RESPONSE obsahující 0 v parametru LLRPStatus. Aplikace poté pošle zprávu SET_READER_CONFIG s parametrem KeepaliveSpec určující periodické posílání zpráv KEEPALIVE. Čtečka potvrdí nastavení zprávou SET_READER_CONFIG_RESPONSE obsahující hodnotu 0 v parametru LLRPStatus. Aplikace poté pošle zprávu GET_READER_CONFIG vyžadující informace o anténách, konfiguraci čtečky, a vstupních a výstupních portech. Čtečka odpoví zprávou GET_READER_CONFIG_RESPONSE obsahující požadované informace.

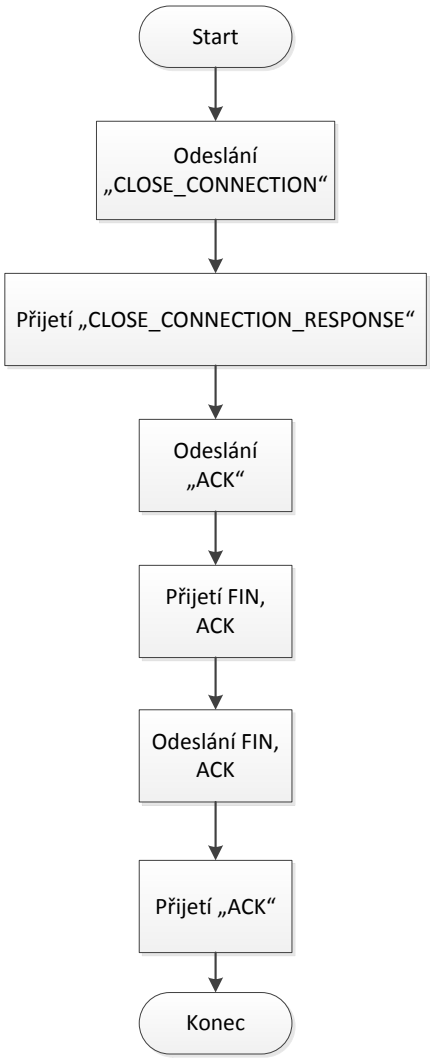
Aplikace si poté vyžádá všechny možnosti a parametry čtečky pomocí zprávy GET_READER_CAPABILITIES. Čtečka odpoví zprávou GET_READER_CAPABILITIES_RESPONSE obsahující informace o možnostech nastavení citlivosti, anténách, vysílacím výkonu, kmitočtovém rozsahu. Aplikace poté pošle zprávu ENABLE_EVENTS_AND_REPORTS čímž povolí čtečce posílat reporty a události. Výměna zpráv KEEPALIVE a KEEPALIVE_ACK je prováděna za účelem udržení LLRP připojení. Pokud by čtečka nepřijala několikrát KEEPALIVE_ACK, tak by mohla vypnout připojení. Pokud se uživatel aplikace odpojil od čtečky, tak proběhne proces odpojení čtečky.

```
public LLRPReader(Context ctx2,String ip, int p) throws IOException {
            //Přiřazení kontextu pro práci s databází
            ctx=ctx2;//
            //Přiřazení IP adresy
            ipAddress=ip;
            //Přiřazení portu čtečky
            readerPort=p;
            // Vytvoření TCP socketu pro komunikaci se čtečkou
            connection = new Socket(ipAddress,readerPort);
            //Vytvoření výstupního streamu pro zápis do socketu
            out = new DataOutputStream(connection.getOutputStream());
            //Vytvoření nového vlákna pro příjem zpráv čtečky
            rt = new ReadThread(connection);
            //Spuštění vlákna
            rt.start();
            // Čekání na odeslání READER_EVENT_NOTIFICATION
            pause(250);
            //Získání zprávy z vlákna pro příjem zpráv čtečky
            LLRPMessage m = rt.getNextMessage();
            //Přiřazení LLRP zprávy správnému typu
            READER_EVENT_NOTIFICATION rEventNoti = (READER_EVENT_NOTIFICATION) m;
            //Získání paramteru ReaderEventNotificationData
            ReaderEventNotificationData red = rEventNoti
                        .getReaderEventNotificationData();
            //Pokud není nulový připojení proběhlo v pořádku
            if (red.getConnectionAttemptEvent() != null) {

                                    …
```

Obr. 5.9 Ukázka části konstruktoru třídy LLRPReader

5.4.3 Proces odpojení čtečky

Cílem tohoto procesu je uzavření připojení mezi aplikací a čtečkou. Aplikace odešle zprávu CLOSE_CONNECTION, na tuto zprávu čtečka odpoví zprávou CLOSE_CONNECTION_RESPONSE, čtečka uzavře připojení s aplikací a dojde k uzavření TCP socketu.

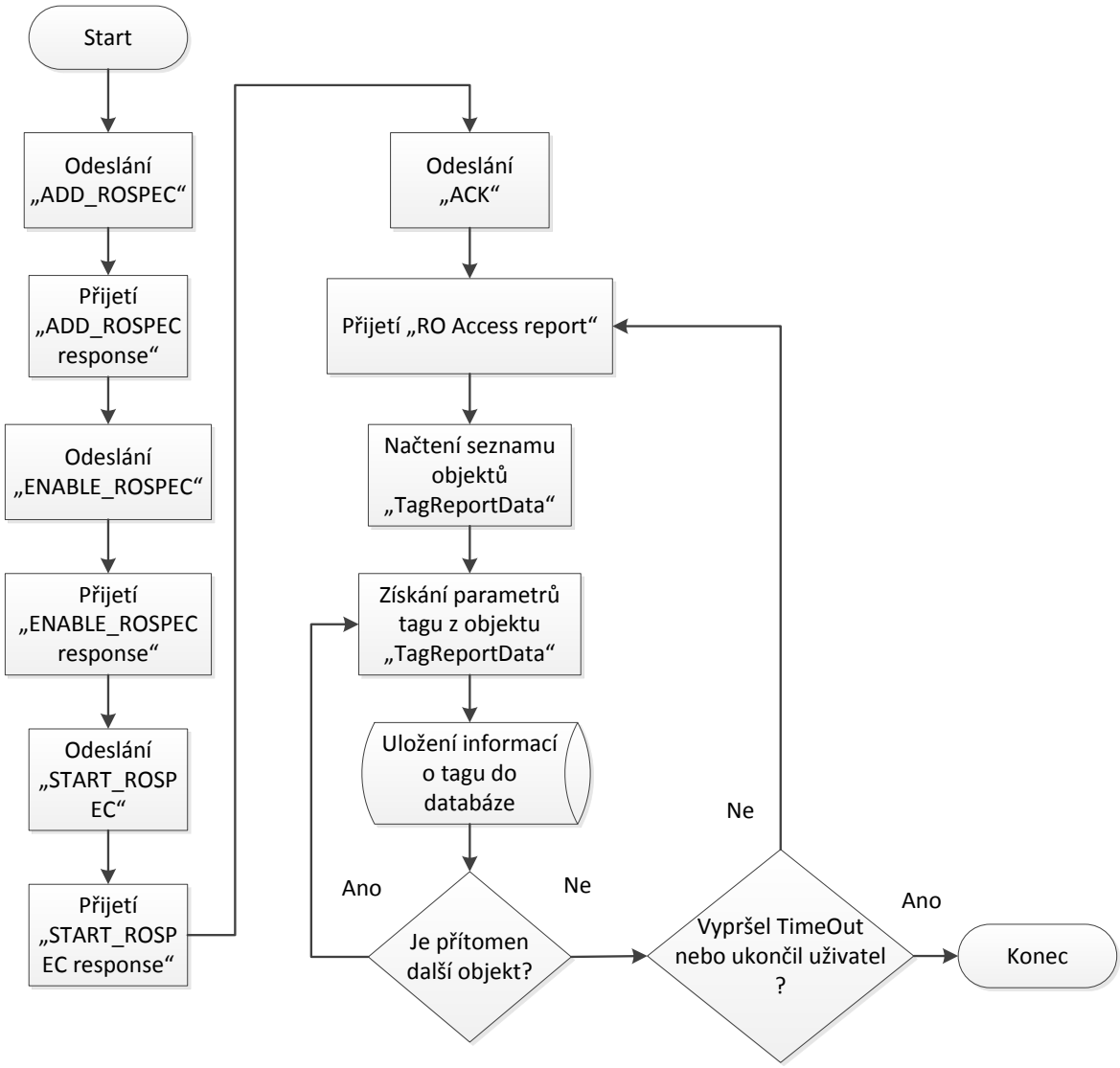


Obr. 5.10 Diagram procesu odpojení od čtečky

```
public void disconectReader() throws InterruptedException{
                        //Vytvoření zprávy CLOSE_CONNECTION
                        CLOSE_CONNECTION cc = new CLOSE_CONNECTION();
                        //Předání proceduře pro zápis do TCP Socketu
                        write(cc);
                        //Počkání na odpověď vlákna pro čtení z TCP Socketu
                        synchronized(rt)
                        {
                              rt.wait();
                        }

                                    …
```

5.11 Ukázka procedury disconnectReader

5.4.4 Proces spuštění inventarizace tagů



Obr. 5.12 Diagram procesu odpojení od čtečky

Cílem tohoto procesu je spuštění na čtečce inventarizaci tagů. Při zvolení uživatelem spuštění inventarizace tagů z kontextové nabídky po připojení k čtečce. Aplikace pošle zprávu ADD_ROSPEC obsahující sadu parametrů určující činnost inventáře čtečky a vyhledávaní tagů. Pokud je nastavení těchto parametrů úspěšné čtečka odpoví zprávou ADD_ROSPEC_RESPONSE s hodnotou 0 v parametru LLRPStatus.

Aplikace poté pošle zprávu ENABLE_ROSPEC s patřičným ROSpecID, tím se ROSpec dostane ze zakázaného stavu do neaktivního stavu. Pokud je ROSpec úspěšně nastaven do neaktivního stavu čtečka odpoví zprávou ENABLE_ROSPEC_RESPONSE s hodnotou 0 v parametru LLRPStatus. Poté aplikace pošle zprávu START_ROSPEC s patřičným ROSpecID, čtečka na základě ROSpec zahájí čtení tagů. Pokud je ROSpec úspěšně nastaven do aktivního stavu čtečka odpoví zprávou START_ROSPEC_RESPONSE s hodnotou 0 v parametru LLRPStatus.

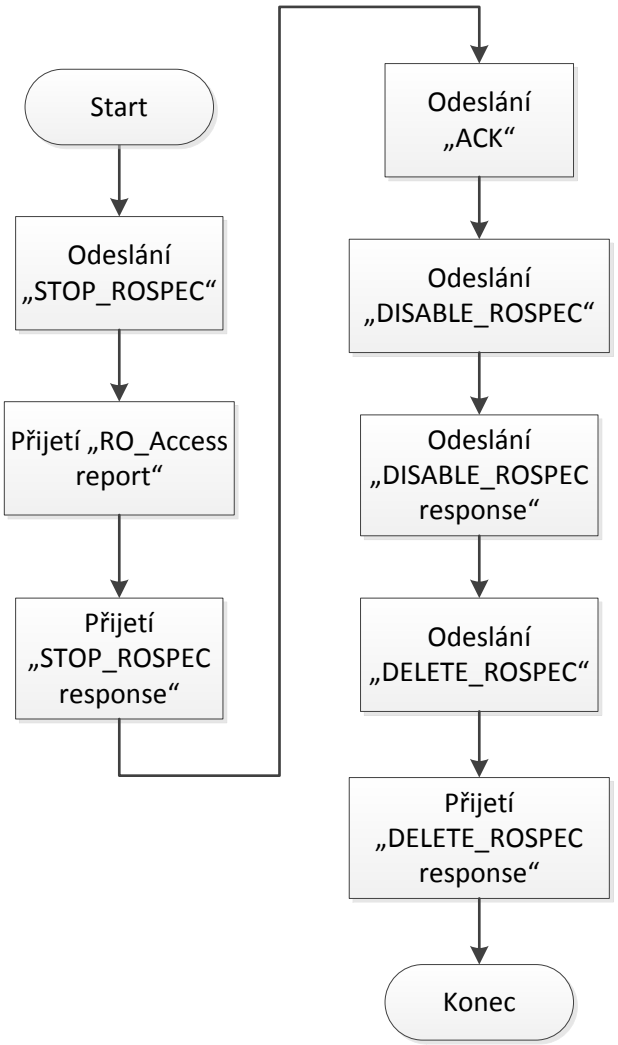
Čtečka poté odesílá zprávy RO_ACCESS_REPORT obsahující parametry inventarizovaných tagů. Mezi tyto parametry patří počet načtení tagu, hodnota EPC, RSSI, čas prvního načtení, čas posledního načtení, ID antény.

Zprávy RO_ACCESS_REPORT čtečka posílá do vypršení časovače, nebo pokud uživatel čtení tagů ukončí.

```
//Přiřazení LLRP zprávy správnému typu
RO_ACCESS_REPORT msg2=new RO_ACCESS_REPORT(msg);
//Získání seznamu dat o jednotlivých tazích
List<TagReportData> l=msg2.getTagReportDataList();
//Cykl pro načtení jednotlivých tagů
for(int i=0;i<l.size();i++){
	//Přiřazení načtených dat tagu
	TagReportData t=l.get(i);
	//Získání jednotlivých parametrů tagu
	String ePC=t.getEPCParameter().toString();
	String fst = t.getFirstSeenTimestampUTC().toString();
	String lst =t.getLastSeenTimestampUTC().toString();
	String rssi = t.getPeakRSSI().toString();
	String chanID = t.getChannelIndex().toString();
	String antID = t.getAntennaID().toString();
	String tagcount = t.getTagSeenCount().toString();

	//Deklarace objektu pro vložení do databáze
	ContentValues valuesEPC = new ContentValues();
	//Vložení jednotlivých hodnot do databáze
	valuesEPC.put(TagsTable.COLUMN_EPC, ePC);
	valuesEPC.put(TagsTable.COLUMN_FST, fst);
	valuesEPC.put(TagsTable.COLUMN_LST, lst);
	valuesEPC.put(TagsTable.COLUMN_RSSI, rssi);
	valuesEPC.put(TagsTable.COLUMN_ChanIND, chanID);
	valuesEPC.put(TagsTable.COLUMN_AntID, antID);
	valuesEPC.put(TagsTable.COLUMN_TagCount, tagcount);
                                        …
```

Obr. 5.13 Ukázka metody getTagInventory

5.4.5 Proces zastavení inventarizace tagů



Obr. 5.14 Diagram procesu zastavení inventarizace tagů

Cílem tohoto procesu je zastavení inventarizace tagů. Při zvolení uživatelem zastavení inventarizace tagů z kontextové nabídky, aplikace odešle zprávu STOP_ROSPEC s patřičným ROSpecID, tím se ROSpec dostane do neaktivního stavu.

Čtečka odpoví zprávou STOP_ROSPEC_RESPONSE s hodnotou 0 v parametru LLRPStatus. Aplikace poté pošle zprávu DISABLE_ROSPEC s patřičným ROSpecID, tím se ROSpec dostane do zakázaného stavu. Čtečka odpoví zprávou DISABLE_ROSPEC_RESPONSE s hodnotou 0 v parametru LLRPStatus. Aplikace poté pošle zprávu DELETE_ROSPEC s patřičným ROSpecID, čímž dojde k vymazání daného ROSpec ze čtečky. Úspěšné vymazání potvrdí čtečka zprávou DELETE_ROSPEC_RESPONSE s hodnotou 0 v parametru LLRPStatus.

```
public void stopReadingTags(){
      //Vytvoření zprávy STOP_ROSPEC
      STOP_ROSPEC stopROSpec = new STOP_ROSPEC();
      stopROSpec.setROSpecID(new UnsignedInteger(ROSPEC_ID));
      //Předání proceduře pro zápis do TCP Socketu
      write(stopROSpec);
      pause(250);

      //Vytvoření zprávy DISABLE_ROSPEC
      DISABLE_ROSPEC disableROSpec = new DISABLE_ROSPEC();
      disableROSpec.setROSpecID(new UnsignedInteger(ROSPEC_ID));
      //Předání proceduře pro zápis do TCP Socketu
      write(disableROSpec);
      pause(250);

      //Vytvoření zprávy DELETE_ROSPEC
      DELETE_ROSPEC deleteROSpec = new DELETE_ROSPEC();

                                          …
```

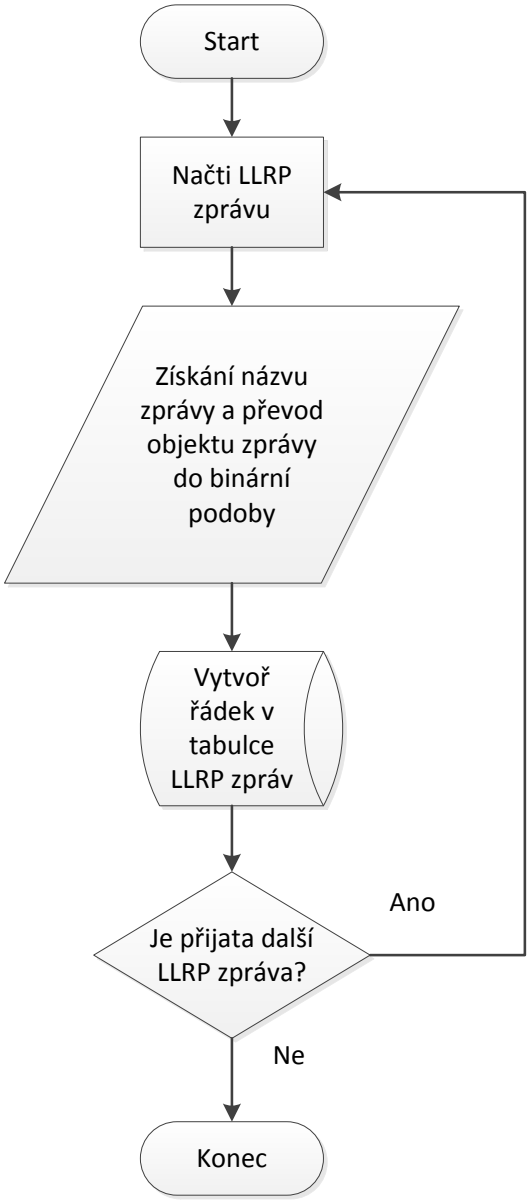
Obr. 5.15 Ukázka procedury stopReadindgTags

5.4.6 Proces ukládání LLRP zpráv

Cílem tohoto procesu je ukládáni veškerých LLRP zpráv pro účely ladění konfigurace čteček. Při přijmu LLRP zprávy nebo při jejím odeslání probíhá ukládání do databáze. Dojde k načtení typu zprávy a poté proběhne uložení do databáze. Do databáze se ukládá kromě názvu zprávy i její binární podoba. Ta se ukládá jako datový typ BLOB (Binary Large Object).

```
      //Převod objektu LLRP zprávy do binární podoby
      byte[] binMessage = msg.encodeBinary();
      //Deklarace objektu pro vložení do databáze
      ContentValues values = new ContentValues();
      //Vložení názvu zprávy a její binární podoby do databáze
      values.put(LLRPMessagesTable.COLUMN_NAME, s);
      values.put(LLRPMessagesTable.COLUMN_MESSAGES, binMessage);
      Uri uri = ((Context)
ctx).getContentResolver().insert(DBContentProvider.CONTENT_URI_MESSAGES,
                        values);

                                          …
```

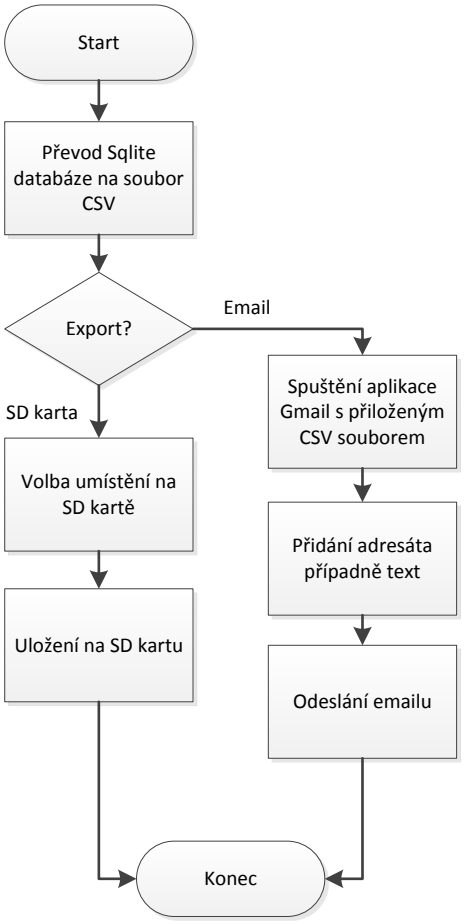
Obr. 5.16 Ukázka uložení LLRP zprávy do databáze



Obr. 5.17 Diagram uložení LLRP zprávy do databáze

5.4.7 Proces exportu databáze

Cílem tohoto procesu je exportovat interní SQLite databázi. Interní databáze se nejprve převede na soubor CSV, který je poté možné odeslat emailem na zvolenou adresu, nebo, který lze uložit na SD kartu.



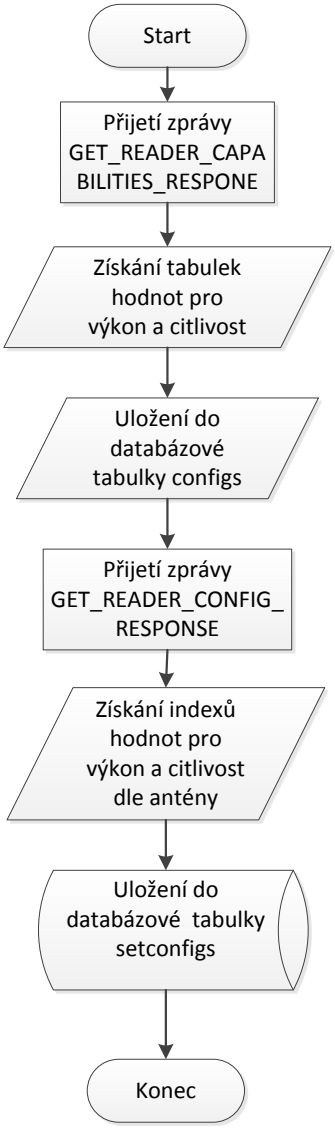
Obr. 5.18 Diagram exportu databáze

5.4.8 Proces získání informací o schopnostech čtečky a její aktuální konfigurace

Cílem tohoto procesu je získat informace o možnostech nastavení výkonu a citlivosti, aktuálním stavu čtečky. Tento proces probíhá při připojení ke čtečce.

```
//Přiřazení LLRP zprávy správnému typu
GET_READER_CAPABILITIES_RESPONSE msg2 = new GET_READER_CAPABILITIES_RESPONSE(msg);
//Přiřazení parametru pro určující regulaci RF
RegulatoryCapabilities rc= msg2.getRegulatoryCapabilities();
//Získání tabulky pro nastavení výkonu
UHFBandCapabilities uhfBC =rc.getUHFBandCapabilities();
List<TransmitPowerLevelTableEntry> l= uhfBC.getTransmitPowerLevelTableEntryList();
      //Získání tabulky pro nastavení výkonu
      for(int i=0;i<l.size();i++){
            TransmitPowerLevelTableEntry tplte = l.get(i);
            UnsignedShort us = tplte.getIndex();
            SignedShort tpv  = tplte.getTransmitPowerValue();
                                    …
```

Obr. 6.19 Ukázka získání parametrů výkonu



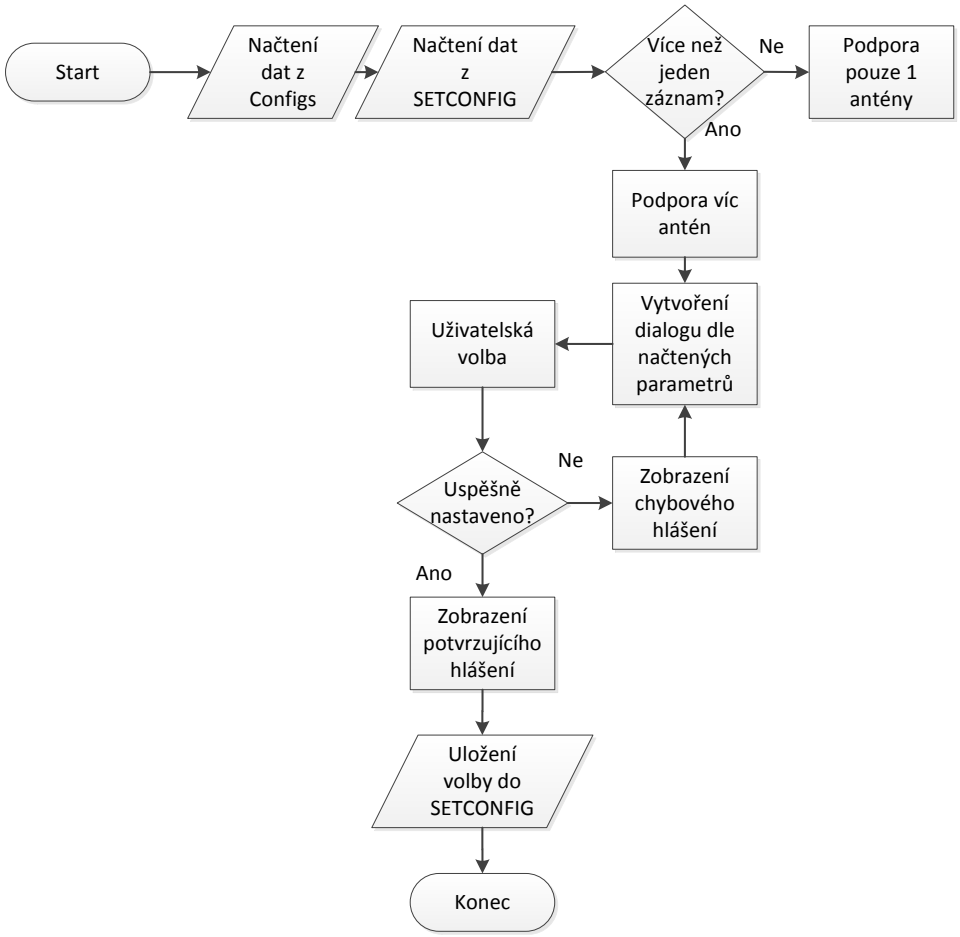
Obr. 5.20 Diagram procesu získání informací o schopnostech čtečky a její aktuální konfigurace

5.4.9 Proces nastavení čtečky

Cílem tohoto procesu je načtení dat z databáze pro možnosti uživatelského nastavení čtečky. Po uživatelově volbě dojde k vytvoření zprávy SET_READER_CONFIG s nastavenými parametry výkonu vysílače a citlivosti přijímače. Tyto parametry mohou být pro každou anténu nastavený zvlášť.

```
//Procedura pro odeslání nastavení čtečce
public void sendConfig(UnsignedShort transmitPower, UnsignedShort receiverSensiti-
vity, UnsignedShort antennaID){
	//Deklarace a přiřazení zprávy SET_READER_CONFIG
	SET_READER_CONFIG src = new SET_READER_CONFIG();

	//Nastavení citlivosti přijímače
	RFReceiver   rFReceiver= new RFReceiver();
	rFReceiver.setReceiverSensitivity(receiverSensitivity);
	//Nastavení výkonu vysílače
	RFTransmitter      rFTransmitter = new RFTransmitter();
	rFTransmitter.setTransmitPower(transmitPower);
	//Nastavení parametrů antény
	AntennaConfiguration ac = new AntennaConfiguration();
	ac.setRFReceiver( rFReceiver);
	ac.setRFTransmitter(rFTransmitter);
	//Pokud pamametr AntennaID je roven 0 tak se nastavení provede pro všechny
anény
	ac.setAntennaID(antennaID);
                                        …
```

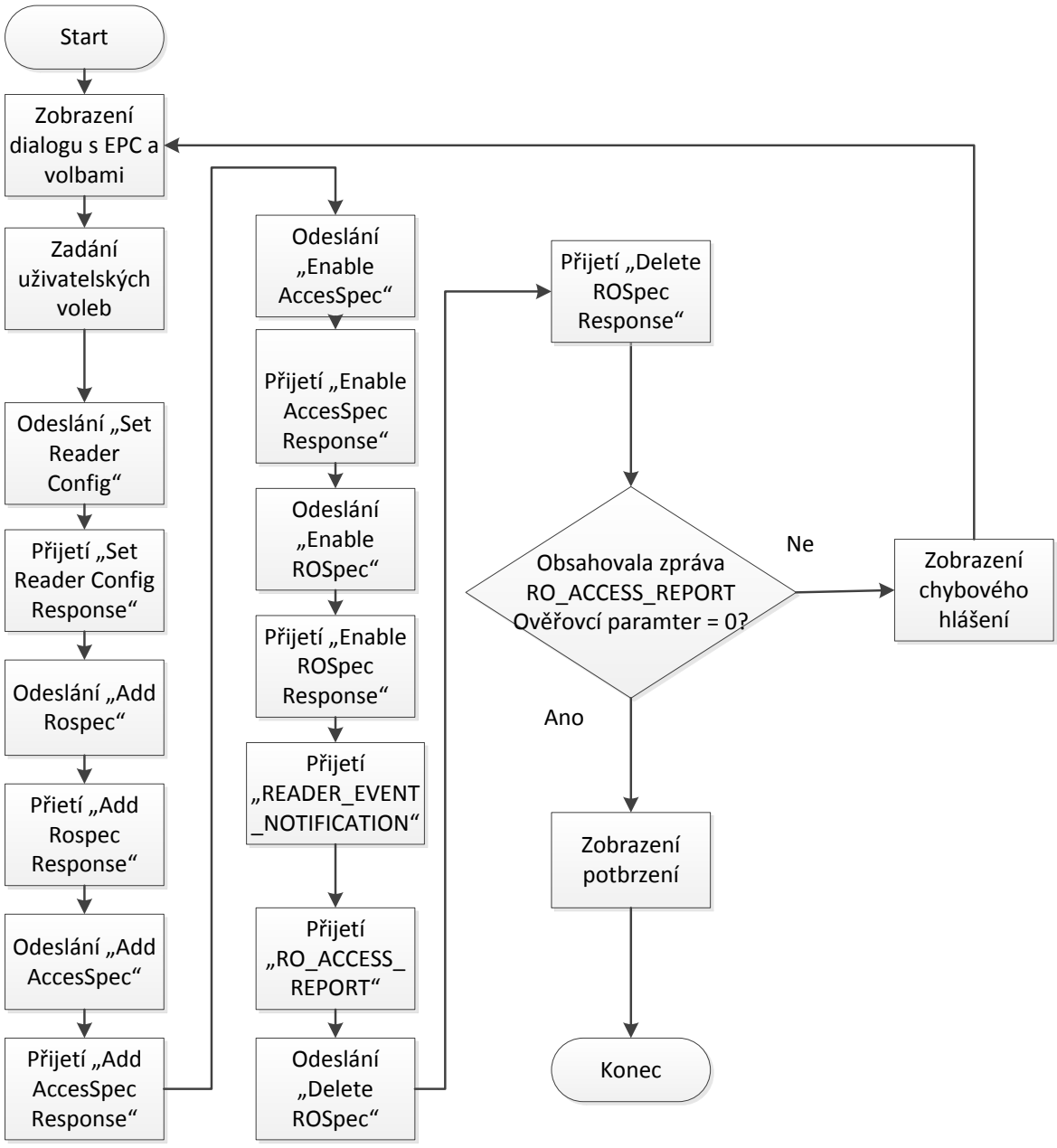
Obr. 5.21 Ukázka části procedury pro nastavení čtečky



Obr. 5.22 Diagram procesu nastavení čtečky

5.4.10 Proces přístupových operací tagů

Cílem tohoto procesu je poskytnutí přístupových operací nad tagy. Přístupovou operaci volí uživatel po stisknutí tagu ze seznamu tagů. Dle přístupové operace se zobrazí zadávací dialog. V něm uživatel vyplní potřebné údaje a poté dojde k vytvoření příslušného parametru AccessSpec dle přístupové operace.



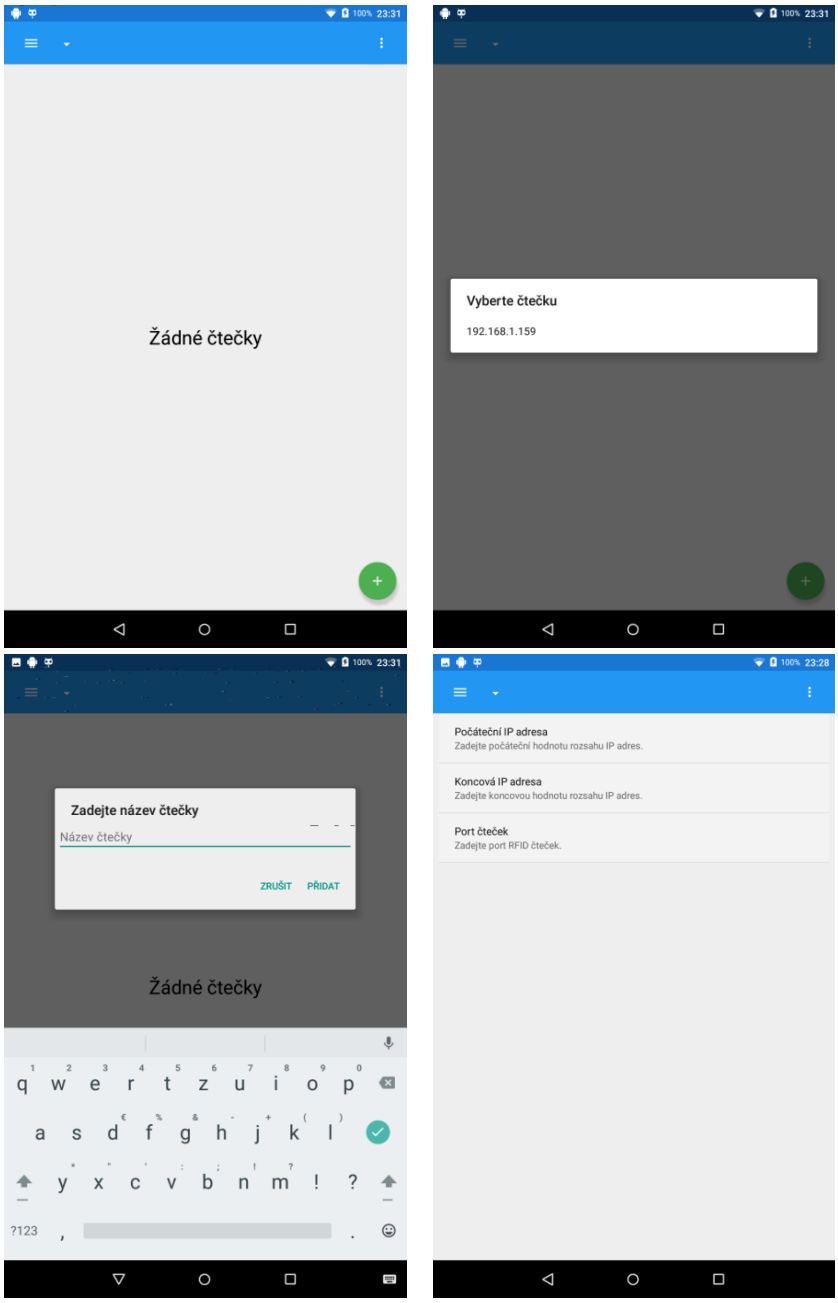
Obr. 5.23 Diagram procesu přístupových operací

```
C1G2TagSpec tagSpec= new C1G2TagSpec();
      //Nastavení parametru C1G2TargetTag dle conformance
      C1G2TargetTag targetTag1 = new C1G2TargetTag();

      TwoBitField mB = new TwoBitField();
      mem=1;
      mB.set(mem);
      //Parametr Paměťová oblast EPC = 1, RFU=0, TID=2, User = 3
      targetTag1.setMB(mB);
      //Parametr Pointer
      targetTag1.setPointer(null);
      //Parametr TagMask - určuje jaký tag nebo tagy se použijí, 0 značí všechny
      targetTag1.setTagMask(null);
      //Parametr TagData
      targetTag1.setTagData(null);
      //Paramter match
      Bit match = new Bit(1);
      targetTag1.setMatch(match);

                                          …
```

Obr. 5.24 Nastavení paramtru pro specifikaci tagů

5.5 Popis uživatelského rozhraní

Při návrhu uživatelského rozhraní bylo dbáno na dodržování Android Design GuideLines [75]. Uživatelské rozhraní je tvořeno čtyřmi obrazovkami. Seznamem RDID čteček, seznamem LLRP zpráv, obrazovkou nastavení, a seznamem inventarizovaných tagů. Z každé z těchto čtyř obrazovek se lze přepínat prostřednictvím komponenty „Navigation Drawer“.



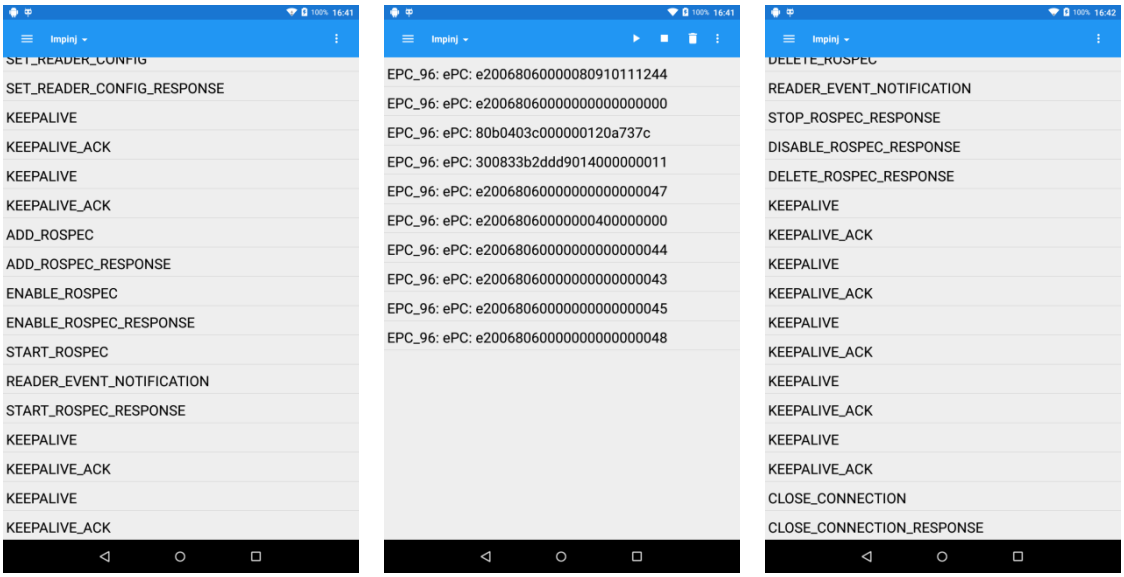
Obr. 5.25 Ukázky uživatelského rozhraní

5.6 Testování aplikace

V emulátoru RiFiDi byly provedeny testy ukládáním až 1000 tagů/s. Při prvotním testování s reálnými čtečkami se projevila absence některých funkcí emulátoru RiFiDi. Proto bylo nutné při dalším vývoji použít reálnou čtečku místo emulátoru.

5.6.1 Testování na reálné čtečce

Funkce aplikace byly otestovány na čtečce Impinj Speedway R420.



Obr 5.26 Testování inventarizace tagů

5.6.2 Testování na více zařízeních

Testování aplikace proběhlo na více různých zařízení, které zahrnovalo telefony a tablety. Na těchto zařízeních byla aplikace testována.

- Nexus 5
- Nexus 7
- Nexus 9
- LG Optimus L9 II
- ZOPO ZP330

Výše zmíněná zařízení se lišila výkonem, velikostí, rozlišením displeje a verzí OS Android. Aplikace podporuje OS Android 4.1 a vyšší. Na výše uvedených zařízeních aplikace pracovala správně. Docházelo ke správnému zobrazování prvků. A přestože se výkonově lišily, na rychlosti aplikace to nebylo příliš znát.

6 Závěr

Cílem této práce bylo navrhnout a vytvořit mobilní aplikaci pro správu UHF RFID čtečky pro OS Android.

V úvodních kapitolách byla probrána problematika RFID a LLRP.

V praktické části je rozebírána rešerše funkcí UHF RFID čteček. Dále návrh a realizace „middleware“. Dopodrobna byly probrány procesy aplikace a komunikace mezi čtečkou a aplikací. Poté došlo k představení grafického rozhraní, z čeho je složené a jak se chová. A nakonec bylo probráno testování aplikace.

Aplikace prošla testováním s reálnou čtečkou.

Budoucí vývoj aplikace pravděpodobně bude směřovat k implementaci EPCIS (Electronic Product Code Information Services) repozitáře a zabezpečení LLRP autentizací TLS (Transport Layer Security) 1.2 pomocí knihovny Bouncy Castle. Lze tak zajistit zabezpečení komunikace se čtečkou, která tuto funkci podporuje.

7 Reference

[1] UJBÁNYAI, Miroslav. Programujeme pro Android. Praha: Grada, 2012. ISBN 978-80-247-3995-3.

[2] OHA [online]. 2013 [cit. 2013-05-16]. Dostupné z: http://www.openhandsetalliance.com/oha_overview.html

[3] Apache License 2.0. *2013* [online]. [cit. 2013-05-16]. Dostupné z: http://www.apache.org/licenses/LICENSE-2.0.html

[4] Source.android.com. [online]. [cit. 2013-05-15]. Dostupné z: http://source.android.com

[5] Google Play. [online]. [cit. 2013-05-15]. Dostupné z: https://play.google.com/store

[6] Video: Michal Šrajer - Android pro vývojáře i uživatele - 1.část [online]. 2009. Dostupné z: http://www.youtube.com/watch?v=2-PvVkxNO2Y

[7] android-4-2-jelly-bean.jpg, In: http://www.pcmag.com, 2013, [cit. 2013-05-15]. Dostupné z http://www2.pcmag.com/media/images/312849-google-android-4-2-jelly-bean.jpg

[8] 1.0-android.jpg, In: http:// talkandroid.com , 2013, [cit. 2013-05-15]. Dostupné z http://img.talkandroid.com/uploads/2012/11/1.0-android.jpg

[9] IDC.com [online]. 2015 [cit. 2015-12-02]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[10] Krátké ohlédnutí za historií Androidu. *Svetandroida.cz* [online]. [cit. 2013-05-14]. Dostupné z: http://www.svetandroida.cz/kratke-ohlednuti-za-historii-androidu-201305

[11] Developer.android.com [online]. [cit. 2015-02-12]. Dostupné z: http://developer.android.com/about/index.html

[12] Source.android.com. [online]. [cit. 2015-02-12]. Dostupné z: http://source.android.com/devices/tech/dalvik/configure.html

[13] Developer.android.com [online]. 2013 [cit. 2013-05-14]. Dostupné z: http://developer.android.com/index.html

[14] Source.android.com. [online]. [cit. 2013-05-15]. Dostupné z: http://source.android.com

[15] Jan Balata – Development Android [online]. 2012. [cit. 2013-05-14]. Dostupné z: http://pda.felk.cvut.cz/system/assets/2255/original/pda_lecture_04_android.pdf?1349780006

[16] Developer.android.com [online]. 2013 [cit. 2013-05-14]. Dostupné z: http://developer.android.com/images/system-architecture.jpg

[17] Video: Pavel Petřek - Android Nálevna [online]. 2010. [cit. 2013-04-16] Dostupné z: <http://www.youtube.com/watch?v=-j85JZ_XW7A>

[18] Bogdan Petrovan. AndroidAuthority.com [online]. 2015. [cit. 2015-02-20] Dostupné z: http://www.androidauthority.com/android-5-0-lollipop-official-538842/

[19] Dave Burke. Android Official Blog. [online]. 2015. [cit. 2015-03-10] Dostupné z: http://officialandroid.blogspot.cz/2015/03/android-51-unwrapping-new-lollipop.html

[20] Video: Dan Sandler. Google I/O 2015 - What's new in Android. [online]. 2015 [cit. 2015-05-28]. Dostupné z: https://www.youtube.com/watch?v=ndBdf1_oOGA&list=PLOU2XLYxmsIJkggzcouEOEcnjDIyJNyAN&index=2

[30] Developer.android.com. [online]. 2015 [cit. 2015-01-05]. Dostupné z: http://developer.android.com/ tools/help/adt.html

[31] Developer.android.com [online]. 2015 [cit. 2015-01-05]. Dostupné z: http://developer.android.com/ tools/help/adb.html

[41] PHYCHIPS INC. ARETE POP Smart RFID Dongle Reader. [Online]. 2014. [citováno 4. 3. 2015] Dostupné z: http://arete-mobile.com/pdf/ARETE%20POP_RFID%20Dongle%20Reader(2014_1.5).pdf

[42] coreRFID [online]. [citováno 4. 3. 2015] Dostupné z: http://www.rfidshop.com/arete-pop---uhf-dongle-reader-2162-p.asp

[43] Elatec GmbH, IDentification Systems. UHF Reader SR113 [CD]. 2007. [citováno 4. 3. 2015] Dostupné z: ELATEC_CD/DB UHF SR113.pdf

[44] Elatec GmbH, IDentification Systems. SR113 User Manual [CD] 2008. [citováno 4. 3. 2015] Dostupné z: ELATEC_CD/TH SR113 Starter Kit.pdf

[45] Google, Inc. ARETE POP [software]. Leden 2014 [4.3..2015] Dostupné z: https://play.google.com/store/apps/details?id=phychips.arete.newver Požadavky na systém: Android 2.3.3

[41] PHYCHIPS INC. ARETE POP Smart RFID Dongle Reader. [Online]. 2014. [citováno 4. 3. 2015] Dostupné z: http://arete-mobile.com/pdf/ARETE%20POP_RFID%20Dongle%20Reader(2014_1.5).pdf

[42] coreRFID [online]. [citováno 4. 3. 2015] Dostupné z: http://www.rfidshop.com/arete-pop---uhf-dongle-reader-2162-p.asp

[43] Elatec GmbH, IDentification Systems. UHF Reader SR113 [CD]. 2007. [citováno 4. 3. 2015] Dostupné z: ELATEC_CD/DB UHF SR113.pdf

[44] Elatec GmbH, IDentification Systems. SR113 User Manual [CD] 2008. [citováno 4. 3. 2015] Dostupné z: ELATEC_CD/TH SR113 Starter Kit.pdf

[45] Google, Inc. ARETE POP [software]. Leden 2014 [4.3..2015] Dostupné z: https://play.google.com/store/apps/details?id=phychips.arete.newver Požadavky na systém: Android 2.3.3

[46]https://support.impinj.com/hc/en-us/article_attachments/204611107/Impinj_SpeedwayR_installation_and_operations_guide.pdf

[47]http://www.atlasrfidstore.com/impinj-speedway-revolution-r420-uhf-rfid-reader-4-port/

[48]http://rfid.atlasrfidstore.com/hs-fs/hub/300870/file-1326702221-pdf/Tech_Spec_Sheets/Impinj/ATLAS_R420_Speedway_Rev_Brochure.pdf

[49] Multireader 6.6.11.240 Windows XP, Windows 7

[50] CD rfi21_1_datasheet_en.pdf

[51] http://www.techfass.cz/archive/m_rfi21_original_cz.pdf

CD rfi21_1_navod_cs.pdf

[52] http://metrablansko.cz/cs/

[53] https://www.manualowl.com/m/Motorola/XR450/Manual/346737

[54]https://ams.com/eng/Support/Demoboards/UHF-RFID/UHF-RFID-Reader-ICs/AS3993-Demo-Kit-Femto

[55] http://www.motorolasolutions.com/en_xu.html

[56]http://www.thingmagic.com/index.php/integrated-readers/astra-ex

[57] http://www.thingmagic.com/images/Downloads/Docs/M6Astra-EX_UserGuide_Nov13.pdf

[58] http://www.atlasrfidstore.com/thingmagic-astra-ex-integrated-rfid-reader-poe/

[51] http://www.techfass.cz/archive/m_rfi21_original_cz.pdf

CD rfi21_1_navod_cs.pdf

[52] http://metrablansko.cz/cs/

[53] https://www.manualowl.com/m/Motorola/XR450/Manual/346737

[54]https://ams.com/eng/Support/Demoboards/UHF-RFID/UHF-RFID-Reader-ICs/AS3993-Demo-Kit-Femto

[55] http://www.motorolasolutions.com/en_xu.html

[57] http://www.thingmagic.com/images/Downloads/Docs/M6Astra-EX_UserGuide_Nov13.pdf

[56]http://www.thingmagic.com/index.php/integrated-readers/astra-ex

[58] http://www.atlasrfidstore.com/thingmagic-astra-ex-integrated-rfid-reader-poe/

68] V3-571.png. In: acer.com [online]. © 2012 Acer Inc. [vid. 12.09.2012]. Dostupné z http://static.acer.com/up/Resource/Acer/Notebooks/AGW2%20Aspire%20V3/Photo%20Gallery/20120919/V3-571-531-black-photo-gallery-04.png

[69] 91_28986a2d47.jpg. In: smartmania.cz [online]. © 2012 SMARTmania s.r.o. [vid. 10.10.2012]. Dostupné z http://smartmania.cz/images/katalog/91_28986a2d47.jpg

[70] 91_762297df05.jpg. In: smartmania.cz [online]. © 2012 SMARTmania s.r.o. [vid. 10.10.2012]. Dostupné z http://smartmania.cz/images/katalog/91_762297df05.jpg

[71] 91_761c0df629.jpg. In: smartmania.cz [online]. © 2012 SMARTmania s.r.o. [vid. 10.10.2012]. Dostupné z http://smartmania.cz/images/katalog/91_761c0df629.jpg

[90] Emulator User's Guide. [online]. Transcends LLC. [vid. 19.01.2015] Dostupné z:

http://wiki.rifidi.net/index.php?title=Emulator_User%27s_Guide

Seznam zkratek

AOT	Ahead of Time
ALE	Application Level Event
LLRP	Low Level Reader Protocol
EPCIS	Electronic Product Code Information Services
UHF	Ultra high frequency
OHA	Open Handset Alliance
GNU	GNU’s Not Unix
SDK	Software Development Kit
DVM	Dalvik Virtual Machine
ART	Android RunTime
JVM	Java Virtual Machine
DEX	Dalvik EXecutable
API	Advanced Programable Interface
JIT	Just In Time
HTTP	Hypertext Tranfer Protocol
UID	User ID
XML	Extensible Markup Language
UI	User Interface
TCP	Transmission Control Protocol
TID	Tag ID
RFU	Reserved for Future Use
RSSI	Received Signal Strength Indication
PoE	Power over Ethernet
TLS	Transport Layer Security)

Obsah přiloženého CD

zitta_tomas.pdf