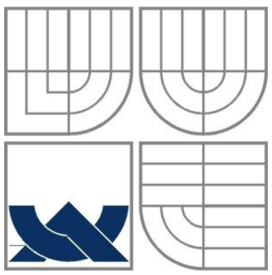
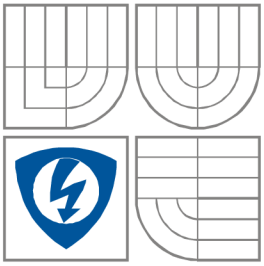


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

ANALYZÁTOR RFID SIGNÁLU JAKO SW RADIO NA BÁZI USB DVB-T PŘIJÍMAČE

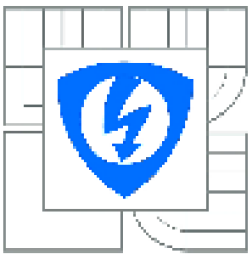
RFID SIGNAL ANALYZER USING USB DVB-T RECEIVER BASED ON SW DEFINED RADIO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Ondřej Sládek
AUTHOR

VEDOUCÍ PRÁCE Ing. Dr. Techn. Vojtěch Derbek
SUPERVISOR

BRNO, 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student: Ondřej Sládek ***ID:*** 154870
Ročník: 3 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Analyzátor RFID signálů jako SW radio na bázi USB DVB-T přijímače

POKYNY PRO VYPRACOVÁNÍ:

Analyzujte možnosti USB DVB-T přijímače jako univerzálního přijímače komplexního IQ signálu. Analyzujte možnosti adaptace jeho ovladače zařízení tak, aby bylo možno tento přijímač využít jako vektorový signálový analyzátor pro RFID. Navrhněte algoritmy pro implementaci funkcí frekvenční a časové analýzy.

V prostředí Labview implementujte SW rozhraní pro komunikaci s přijímačem. Implementujte streaming dat do souboru. Pomocí toolkitů prostředí Labview implementujte navržené funkce jednoduchého spektrálního analyzátoru s funkcí demodulátoru signálu.

DOPORUČENÁ LITERATURA:

[1] REED, J.H., Software Radio: A Modern Approach to Radio Engineering, Prentice Hall, 2002.

[2] OsmocomSDR [online]. [cit. 11.5.2014]. Dostupné na http://sdr.osmocom.org/trac/wiki/rtl-sdr.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 28.5.2015

Vedoucí práce: Ing. Dr. Techn. Vojtěch Derbek
Konzultanti bakalářské práce:

doc. Ing. Tomáš Kratochvíl, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato bakalářská práce se zabývá vývojem aplikace analyzátoru RFID signálu pro pásmo UHF s využitím USB DVB-T přijímače. Práce nejprve popisuje RFID systémy obecně a dále se zaměřuje na standard EPC Class-1 Generation-2. Jsou v ní analyzovány možnosti USB DVB-T přijímače jako univerzálního přijímače komplexního IQ signálu a možnosti jeho propojení s LabVIEW. Dále je popsáno vytvořené komunikační rozhraní a samotná aplikace v LabVIEW. Na závěr práce jsou uvedeny ukázky zachycené RFID komunikace a je představeno možné budoucí využití vytvořeného kódu pro aplikaci rádiového skeneru.

KLÍČOVÁ SLOVA

RFID, SDR, USB DVB-T přijímač, RTL-SDR, LabVIEW, wrapper DLL

ABSTRACT

This bachelor's work deals with the development of software analyzer of RFID signals in the UHF band using a USB DVB-T receiver. First part of the work describes RFID systems generally with emphasis on EPC Class-1 Generation-2 standard. The work also analyses possibilities of using a USB DVB-T receiver to receive complex IQ signal and possible options for interfacing it with LabVIEW. The created interface is described as well as the LabVIEW application itself. Furthermore, examples of recorded RFID communication are shown and possible future use of the developed code for a radio-band scanner is demonstrated at the end of the work.

KEYWORDS

RFID, SDR, USB DVB-T receiver, RTL-SDR, LabVIEW, wrapper DLL

SLÁDEK, O. *Analyzátor RFID signálů jako SW radio na bázi USB DVB-T přijímače*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 44 s. Vedoucí bakalářské práce Ing. Dr. Techn. Vojtěch Derbek.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Analyzátor RFID signálů jako SW radio na bázi USB DVB-T přijímače“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu bakalářské práce Ing. Dr. Techn. Vojtěchu Derbekovi za trpělivost při konzultacích, vstřícnost a ochotu učit a vysvětlovat.

OBSAH

Seznam obrázků viii

Seznam tabulek x

Úvod 1

1 RFID 2

1.1 Tagy 2

1.2 Čtečky 3

1.3 Protokol Gen2 4

1.3.1 Průběh komunikace 4

1.3.2 Požadavky na čtečku 5

1.3.3 Požadavky na tag 7

2 USB DVB-T přijímač jako SDR 9

2.1 Softwarové rádio 9

2.2 Vznik levného SDR 10

2.3 Rozbor hardwarové části 11

2.3.1 Tuner 11

2.3.2 RTL2832U 13

2.4 RTL-SDR software 13

2.4.1 Ovladače a knihovny 14

2.4.2 SDR aplikace 14

3 Analýza RFID signálů 16

3.1 IQ signál 16

3.2 Časová analýza 17

3.3 Spektrální analýza 18

3.3.1 DFT 18

3.3.2 FFT 20

3.4 Specifika RFID signálů 20

4 Komunikace s externím hardwarem v LabVIEW 22

4.1 NI-VISA 22

4.2 TCP spojení 23

4.3 Sdílené knihovny DLL 23
4.3.1 Princip wrapperu DLL knihovny 23
4.3.2 Kompatibilita mezi C a LabVIEW 24

5 Wrapper DLL **25**
5.1 Inicializace 25
5.2 Konfigurace 26
5.3 Funkce pro čtení 26
5.3.1 Synchronní čtení 26
5.3.2 Asynchronní čtení 27
5.3.3 Implementace asynchronního čtení 27
5.4 Dosažená funkčnost 28

6 Aplikace v LabVIEW **29**
6.1 Popis činnosti programu 29
6.1.1 Start programu 30
6.1.2 Čtení dat 30
6.1.3 Zpracování přijatých dat 30
6.2 Uživatelské prostředí 31
6.2.1 Ovládání programu 31
6.2.2 Zobrazované grafy a hodnoty 32
6.3 Ukázky zachycené komunikace 32
6.3.1 Časové průběhy 32
6.3.2 Spektrální analýza 35

7 Skener pásma **38**
7.1 Popis aplikace 38
7.2 Ukázky 39

8 Závěr **41**

Literatura **42**

Seznam symbolů, veličin a zkratek **43**

SEZNAM OBRÁZKŮ

Obrázek 1.1 Pasivní UHF tagy, ukázka rozměrů 3
Obrázek 1.2 Čtečka Metra RFI21.1 [2] 4
Obrázek 1.3 Definice parametrů ASK modulací [3] 5
Obrázek 1.4 Způsob kódování symbolů 0 a 1 v pulzně intervalovém kódování PIE [3] 6
Obrázek 1.5 Časový průběh preambule a Frame-Sync [3] 7
Obrázek 1.6 Symboly a sekvence dat při kódování FM0 [3] 7
Obrázek 1.7 Ukázka Millerova kódování tříbitových sekvencí dat [3] 8
Obrázek 2.1 Blokové schéma softwarového rádia současnosti [4] 9
Obrázek 2.2 Ukázka kompatibilního RTL-SDR přijímače [8] 11
Obrázek 2.3 Zjednodušené blokové schéma R820T, převzato z [8] 12
Obrázek 2.4 Blokové schéma RTL2832U [8] 13
Obrázek 2.5 Ukázka prostředí SDR# 14
Obrázek 3.1 Blokové schéma postupu pro získání IQ dat [12] 16
Obrázek 3.2 Příklad časového průběhu diskrétního IQ signálu [13] 18
Obrázek 3.3 Výsledek DFT pro harmonický signál konečné délky s frekvencí a) 8 kHz, b) 8,5 kHz, c) 8,75 kHz [12] 20
Obrázek 4.1 Princip wrapperu DLL 24
Obrázek 6.1 Vývojový diagram analyzátoru 29
Obrázek 6.2 Čelní panel aplikace 32
Obrázek 6.3 Příkaz QueryRep bez průměrování 33
Obrázek 6.4 Příkaz QueryRep s průměrováním 33
Obrázek 6.5 Příkaz Query 34
Obrázek 6.6 Příkaz Req_RN s odpovědí tagu 34
Obrázek 6.7 Část odpovědi tagu - RN16 35
Obrázek 6.8 Výkonové spektrum čtečky 35
Obrázek 6.9 Spektrum čtečky s korekcí 36
Obrázek 6.10 Špička lokálního oscilátoru 36
Obrázek 6.11 Špička LO při použití průměrování 37
Obrázek 6.12 Odpovědi tagu ve spektru 37
Obrázek 7.1 Čelní panel aplikace skeneru 38
Obrázek 7.2 Postup sestavování výsledného spektra 39

Obrázek 7.3 Vysílání DVB-T 39
Obrázek 7.4 Vysílání FM rozhlasu 40
Obrázek 7.5 Parazitní impulsy 40

SEZNAM TABULEK

Tabulka 1.1 Používaná frekvenční pásma v RFID komunikaci [1] 2
Tabulka 1.2 Přípustné rozptyly parametrů modulace ASK [3] 6
Tabulka 2.1 Frekvenční rozsahy vybraných tunerů [9] 12
Tabulka 6.1 Příkazy čtečky [3], upraveno 31

ÚVOD

Technologie RFID nachází stále větší uplatnění v oblasti monitorování pohybu věcí, zvířat či osob. K jejímu rozšiřování přispívá i klesající trend pořizovacích nákladů tagů a čtecích zařízení. RFID proto nezůstalo pouze výsadou vědeckých či vojenských institucí, naopak se stává nepostradatelným pomocníkem v odvětví výroby či logistiky, slouží pro monitorování pohybu zboží v obchodech atd.

Cílem této bakalářské práce je prozkoumat možnosti snadného odposlouchávání a analýzy RFID komunikace v pásmu UHF a poté realizace aplikace analyzátoru RFID komunikace v LabVIEW. Toho má být dosaženo vytvořením vhodného software pro USB DVB-T přijímač. Vedle frekvenčního rozsahu jsou hlavním kritériem výběru vhodného USB DVB-T přijímače jeho pořizovací náklady, aby byla výsledná aplikace dostupná i radioamatérům a jiným nadšencům. V této práci je tedy propojena technologie RFID se softwarovým rádiem jako metodou pro zpracování signálů.

Vektorové analyzátory pro RFID běžně dostupné na trhu jsou sice kvalitní, ale tomu také odpovídá jejich cena, která je činí těžko dostupnými. Naproti tomu, USB DVB-T přijímač je možné pořídit za řádově 10 $. V kombinaci s vhodným softwarem by mělo být možné vytvořit tímto způsobem systém pro odposlech RFID komunikace v pásmu UHF, který by mohl sloužit např. pro výukové účely.

Struktura písemné části práce je ovlivněna rozdělením práce na dvě etapy. Nejprve bylo nezbytné seznámit se se zadaným tématem a shromáždit velké množství informací z různých zdrojů, aby byl zvolen co nejoptimálnější přístup pro implementaci zadané aplikace. Na základě získaných poznatků byl později vytvořen analyzátor RFID v LabVIEW. První část práce lze tedy považovat za teoretický rozbor, zatímco druhá část se téměř výhradně zabývá praktickým řešením.

První kapitola této práce představuje technologii RFID a hlouběji se věnuje standardu pro komunikaci v pásmu UHF. V další kapitole jsou rozebrány možnosti použití USB DVB-T přijímače pro analýzu signálu a představen princip činnosti vybraného přijímače, který byl použit pro realizaci projektu. Způsoby analýzy signálů popisuje třetí kapitola, ve které jsou zároveň naznačeny možné komplikace vzhledem k omezeným možnostem levného přijímače a specifickým vlastnostem RFID signálů. Ve čtvrté kapitole jsou uvedeny důvody, proč bylo pro vývoj software vybráno prostředí LabVIEW a jakým způsobem mohou být v tomto prostředí sbírána data z USB přijímače.

Praktická část práce je uvedena pátou kapitolou, která se zaměřuje na wrapper DLL jakožto komunikační rozhraní. Samotná aplikace v LabVIEW je popsána v kapitole šesté, ve které jsou také ukázány vybrané výstupy aplikace. Sedmá kapitola se věnuje programu pro skenování vytíženosti rádiového prostředí, který byl realizován nad rámec zadání. V závěru jsou dosažené výsledky shrnuty a je diskutována možnost budoucího využití vytvořeného software.

1 RFID

Radiofrekvenční identifikace je technologie, sloužící k efektivnímu sledování pohybu zboží, zvířat i osob. RFID systém typicky obsahuje dva hlavní elementy: tag jako monitorovaný objekt a čtečku jakožto monitorovací zařízení. Identifikační data tagu i příkazy čtečky jsou přenášeny bezdrátově pomocí vhodně modulovaného signálu. Oproti svému předchůdci čárovým kódům má RFID mnoho výhod. Především nepotřebuje pro komunikaci mezi tagem a čtečkou přímou viditelnost, dokáže uchovat větší množství dat než jednorozměrný čárový kód. V porovnání s dvourozměrnými QR kódy mohou mít RFID tagy možnost programování, tedy úpravy uložených dat, což rozšiřuje možnosti jejich uplatnění.

RFID systémy mohou pracovat v různých frekvenčních pásmech, viz tabulka 1.1. Liší se především dosahem a přenosovou rychlostí. Tato práce se dále zabývá pouze UHF tagy pracujícími v pásmu 865 – 868 MHz.

Tabulka 1.1 Používaná frekvenční pásma v RFID komunikaci [1]

Pásmo	Frekvenční rozsah	Rychlost přenosu	Dosah
LF	120–150 kHz	Nízká	10 cm
HF	13,56 MHz	Nízká až střední	0,1 - 1 m
UHF	433 MHz	Střední	1 – 100 m
UHF (Evropa)	865-868 MHz	Střední až vysoká	1 – 12 m
UHF (Severní Amerika)	902-928 MHz	Střední až vysoká	1 – 12 m
SHF	2 450-5 800 MHz	Vysoká	1 – 2 m
SHF	3,1–10 GHz	Vysoká	až 200 m

1.1 Tagy

Tag, nebo také transpondér, bývá umístěn na objektu, který chceme sledovat. V nejběžnějším uspořádání se skládá z antény a integrovaného obvodu, ve kterém jsou uložena data. Od pracovní frekvence se odvíjí rozměry tagů, které můžeme přibližně odhadnout z obrázku 1.1. Tagy se obvykle dělí na:

- pasivní – energii potřebnou k vysílání získávají z přijímaného signálu
- aktivní – mají vlastní zdroj energie

Jelikož pasivní tagy nemají vlastní zdroj energie, vysílají vždy až v reakci na vysílání čtečky. Na přijímaný nemodulovaný signál namodulují data a výsledný signál je odeslán přes anténu tagu. Tento způsob komunikace se označuje termínem backscattering.

Aktivní tagy mohou samy iniciovat přenos. Díky vlastnímu napájení jsou schopné komunikovat na větší vzdálenosti, proto se dají použít v náročnějších aplikacích.

Někdy se jako třetí skupina tagů uvádí semiaktivní či semipasivní tagy. Tyto

transpondéry čekají stejně jako pasivní tagy na pokyn čtečky, než začnou vysílat. Navíc obsahují baterii, která může sloužit ke zvýšení vysílaného výkonu nebo jako napájení připojeného senzoru.

Obrázek 1.1 Pasivní UHF tagy, ukázka rozměrů

1.2 Čtečky

Čtečka je zařízení, které skrz anténu komunikuje s tagem. Převádí rádiové vlny na digitální informaci, která je dále zpracovávána např. hostitelským PC. Vysílání čtečky je také ovládáno softwarově. Při manipulaci s pasivními tagy je úkolem čtečky vyhledávat tagy v dosahu, vysílat příkazy a čekat na odpověď. U aktivních tagů čtečka čeká na signál vyslaný tagem. V obou případech musí mít implementovány antikolizní algoritmy, které umožní zvládnout komunikaci i v případě současné odpovědi více tagů.

Z používaného frekvenčního pásma vyplývají požadavky na anténu čtečky. Pro pásmo UHF se používají antény s rozměry srovnatelnými s délkou vlny $\lambda \approx 33$ cm [1]. V praxi mají největší uplatnění antény s kruhovou polarizací, jejichž útlum se při rotaci lineárně polarizované přijímací antény nemění. Na obrázku 1.2 je ukázána čtečky firmy METRA BLANSKO a.s., která byla při testování analyzátoru použita jako referenční.

Obrázek 1.2 Čtečka Metra RFI21.1 [2]

1.3 Protokol Gen2

Pro RFID je stejně jako pro jiné způsoby elektronické komunikace důležitá standardizace samotného komunikačního procesu. Definováním pravidel pro komunikaci se významně snižuje riziko chybné interpretace přenášeného sdělení. Pro RFID existuje několik standardů, které je možno použít v závislosti na účelu aplikace.

Protokol EPCglobal UHF Class 1 Generation 2, dále jen Gen2, definuje požadavky na RFID systémy s pasivními tagy, pracující v pásmu 860 MHz – 960 MHz. V roce 2006 se stal protokol Gen2 součástí standardu ISO 18000-6C, čímž byla zabezpečena vzájemná kompatibilita systémů pracujících s tímto standardem. Tato práce se bude dále zabývat pouze tímto protokolem. Pro větší přehlednost jsou v následujícím textu příkazy uvedeny kurzívou (např. *Query*) a stavy tagu kurzívou s tučným písmem (např. ***ready***).

1.3.1 Průběh komunikace

Protokol Gen2 definuje tři základní typy operací, které čtečka provádí v rámci komunikace s populací tagů. Jsou to operace Select, Invertory a Access, neboli výběr, inventarizace a přístup do paměti. Čtečka má v každém kole k dispozici sadu příkazů, pomocí kterých může měnit stav tagu, číst z něj nebo zapisovat do jeho paměti.

V kole výběru *Select* může čtečka specifikovat, s jakými tagy chce dále komunikovat. Kritériem pro výběr může být např. EPC – Electronic product code (unikátní identifikační číslo tagu). Výběr se provádí povinným příkazem *Select*, v závislosti na vyhodnocení kritérií výběru může tag invertovat své bity SL flag nebo invertoried a svůj stav změnit na ***ready***.

Inventarizaci začíná čtečka příkazem *Query*. Příkaz *Query* obsahuje důležitý parametr *Q*, kterým je celé číslo v rozsahu 0 – 15. Z něj vyplývá pravděpodobnost, s jakou budou tagy odpovídat. Nastavení parametru *Q* tudíž hraje velkou roli v systémech s početnou populací tagů. Po přijetí příkazu *Query* vygeneruje tag náhodné číslo z intervalu $0 – 2^Q - 1$ a uloží si jej do paměti. Čtečka potom používá příkaz

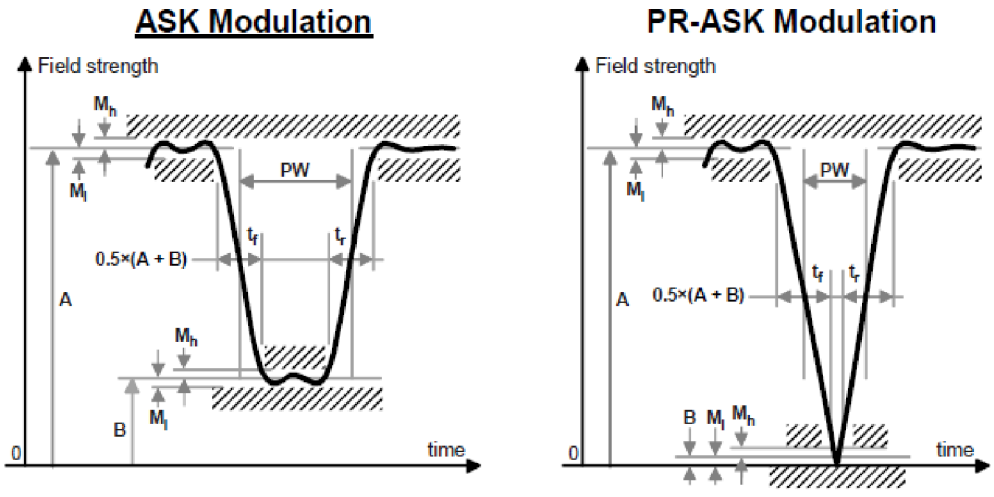
QueryRep jako pokyn k dekrementaci čísla uloženého v paměti tagu nebo *QueryAdjust* pro aktualizaci parametrů předešlého příkazu *Query*.

Pokud je číslo uložené v paměti tagu rovno nule, tag vygeneruje náhodné 16bitové číslo RN16 a odvysílá jej. Čtečka potvrdí příjem příkazem ACK obsahujícím totéž číslo. Pokud jsou obě čísla shodná, znamená to, že přenos proběhl v pořádku, a tag se přepne do stavu ***acknowledged***. Následné odvysílání příkazu *QueryRep* nebo *QueryAdjust* způsobí změnu stavu tagu zpět na ***ready***. Pokud je číslo v paměti tagu větší než nula, tag nezačne vysílat, dokud neobdrží *Q*-krát příkaz *QueryRep*.

Aby se tag mohl zúčastnit kola Access, musí být ve stavu ***acknowledged***. Čtečka požaduje nové číslo RN16 vysláním příkazu *Req_RN*, které následně slouží pro ověřování přístupu jako tzv. handle. V tomto kole může použít čtečka příkazy *Write*, *Read*, *Kill*, *Lock* atd. Každý příkaz čtečky musí jako parametr obsahovat handle. Tag potom porovnává handle příkazu s hodnotou uloženou v paměti, a pouze v případě shody vykoná přijatý příkaz.

1.3.2 Požadavky na čtečku

Čtečka vysílá data jednoduchou amplitudovou modulací vysokofrekvenční nosné, konkrétně může využít dvoustavové modulace DSB-ASK, SSB-ASK, nebo PR-ASK. V jejich časovém průběhu sledujeme hloubku modulace, šířku pulzu PW, přípustné zvlnění a dobu náběžné a sestupné hrany. Tyto veličiny definuje obrázek 1.3. Modulace se dvěma postranními pásmy je nejjednodušší na obvodovou realizaci, ale oproti jednostranné modulaci zabírá ve spektru dvojnásobnou šířku. V náročnějších aplikacích se používá PR-ASK (Phase-Reversal) modulace, u které se s každou hranou datového signálu změní fáze, přičemž amplituda se téměř nemění. Z uvedených modulací má nejlepší poměr nosná-šum a nejmenší šířku ve spektru. Požadavky na časový průběh obálky signálu jsou vztaženy k referenčnímu časovému intervalu Tari, který může být nastaven na hodnotu 6,25 µs až 25 µs. Přesné hodnoty můžeme nalézt v tabulce 1.2.

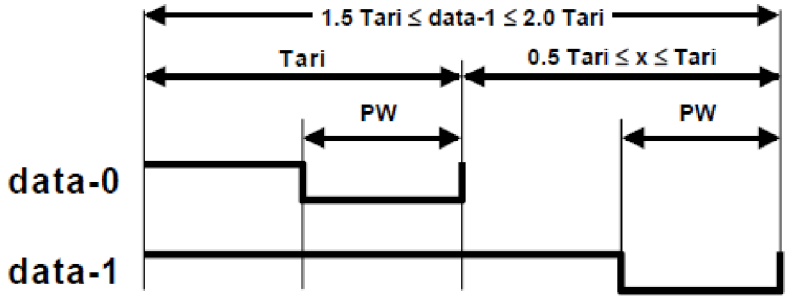


Obrázek 1.3 Definice parametrů ASK modulací [3]

Tabulka 1.2 Přípustné rozptyly parametrů modulace ASK [3]

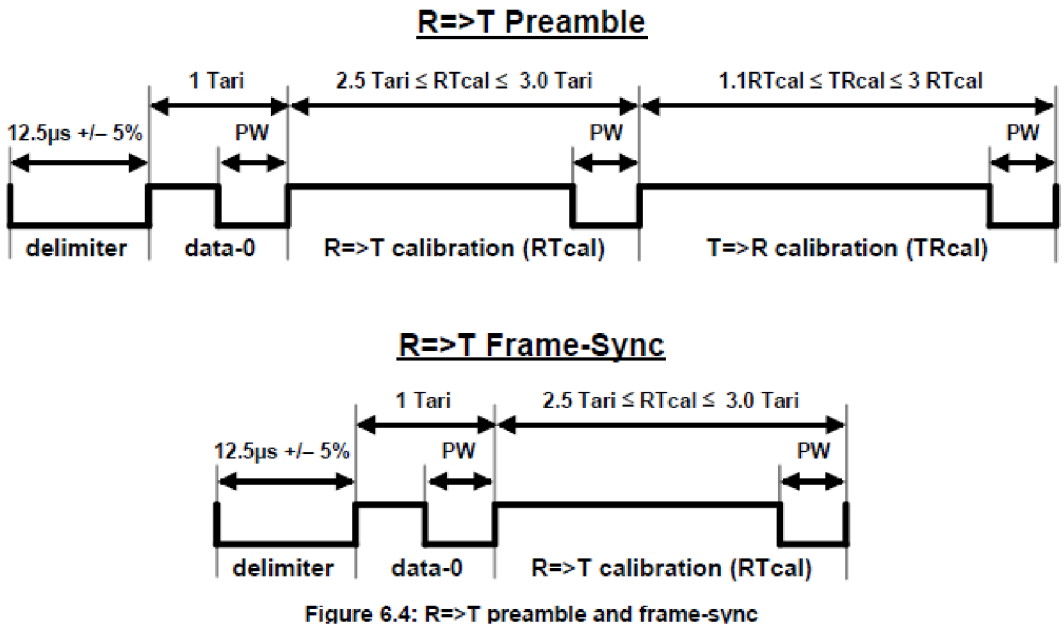
Tari	Parameter	Symbol	Minimum	Nominal	Maximum	Units
6.25 µs to 25 µs	Modulation Depth	(A–B)/A	80	90	100	%
	RF Envelope Ripple	$M_h = M_l$	0		0.05(A–B)	V/m or A/m
	RF Envelope Rise Time	$t_{r,10-90\%}$	0		0.33Tari	µs
	RF Envelope Fall Time	$t_{f,10-90\%}$	0		0.33Tari	µs
	RF Pulsewidth	PW	MAX(0.265Tari, 2)		0.525Tari	µs

Samotná data jsou zakódována pulsně intervalovým kódováním PIE. Sekvence pro data-0 a data-1 jsou ukázány na obrázku 1.4 spolu s definicí časového intervalu Tari. Tento způsob kódování poskytuje tagu dostatek energie pro vykonání požadovaných operací i v případě, že příkaz obsahuje převážně logické nuly (nízká úroveň signálu). Čtečka vysílá s datovou rychlostí 26,7 kbps až 128 kbps (pro náhodná data) v pořadí od MSB po LSB.



Obrázek 1.4 Způsob kódování symbolů 0 a 1 v pulzně intervalovém kódování PIE [3]

Čtečka musí příkazy spadající do skupiny *Query* uvést preambulí, ostatním příkazům předchází Frame-Sync. Jejich požadovaný průběh je ukázán na obrázku 1.5. Čtečka také určuje, zda má tag použít FM0 nebo Millerovo kódování. Po odvysílání příkazů začne čtečka vysílat nemodulovanou nosnou, aby na ni mohl tag namodulovat svou odpověď (viz backscattering v kapitole 1.1).

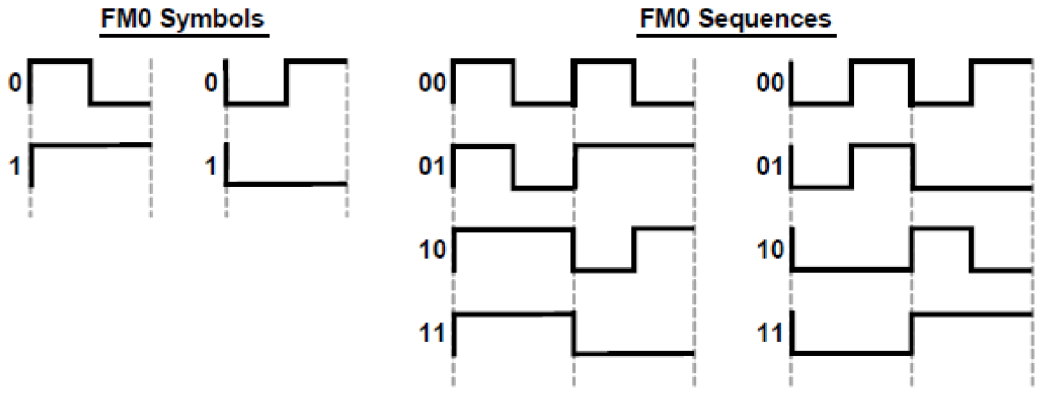


Obrázek 1.5 Časový průběh preambule a Frame-Sync [3]

1.3.3 Požadavky na tag

Pro vysílání odpovědi tag používá backscattering, mění stav své antény (přizpůsobení a vedení nakrátko). Z výroby má tag implementovanou ASK nebo PSK modulaci. V závislosti na požadavku čtečky kóduje data FM0 nebo Millerovým kódováním.

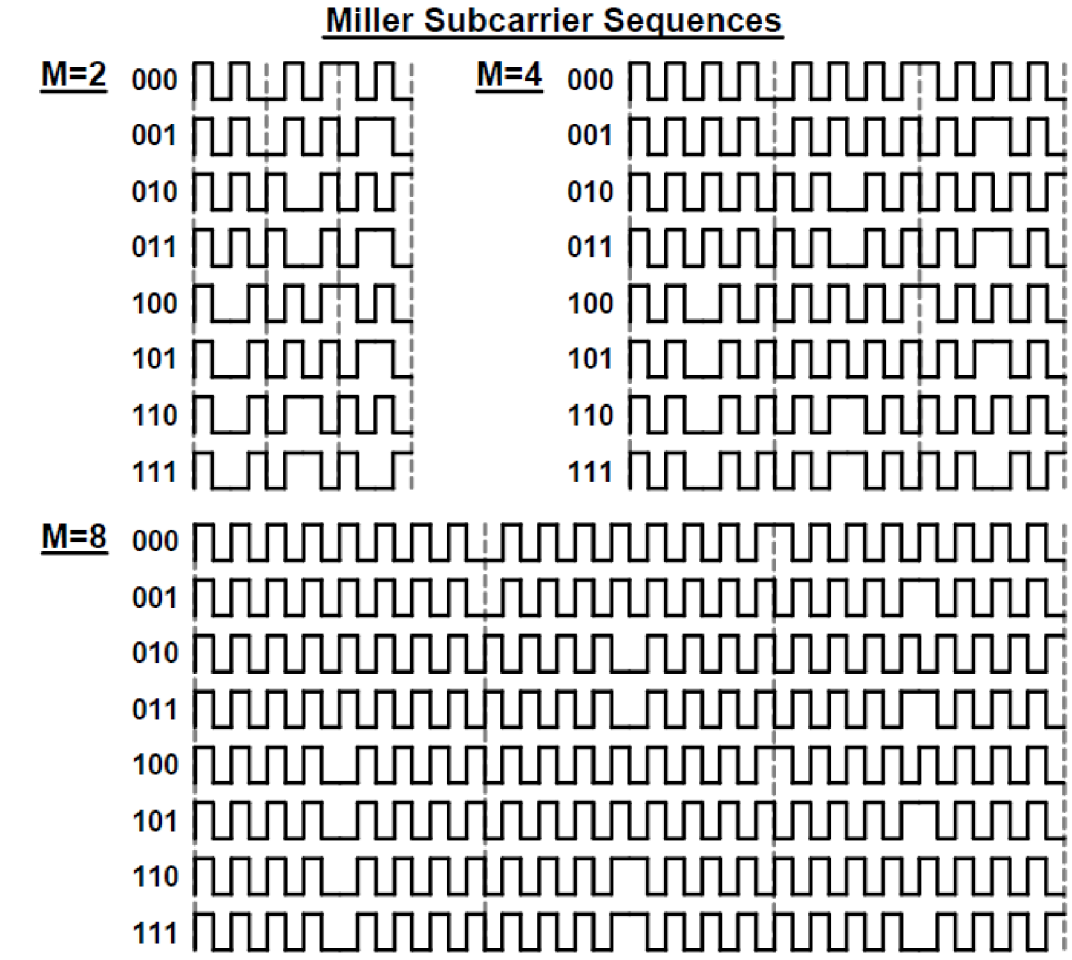
Při kódování FM0 se vždy na rozhraní symbolu mění fáze nemodulovaného nosného signálu, pro symbol 0 nastává další změna fáze uprostřed symbolu, jak je patrno z obrázku 1.6. Vysílání tag začíná FM0 preambulí a ukončuje jej bitem „dummy 1“. Přenosová rychlost u tohoto kódování může být 40 kbps až 640 kbps.



Obrázek 1.6 Symboly a sekvence dat při kódování FM0 [3]

Millerovo kódování přenáší užitečná data na frekvenci podnosné, čímž usnadňuje dekódování odpovědi tagu. Datová sekvence se násobí se zdrojem podnosné v digitální podobě. Po konverzi na analogový signál je tento signál směšován s nosnou a pomocí

zpětného šíření odvysílán směrem ke čtečce. Parametrem Millerova kódování je číslo M nabývající právě jedné z hodnot 2, 4, 8, čímž je určen počet period podnosné na jeden bit dat. Pro kódování dále platí pravidla pro změnu fáze mezi datovými bity, která jsou naznačená na obrázku 1.7. K otočení fáze tedy dojde na rozhraní dvou nulových bitů a uprostřed každého jedničkového bitu. Jelikož frekvence podnosné může nabývat hodnot 40 kHz až 640 kHz, je možno dosáhnout bitové rychlosti od 5 kbps (*M*=8 při 40 kHz) do 320 kbps (*M* = 2 při 640 kHz). Vysílání je opět zahájeno preambulí a ukončeno bitem „dummy 1“.



Obrázek 1.7 Ukázka Millerova kódování tříbitových sekvencí dat [3]

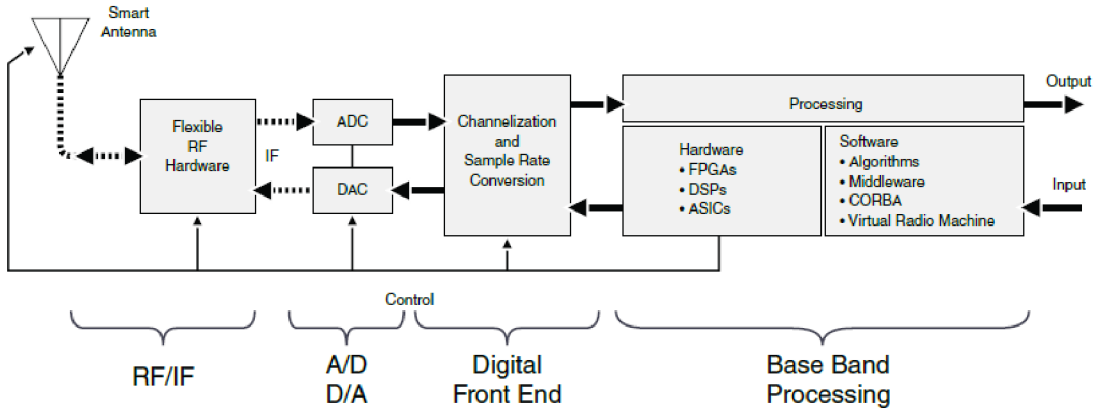
2 USB DVB-T PŘIJÍMAČ JAKO SDR

USB DVB-T přijímač je zařízení původně určené pro příjem pozemní digitální televize na PC. Kromě digitální televize dokážou některé modely přijímat FM rádio nebo digitální rozhlas. K hostitelskému počítači se tento přijímač připojuje přes běžně dostupné rozhraní USB. Za určitých podmínek může pracovat jako softwarové rádio, jehož pořizovací náklady jsou oproti jiným produktům stejného zaměření mnohonásobně nižší. Cena zmiňovaného USB přijímače se pohybuje v rozmezí 10-20 $, zatímco komerční SDR se srovnatelným kmitočtovým rozsahem stojí cca 150 $ (např. Airspy, FUNcube Dongle).

2.1 Softwarové rádio

Softwarové rádio je moderní přístup ke zpracování signálu, kdy se majoritní část zpracovávání uskutečňuje v digitální podobě. Chování a vlastnosti softwarového rádia se také mění softwarově. Od toho se odvíjí velká výhoda softwarového rádia, a tou je snadná přizpůsobivost. Tam, kde bylo dříve nutno fyzicky navrhnout a sestavit nové funkční bloky si softwarové rádio vystačí s pouhým přeprogramováním, což výrazně zjednodušuje implementaci nových postupů, např. přechod na nový komunikační standard atd.

Ideální softwarové rádio by obsahovalo pouze jeden čistě analogový blok – anténu, přijatý signál by byl poté přiveden na vstup AD převodníku a dále zpracováván v číslicové podobě. Podmiňovací způsob je na místě, jelikož umístění AD převodníku hned za anténu je zejména s ohledem na cenu a dosažitelné parametry převodníku nereálné. V praxi se proto signál z antény směšuje do základního pásma nebo do mezifrekvence a až poté je navzorkován (obrázek 2.1). Tímto postupem se sníží nároky na další obvody zpracovávající signál, což se opět promítne do ceny.



Obrázek 2.1 Blokové schéma softwarového rádia současnosti [4]

Celá sekce zajišťující směšování a vzorkování se označuje jako RF front-end. Protože vzorkování je limitováno rychlostí a dynamickým rozsahem AD převodníku, je potřeba signál na vzorkování připravit. Dynamický rozsah je dán poměrem výkonu nejslabšího a nejsilnějšího detekovaného signálu. Aby byly potlačeny nežádoucí silné

signály, bývá na výstup antény připojen filtr. Dále je signál směšován na frekvenci podle vzorkovací frekvence AD převodníku. Maximální směšovací frekvenci f_{IFmax} můžeme určit z rovnice

$$f_{\mathrm{IFmax}} = \frac{f_{\mathrm{vz}}}{2} - B \tag{2.1}$$

kde f_{vz} je vzorkovací frekvence AD převodníku a B je šířka pásma přijímaného signálu. Aby měl signál na vstupu převodníku dostatečnou úroveň výkonu, je zesílen v nízkošumovém zesilovači. Všechny uvedené operace je třeba provádět s co nejmenším zkreslením a minimalizovat vnesení aditivního šumu.

Vzorkovací frekvence je obvykle volena s rezervou vzhledem k nejvyššímu kmitočtu vyskytujícímu se v signálu, aby se snížily požadavky na antialiasingový filtr. Dochází tak k oversamplingu (převzorkování) a vysoký počet vzorků zvyšuje výpočetní nároky na následující obvody. Proto se využívá decimace, při které se sníží počet vzorků, aniž by došlo ke znehodnocení informace aliasingem [5]. Obecně se v obvodech zpracovává signál s různými rychlostmi, jelikož s rychlostními nároky na obvody prudce roste jejich cena.

2.2 Vznik levného SDR

V roce 2010 bylo implementováno softwarové rozhraní, které umožnilo získat z USB DVB-T přijímače surová IQ data, výrobcem původně určená pro demodulaci signálu FM a DAB. Tím byl položen základ pro vznik velmi levného softwarově definovaného rádia. Následně byly skupinou Osmocom vytvořeny ovladače, umožňující radioamatérům, studentům a dalším nadšencům tyto vzorky přijímat a současně ovládat součásti USB přijímače v souladu s nároky kladenými na SDR.

Jako levné softwarové rádio se dnes často používají USB přijímače s integrovaným obvodem Realtek RTL2832U, který funguje jako AD převodník a současně zajišťuje komunikaci s PC přes USB sběrnici (obr. 2.2). USB přijímače, které obsahují jiný chipset než RTL2832U, nejsou kompatibilní s ovladači Osmocomu [6], což výrazně zhoršuje jejich možnosti uplatnění jako SDR. Název podporovaného chipsetu se promítl i do zkratky RTL-SDR, kterou se obvykle označuje softwarové rádio, založené na USB DVB-T přijímači s integrovaným obvodem RTL2832U. Podrobné informace o RTL-SDR přijímačích včetně možných způsobů využití v různých projektech jsou zkompletovány například v publikaci [7].

Obrázek 2.2 Ukázka kompatibilního RTL-SDR přijímače [8]

2.3 Rozbor hardwarové části

Základním prvkem každého RTL-SDR přijímače je tuner a AD převodník RTL2832U. Vysokofrekvenční signál z antény po zesílení v nízkošumovém zesilovači LNA přiveden na vstup tuneru, kde dochází ke směšování a filtrování. Vzniklý signál je navzorkován převodníkem ADC a IQ data jsou odesílána přes rozhraní USB do PC k dalšímu zpracování.

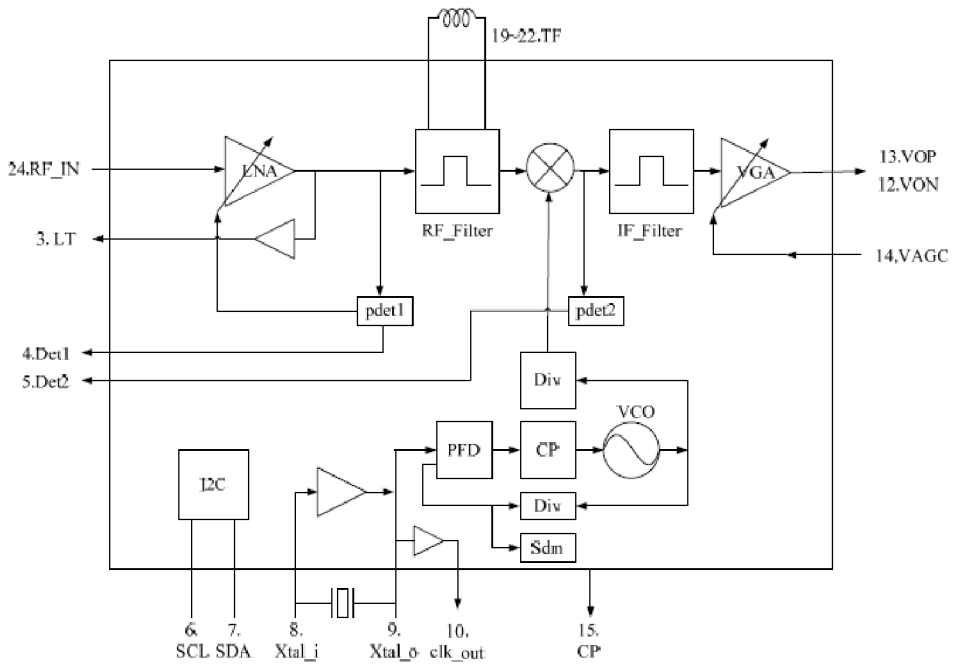
2.3.1 Tuner

Při výběru USB přijímače je nutné znát frekvenci signálu, který chceme analyzovat a podle toho vybrat zařízení s vhodným tunerem. V tabulce 2.1 můžeme najít frekvenční rozsahy dostupných tunerů. Pro naši aplikaci byl vybrán USB přijímač s tunerem Rafael Micro R820T, který splňuje požadavek na frekvenční rozsah pro UHF RFID komunikaci, navíc má nejnižší pořizovací náklady a je snadno dostupný na trhu.

Tabulka 2.1 Frekvenční rozsahy vybraných tunerů [9]

Typ tuneru	Frekvenční rozsah
Elonics E4000	52 - 2200 MHz s mezerou v pásmu 1100-1250 MHz (může se lišit)
Rafael Micro R820T/R820T2/R828D	24 - 1766 MHz
Fitipower FC0013	22 - 1100 MHz
Fitipower FC0012	22 - 948.6 MHz
FCI FC2580	146 - 924 MHz s mezerou v pásmu 308-438 MHz

Na vstup tuneru je přiveden signál z antény, který je zesílen nízkošumovým zesilovačem LNA s šumovým číslem 3,5 dB. Hlavním úkolem tuneru je převést vysokofrekvenční signál na takový kmitočet, který dokáže ADC spolehlivě navzorkovat. Součástí R820T je proto směšovač, jehož výstupní signál má frekvenci f_{IF} = 3,57 MHz. Udávaná chyba lokálního oscilátoru směšovače je přibližně 30 ± 20 ppm [8]. Tento tuner navíc umožňuje softwarově nastavit hodnotu zisku vstupního nebo výstupního zesilovače, případně využít funkce automatického nastavení zisku AGC, a tím dosáhnout optimální hodnoty SNR zkoumaného signálu. Zjednodušené vnitřní zapojení tuneru ukazuje obr. 2.3. Detailní informace je možno nalézt v dokumentaci k R820T, dostupné například na [8]. Ačkoliv výrobcem garantovaný frekvenční rozsah je dle dokumentace pouze 42 až 1002 MHz, výsledky testování v režimu SDR [8, 9, 10, 11] ukazují, že tuner je opravdu schopen dosáhnout na hodnoty uvedené v tabulce 2.1.

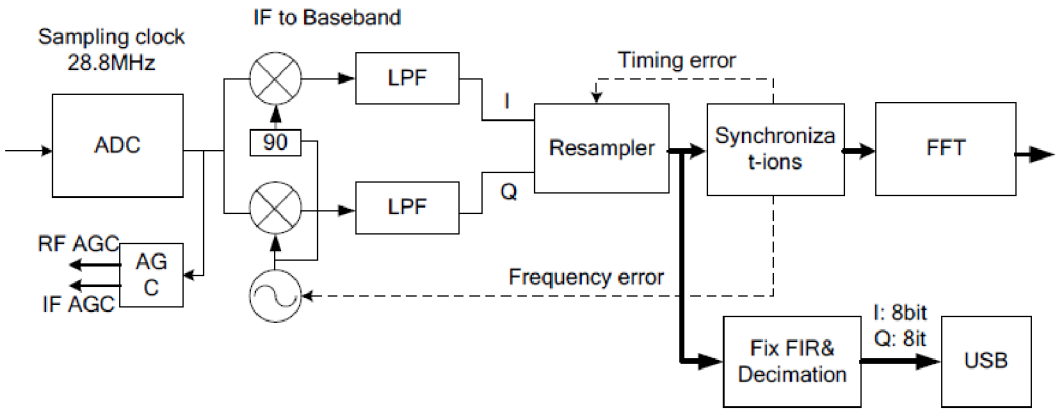


Obrázek 2.3 Zjednodušené blokové schéma R820T, převzato z [8]

2.3.2 RTL2832U

Integrovaný obvod společnosti Realtek vzorkuje a zpracovává signál z tuneru, a odesílá do počítače surová IQ data. Dokumentace k RTL2832U nebyla veřejnosti zpřístupněna, proto jsou poznatky o tomto obvodu získávány spíše pokusnými metodami reverzního inženýrství [8, 9, 10, 11].

Z nich vyplynulo, že RTL2832U obsahuje dva 8bitové AD převodníky, které vzorkují s konstantní rychlostí 28,8 MS/s. Pokud tuner směšuje do základního pásma, pracují oba převodníky současně, při vzorkování signálu směšovaného do mezifrekvence (což je případ tuneru R820T) je v činnosti jen jeden. Navzorkovaný signál je poté v digitálním down-konvertoru DDC převeden do základního pásma. Paralelním směšováním s rozdílem fází 90° získáme kvadraturní signál reprezentovaný tokem I a Q dat. Nežádoucí frekvenční produkty jsou poté odstraněny filtry typu dolní propust a signál je převzorkován. Přenos takto získaných dat po USB zajišťuje jádro 8051. Celý proces zpracování signálu popisuje obr. 2.4. Je potřeba si uvědomit, že IQ data posílaná do PC jsou osmibitová a neznaménková. Výstup je prokládaný, tedy jeden byte I, jeden Q, bez hlavičky nebo jiných metadat [8].



Obrázek 2.4 Blokové schéma RTL2832U [8]

2.4 RTL-SDR software

K USB DVB-T přijímači je obvykle dodáván i software potřebný pro příjem digitální televize. Přiložený ovladač zařízení je optimalizován pro příjem DVB-T a pro účely SDR se nepoužívá. Aby USB přijímač fungoval jako softwarové rádio, je potřeba mít nainstalované ovladače pro USB a vhodný software pro analýzu (SDR#, GNU Radio atd.).

2.4.1 Ovladače a knihovny

Pro potřeby SDR je vhodné mít nainstalovaný obecný ovladač USB. Pro platformu Windows jím je balík WinUSB, který zajišťuje komunikaci po USB sběrnici. Ten zajistí všechny procedury nezbytné k tomu, aby komunikace probíhala podle protokolu USB. Konkrétně se používá ovladač libusb-1.0.dll.

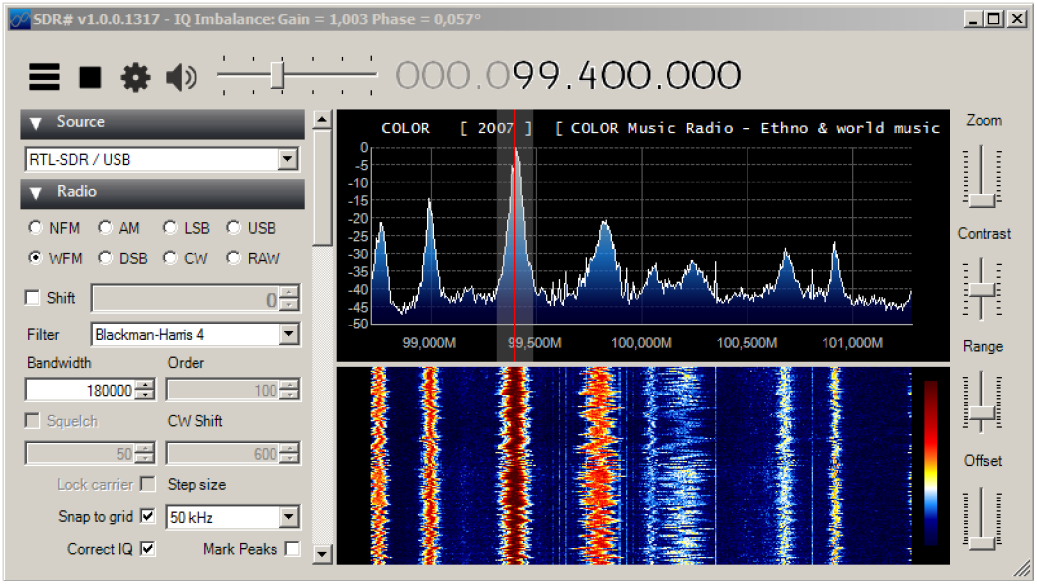
Pro naprogramování vlastní RTL-SDR aplikace se využívá knihovny, vytvořené skupinou Osmocom. Její zdrojový kód je pod označením librtlsdr.c volně přístupný na webu [9], spolu s ostatními zdrojovými kódy. Tato knihovna obsahuje všechny funkce běžně používané pro komunikaci s USB přijímačem. Pro aplikace ve Windows se používá rtlsdr.dll, což je právě knihovna librtlsdr.c zkompilovaná pro Windows, odpovídající verze knihovny pro Linux je dostupná pod názvem librtlsdr.so. Pomocí vhodného nástroje (např. DLL Export Viewer) je možné zobrazit názvy importovaných funkcí knihovny. Jejich název koresponduje s funkcí funkcí, pár příkladů pro ilustraci:

- rtlsdr_open inicializuje zařízení
- rtlsdr_set_sample_rate nastaví vzorkovací rychlost
- rtlsdr_get_tuner_gains vrátí všechny hodnoty zesílení podporované tunerem
- rtlsdr_set_agc_mode aktivuje režim automatického zesílení

Pokud potřebujeme znát také parametry těchto funkcí, lze je opět nalézt v librtlsdr.c.

2.4.2 SDR aplikace

Na stránkách Osmocomu [9] je dostupných několik základních aplikací přímo pro RTL-SDR. Jde spíše o jednoduché skripty s jednou funkcí a pracují s příkazovým řádkem. Dají se ale s výhodou použít při programování rozsáhlejších aplikací. Příkladem takového skriptu je rtl_tcp.exe, který vytvoří TCP spojení s USB přijímačem a umožní tak komunikaci se zařízením přes transportní protokol.



Obrázek 2.5 Ukázka prostředí SDR#

Robustnějším SDR nástrojem umožňujícím rozsáhlou analýzu může být např. SDR#, HDSDR nebo GNU Radio. Tato prostředí už nejsou určena pouze pro RTL-SDR, ale i pro výkonnější hardware jako je AIRSPY, HackRF a jiné. Jak je vidět na obr. 2.5, v prostředí SDR# se v reálném čase zobrazuje spektrum přijímaného signálu spolu se spektrogramem. Signál je demodulován podle nastaveného typu demodulace (AM, FM, DSB atd.) a digitálně filtrován.

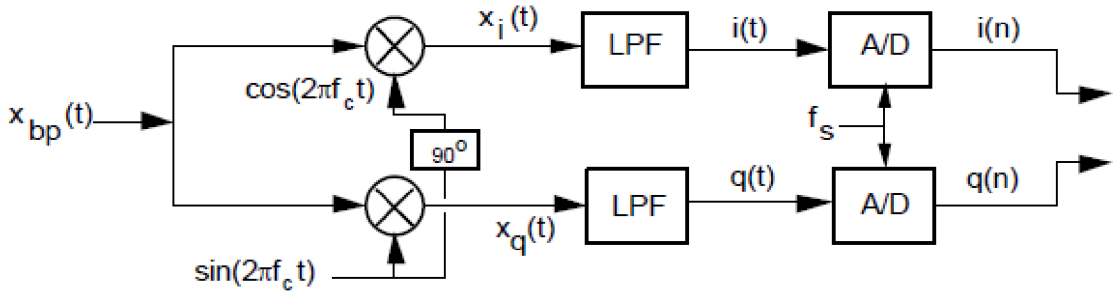
3 ANALÝZA RFID SIGNÁLŮ

Pro úspěšné naprogramování analyzátoru je nezbytné znát povahu analyzovaných signálů. Při vyvíjení aplikace je potřeba brát ohled na specifika RFID signálů a omezené možnosti přijímače. Jelikož výstupem vybraného USB přijímače jsou IQ data, bude jim v úvodu věnována pozornost. Cílem této kapitoly je shrnout základní typy analýzy a posoudit proveditelnost těchto analýz pro RFID signál přijímaný USB DVB-T přijímačem.

3.1 IQ signál

Signál, jehož okamžitá hodnota je v daný časový okamžik určena jedním komplexním číslem se nazývá IQ signál. I symbolizuje reálnou část (*in-phase*) a Q imaginární část (*quadrature*) komplexního čísla. IQ signály (někdy také označované jako komplexní signály) hrají důležitou roli ve velkém množství dnešních aplikací, zaměřených na digitální zpracování signálu.

Postup pro převedení reálného spojitého signálu na IQ data je naznačen na obr. 3.1. Mimo jiné vidíme, že komplexní signál tvoří dvě sinusoidy o stejné frekvenci s relativním fázovým posuvem 90°. Komplexní signál tedy není žádnou matematickou abstrakcí, můžeme jej zobrazit například na osciloskopu ve vhodném režimu. Poněvadž směšováním reálného signálu vznikají nežádoucí frekvenční produkty, je potřeba vstup AD převodníku filtrovat.



Obrázek 3.1 Blokové schéma postupu pro získání IQ dat [12]

Výše uvedený postup se označuje také jako kvadraturní vzorkování. Získaný komplexní signál je tedy reprezentován vzorky tvořenými komplexními čísly ve složkovém tvaru. Ačkoliv složkové vyjádření IQ signálu se zdá na první pohled méně intuitivní než polární vyjádření pomocí modulu a fáze, s ohledem na obvodovou realizaci je výhodnější pracovat s I a Q signály. Pokud bychom chtěli měnit fázi nosné přímo, lze s pomocí trigonometrické věty 3.1 odvodit, že by obvod musel realizovat funkci danou rovnicí 3.2.

$$\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta) \tag{3.1}$$

$$M\cos(2\pi f_c t + \varphi) = M\cos(2\pi f_c t)\cos(\varphi) - M\sin(2\pi f_c t)\sin(\varphi) \tag{3.2}$$

Jelikož přesná změna fáze vysokofrekvenční nosné v závislosti na vstupním signálu je složitě dosažitelná, lze s výhodou použít IQ signál. Jak ukazuje rovnice 3.5, fázi nosné je také možné měnit změnou amplitudy I a Q signálu, což je v obvodech mnohem snadněji realizovatelná funkce. Používání IQ signálu tedy usnadňuje modulaci i demodulaci RF signálu. Pro složky I a Q platí, že:

$$I = M\cos(\varphi) = M\cos(2\pi f_c t) \tag{3.3}$$

$$Q = M\sin(\varphi) = M\sin(2\pi f_c t) \tag{3.4}$$

S využitím věty 3.2 a vztahů 3.3 a 3.4 dojdeme k rovnici:

$$M\cos(2\pi f_c t + \varphi) = I\cos(2\pi f_c t) - Q\cos(2\pi f_c t) \tag{3.5}$$

Charakteristickým znakem komplexního signálu je jednostranné spektrum. Tento jev lze lehce objasnit, pokud zapíšeme komplexní číslo v goniometrickém tvaru $z = \mathrm{M}[\cos(2\pi f_0 t) + \mathrm{j}\sin(2\pi f_0 t)]$. Reálné funkce sinus a kosinus mohou být vyjádřeny součtem dvou komplexních exponenciál (rovnice 3.7 a 3.8), z nichž každá odpovídá jedné spektrální čáře.

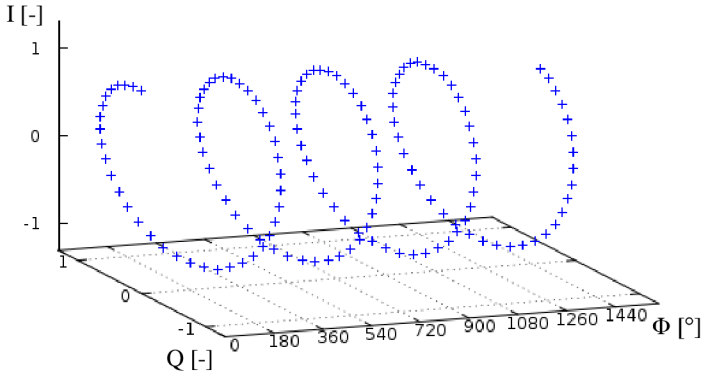
$$\cos(2\pi f_0 t) = \frac{e^{\mathrm{j}2\pi f_0 t}}{2} + \frac{e^{-\mathrm{j}2\pi f_0 t}}{2} \tag{3.7}$$

$$\sin(2\pi f_0 t) = \frac{e^{\mathrm{j}2\pi f_0 t}}{2\mathrm{j}} - \frac{e^{-\mathrm{j}2\pi f_0 t}}{2\mathrm{j}} \tag{3.8}$$

Po dosazení do goniometrického tvaru komplexního čísla dostaneme exponenciální tvar $z = M \cdot e^{\mathrm{j}2\pi f_0 t}$. Je tedy zjevné, že zatímco spektrum reálného signálu je oboustranné a souměrné podle vertikální osy, komplexní signál má jednostranné spektrum.

3.2 Časová analýza

Pokud je přijímaný IQ signál digitální, můžeme jej na původní podobu rekonstruovat interpolací a získat tak průběh ve spojitém čase. IQ signál nese zároveň informaci o modulu i fázi signálu v čase, proto se zobrazuje v trojrozměrném prostoru (obr. 3.2). Průběh jednotlivých signálů I a Q získáme promítnutím do roviny kolmé na imaginární osu (složka I), respektive kolmé na reálnou osu (složka Q).



Obrázek 3.2 Příklad časového průběhu diskrétního IQ signálu [13]

Zobrazení v 3D grafu může klást větší nároky na výpočetní výkon, navíc v mnoha případech není nezbytně nutné. S využitím jednoduchých vztahů 3.9 a 3.10 dokážeme z IQ signálu extrahovat informaci o modulu i fázi signálu.

$$M = |z| = \sqrt{I^2 + Q^2} \tag{3.9}$$

$$\varphi = \arg(z) = \tan^{-1}\left(\frac{Q}{I}\right) \tag{3.10}$$

Často je potřeba určit výkon vstupního signálu. Výkon je přímo úměrný druhé mocnině velikosti vstupního signálu. Za předpokladu, že konstantou úměrnosti je jednička můžeme psát:

$$P = I^2 + Q^2 \tag{3.11}$$

Spojení analýzy v časové a frekvenční oblasti se označuje zkratkou JTFA (Joint Time-Frequency Analysis). Tento druh analýzy umožňuje zachytit průběh změn ve spektru, proto má v oblasti softwarového rádia značné využití. Nejčastěji se zobrazuje ve formě spektrogramu, kde se intenzita pole pro daný čas a frekvenci odlišuje barevně (ukázka spektrogramu je např. na obr. 2.4).

3.3 Spektrální analýza

Fourierova transformace je nástrojem, který dokáže vyjádřit časový průběh jisté veličiny pomocí spektrálních složek. Pro číslicové zpracování signálu je důležitá zejména diskrétní Fourierova transformace (DFT). Výpočet DFT přímo z definiční rovnice je ovšem extrémně neefektivní, proto se v praxi používá rychlá Fourierova transformace (FFT). Objev algoritmu FFT znamenal velký přelom v oboru analýzy signálu. Ve spojení s vysokým výpočetním výkonem dnešních procesorů umožňuje používání operací, které by jinak nebyly nerealizovatelné.

3.3.1 DFT

Diskrétní Fourierova transformace umožňuje přechod z časové do frekvenční oblasti

pro vstupní diskrétní signál konečné délky. Výsledkem DFT je opět posloupnost diskrétních hodnot. Diskrétní Fourierova transformace je definována rovnicí 3.12. Pokud tuto rovnici upravíme do tvaru 3.13, lze si všimnout, že DFT je založena na principu hledání korelace mezi vstupním signálem a komplexní sinusoidou definovanou vztahem $\cos\varphi - j\sin\varphi$. Argument komplexní sinusoidy φ určuje analyzovanou frekvenci ve vstupním signálu. Pokud se v něm vyskytuje, je výsledkem korelace na dané frekvenci nenulová hodnota.

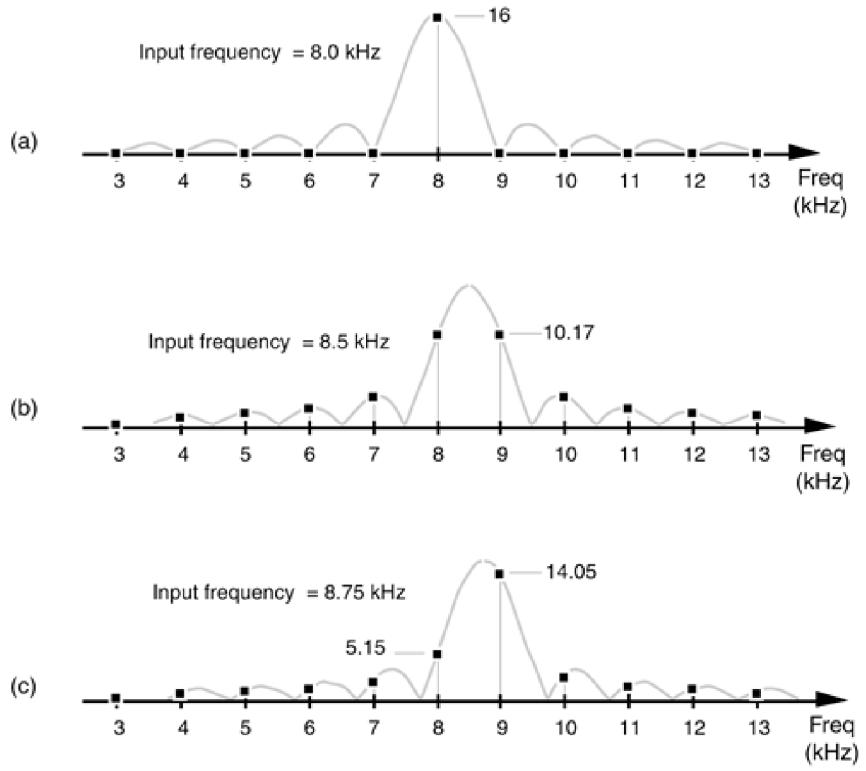
$$X(m) = \sum_{n=0}^{N-1} \mathrm{x(n)e^{-j2\pi nm/N}} \tag{3.12}$$

$$X(m) = \sum_{n=0}^{N-1} \mathrm{x(n)}\left(\cos\frac{2\pi nm}{N} - j\sin\frac{2\pi nm}{N}\right) \tag{3.13}$$

Počet výstupních vzorků DFT je roven počtu vstupních vzorků *N*. Přesné hodnoty frekvence komplexní sinusoidy ve vztahu 3.8 závisí na počtu vzorků *N* a vzorkovací frekvenci f_s. Výpočet korelace potom probíhá pro základní frekvenci f_s/N a její násobky z intervalu 0 – (*N*-1). Tato základní frekvence tedy určuje rozlišovací schopnost DFT.

Protože DFT pracuje se vstupním signálem konečné délky, je potřeba nekonečný vstupní signál nejprve omezit. Toho můžeme dosáhnout vynásobením v časové oblasti s takzvanou okénkovou funkcí. Nejjednodušší okénkovou funkcí je obdélníkové okno, které nabývá hodnoty 1 v úseku „okna“ a mimo něj hodnoty 0, tudíž dokáže přesně vybrat požadovanou část časového průběhu. Tato operace se ovšem projeví i ve spektru původního signálu. Násobení v časové oblasti je ekvivalentní s konvolucí ve frekvenční oblasti, proto bude nový tvar spektra dán konvolucí spektra původního signálu s frekvenční charakteristikou okénkové funkce. Jak již bylo řečeno, během DFT dochází ke vzorkování spektra je na násobcích základní frekvence f_s / N. Vlivem okénkové funkce tedy může dojít k velkému zkreslení výsledku. Příklad zkreslení je pro kladnou frekvenci ukázán na obrázku 3.3. Rozlišení DFT zde je 1 kHz, vstupní signál je harmonický.

Z obrázku 3.3 je patrné, že ke zkreslení výsledného spektra nedojde pouze v případě, kdy je frekvence vstupního harmonického signálu celočíselným násobkem základní frekvence DFT. V ostatních případech budou do spektra prosakovat i jiné frekvenční složky, které se v původním signálu vůbec nevyskytovaly. Tento jev se označuje jako DFT únik. Jeho vliv se dá omezit použitím jiné okénkové funkce, než obdélníkové. V praxi se nejčastěji využívá Hammingovo nebo Hanningovo okno, které mají tvar podobný trojúhelníkovému oknu. Frekvenční charakteristiky těchto funkcí mají stejně jako obdélníkové okno tvar sinc, ovšem úroveň postranních laloků je výrazně nižší.



Obrázek 3.3 Výsledek DFT pro harmonický signál konečné délky s frekvencí a) 8 kHz, b) 8,5 kHz, c) 8,75 kHz [12]

3.3.2 FFT

Rychlá Fourierova transformace (FFT) je algoritmus, který umožňuje snížit standardní časovou náročnost DFT (N^2 operací násobení komplexních čísel) na přibližně

$$\frac{N}{2}\log_2 N. \tag{3.14}$$

Během počítání DFT klasickým způsobem se některé operace se stejnými čísly provádí mnohokrát. Ve výpočtu FFT jsou tyto nadbytečné operace eliminovány, čímž se sníží časová náročnost. Způsobů eliminace je více, v současné době je nejpoužívanějším algoritmem radix-2. Je nutné poznamenat, že na přesnost výsledku nemá toto urychlení vliv. FFT tedy není pouze aproximací DFT, je její identitou. Proto musíme při výpočtu FFT brát v úvahu stejné jevy, které nastávají při výpočtu DFT (např. DFT únik).

Algoritmus radix-2 vyžaduje, aby počet vstupních vzorků byl mocninou dvou. Pokud nemáme k dispozici přesně 2^n vzorků, je obecně výhodnější sekvenci doplnit nulami, než některé vzorky ignorovat. Pokud používáme okénkovou funkci, je potřeba doplnit sekvenci nulami až po vynásobení oknem.

3.4 Specifika RFID signálů

Jedním z hlavních parametrů RFID signálu je přenosová rychlost. Ta společně se způsobem kódování určuje, s jakou frekvencí musíme demodulovaný signál vzorkovat, abychom byli schopni přesně dekódovat data. V protokolu Gen2 se používá několik

přenosových rychlostí (viz kapitola 1.3). Pro zabezpečení správné funkčnosti analyzátoru je potřeba provést analýzu pro nejhorší možný případ, tedy největší přenosovou rychlost.

Maximální rychlost vysílání čtečky je pro náhodná data 128 kbps (kap. 1.3.2). Bity jsou kódovány kódováním PIE, kde bit 0 má délku časového intervalu Tari. Nulový bit je v podstatě impulsem s definovanou šířkou. Typicky je šířka impulsu rovna Tari/2, ovšem podle standardu Gen2 musí být minimálně 2 µs. Tato doba je nejmenší časovou změnou, kterou můžeme v demodulovaném signálu očekávat. Aby ji analyzátor dokázal zaznamenat, musí za dobu 2 µs přečíst alespoň dva vzorky. To odpovídá vzorkovací frekvenci 1 Msps.

Odpověď tagu je kódována FM0 nebo Millerovým kódováním. Maximální bitová rychlost respektive frekvence podnosné je v obou případech 640 kbps. Nejhorší případ nastane u kódování FM0, kde je úroveň signálu invertována uprostřed každého nulového bitu. Nejmenší délka konstantní části signálu je tudíž 0,78 µs. Aby během této doby USB přijímač získal dva vzorky, musí vzorkovat s frekvencí 2,56 Msps.

Ačkoliv dokáže USB přijímač odesílat až 3,2 miliónů vzorků za sekundu, největší rychlost, při které ještě nedochází k náhodným ztrátám vzorků je 2,56 Msps [9]. Je tedy jasné, že z tohoto hlediska je možné USB přijímač bez výhrad použít pro dekódování vysílání čtečky. Rozluštění signálu tagu může být při maximální přenosové rychlosti obtížné až nemožné, zvlášť pokud uvažujeme reálný signál, který má konečnou strmost hran a jistý rozptyl vzhledem k ideálním parametrům.

Pro analýzu je také důležité rozlišení na vertikální, závislé ose. Toto rozlišení odpovídá počtu bitů na vzorek. Jak bylo popsáno v kapitole 2.3.2, výstupem vybraného USB přijímače jsou 8bitová IQ data. Za předpokladu, že IQ data reprezentují amplitudu signálu, můžeme určit dynamický rozsah DR v decibelech

$$DR = 20\log(2^b - 1) \tag{3.15}$$

kde b je počet bitů, vyhrazených pro vyjádření velikosti hodnoty. Pro náš USB přijímač $b = 7$, poněvadž osmý bit je znaménkový. Dynamický rozsah USB přijímače je tedy přibližně 42 dB. Dynamický rozsah signálu určuje rozdíl úrovní mezi nejsilnějším a nejslabším signálem. USB přijímač má implementovány AGC obvody, které zaručují, že velikost nejsilnějšího přijímaného signálu bude vyjádřena maximální bitovou hodnotou. Aby byl přijímač dokázal detekovat signál, musí mít přijímač dynamický rozsah větší než samotný signál. Popisovaný USB přijímač je tedy schopen zachytit signály, které mají o 42 dB nižší úroveň, než nejsilnější signál ve spektru.

Pro RFID komunikaci je typický velký rozdíl mezi úrovněmi signálu. Zatímco čtečka vysílá s relativně velkým výkonem a hloubkou modulace min. 80 %, tag odpovídá pomocí backscatteringu, tudíž s výkonem velmi malým. Vzhledem k dynamickému rozsahu USB přijímače se dá předpokládat, že se s daným hardwarem podaří dekódovat pouze příkazy čtečky.

4 KOMUNIKACE S EXTERNÍM HARDWAREM V LABVIEW

LabVIEW je vývojové prostředí společnosti National Instruments, jež funguje na principu virtuální instrumentace. Hlavní výhodou je abstrakce zhusta používaných funkcí, díky čemuž vyžaduje práce v tomto prostředí méně čistě „programátorských" znalostí. Uživatel se tak může plně soustředit na technické aspekty svého projektu. Prostředí LabVIEW se někdy označuje jako G-jazyk. Označení vyplývá z faktu, že převážná část programování v tomto jazyce se odehrává v grafické podobě. Podrobné informace o prostředí LabVIEW lze nalézt na stránkách National Instruments.

Pro aplikaci analyzátoru RFID signálů bylo prostředí LabVIEW vybráno především kvůli velkým možnostem v oblasti zpracování signálů. Díky dostupnosti toolkitů jsou operace jako filtrování, transformace z časové do frekvenční oblasti, spektrální analýza a mnoho dalších velmi intuitivní. Je potřeba vzít v úvahu také fakt, že ačkoliv je licence k vývojovému prostředí LabVIEW placená, v něm vytvořené programy lze spouštět i na počítačích, kde je instalován pouze LabVIEW Run-Time Engine, který je zdarma.

LabVIEW se vyznačuje velkou podporou uživatelských aplikací, zaměřených na sběr dat z externích zařízení a jejich ovládání. Obsahuje sadu VI, které jsou pro tento účel předpřipraveny a které z pohledu uživatele celý proces významně zjednodušují. Společnost National Instruments je rovněž výrobcem hardwaru různého zaměření, takže kompatibilita s LabVIEW je zaručena. Základní možnosti pro komunikaci s externími zařízeními popisuje následující text. Pozornost je věnována především nástrojům, jejichž použití připadá v úvahu pro signálový analyzátor na bázi RTL-SDR.

4.1 NI-VISA

VISA je standard, který si klade za cíl sjednotit specifikace pro vstupně-výstupní zařízení a vytvořit komunikační software, který bude spolehlivě fungovat pro přístroje různých výrobců. Velkou výhodou je také nezávislost na typu sběrnice, platformě a vývojovém prostředí. Mezi společnostmi podporujícími tento standard je např. Tektronix, Hewlett-Packard a National Instruments.

V LabVIEW existuje paleta funkcí obsahující terminály pro komunikaci pomocí standardu VISA. Nejprve je ale třeba mít pro dané zařízení nainstalovaný ovladač, který dokáže s tímto standardem pracovat. V případě, že výrobce tento ovladač nedodává (což je případ USB DVB-T přijímače), je možnost ovladač vytvořit pomocí nástroje NI-VISA Driver Wizard. Pokud je zařízení připojeno přes USB, přidělí mu Driver Wizard třídu USB INSTR nebo USB RAW. Do třídy USB INSTR spadají ta zařízení, která používají protokol IEEE-488.2. V této třídě je možné přímo používat funkce VISA Read, VISA Write atd. U zařízení třídy USB RAW toto možné není, protože tato zařízení mají svůj vlastní komunikační protokol [14].

Bylo zjištěno, že USB přijímač, který je použit v této práci, patří do třídy USB RAW. Vzhledem k absenci dokumentace USB přijímače nelze jednoduše zjistit sadu příkazů, kterou používá. Tento způsob tudíž není vhodný pro implementaci

komunikačního rozhraní mezi USB přijímačem a LabVIEW.

4.2 TCP spojení

Protokol TCP umožňuje navázání spojení mezi dvěma aplikacemi. Mezi základní rysy tohoto protokolu patří bezztrátovost (ztracená data jsou vyžádána znovu) a garance doručování ve správném pořadí. Při TCP spojení se vytvoří virtuální okruh, který je duplexní, tj. data se mohou přenášet nezávisle oběma směry.

LabVIEW obsahuje pro práci s TCP několik funkcí. Spojení vytvoří funkce TCP Open Connection, a dále předává ID tohoto spojení jako referenci. Funkce TCP Read a TCP Write umožňují přijímat a posílat data v rámci daného spojení. Spojení je nutné ukončit funkcí TCP Close Connection. Přijímaná i odesílaná dat jsou ve formátu string.

Jak již bylo uvedeno, pro streamování IQ dat z USB přijímače přes TCP je možné použít skript rtl_tcp.exe. Konfigurace USB přijímače se zde provádí zapsáním čísla příkazu a hodnoty, kterou mu chceme přiřadit. Přesná čísla příkazů lze zjistit ze zdrojového kódu (soubor rtl_tcp.c dostupný na [9]). Například příkaz nastavení vzorkovací rychlosti má číslo 2, za ním následuje hodnota rychlosti ve vzorcích za sekundu.

Pro aplikaci signálového analyzátoru by tedy bylo možné spojení TCP využít, jelikož je podporováno jak ze strany LabVIEW, tak i USB přijímačem. Rtl_tcp.exe zde pouze zprostředkovává přenos dat mezi USB a LabVIEW, lze jej tedy v určitém smyslu považovat za nadbytečnou zátěž pro operační systém. Z tohoto hlediska se jako efektivnější možnost jeví použití sdílené knihovny funkcí.

4.3 Sdílené knihovny DLL

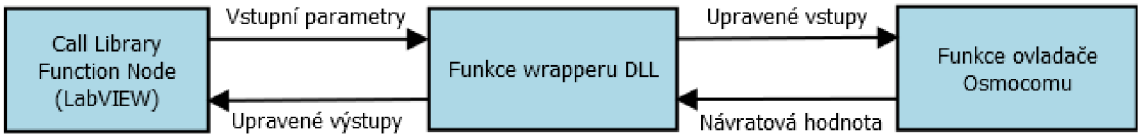
Sdílené knihovny uchovávají zdrojové kódy funkcí a procedur, aby je bylo možné použít v různých počítačových programech. Funkce statické knihovny jsou při kompilaci přidány do výsledného spustitelného souboru, tudíž pro pozdější spouštění souboru není přítomnost statické knihovny nutná. Naproti tomu, pokud používáme ve zdrojovém kódu funkce dynamické knihovny, ke spustitelnému souboru se připojí pouze odkazy na dané funkce. Pro chod programu je tedy třeba mít k dispozici celou dynamickou knihovnu. Dynamické knihovny mají ve Windows příponu .dll.

Jak bylo zmíněno v kapitole 2.4, pro práci s RTL-SDR přijímačem je k dispozici knihovna rtlsdr.dll. Obsahuje všechny funkce potřebné k tomu, aby bylo možné použít RTL-SDR přijímač jako softwarové rádio. Prostředí LabVIEW umožňuje importovat funkce dynamické knihovny a následně je volat pomocí Call Library Function Node (dále jen CLFN). Vstupní a výstupní parametry tohoto nodu jsou totožné se vstupy a výstupy importované funkce.

4.3.1 Princip wrapperu DLL knihovny

Jelikož definice datových typů v jazyce C a LabVIEW se může lišit, je nutné zajistit jejich správnou konverzi. K tomu se používají takzvané wrappery DLL, jejichž úkolem je zprostředkovat kompatibilitu mezi knihovnou v jazyce C a LabVIEW.

Wrapper musí být vytvořen tak, aby byl schopen převzít jako parametry datové typy a hodnoty, které jsou funkci předány z LabVIEW. Pokud je to nutné, musí poté tyto data zkonvertovat do podoby, v jaké je očekává ovladač, v tomto případě knihovna rtlsdr.dll. Teprve pak zavolá funkce wrapperu odpovídající funkci ovladače. Pokud má funkce ovladače návratovou hodnotu, musí být tato hodnota wrapperem převedena do podoby, vhodné pro LabVIEW. Celou situaci při volání jedné funkce ilustruje obr. 4.1.



Obrázek 4.1 Princip wrapperu DLL

4.3.2 Kompatibilita mezi C a LabVIEW

Pro jednodušší datové typy ovšem není nutné ve wrapperu provádět výše zmiňovanou konverzi, protože tyto typy jsou v C i LabVIEW shodné. Konkrétně není nutné konvertovat následující datové typy:

- signed/unsigned integer 8, 16, 32 nebo 64-bit
- signed/unsigned pointer-sized integer
- čísla s pohyblivou řádovou čárkou (single, double)
- ukazatele na výše uvedené datové typy
- pole (i vícerozměrné) sestávající z celočíselných typů nebo z čísel s pohyblivou řádovou čárkou
- textový řetězec

Při předávání celočíselných datových typů je stačí přesně specifikovat (např. unsigned integer 32-bit). Pokud je potřeba předat pole, předává se ukazatel na toto pole. Textové řetězce v LabVIEW obsahují v prvních čtyřech bitech informaci o délce řetězce, zatímco v C se konec řetězce označí hodnotou NULL. Proto je nutné při předávání textových řetězců mezi LabVIEW a C vybrat v CLFN formát „C String Pointer“. Typ „Adapt to Type“ umožňuje předávat jednoduchou strukturu, kterou sestaví ze vstupních dat, je ovšem nezbytné připojit terminály v odpovídajícím pořadí.

5 WRAPPER DLL

Tato kapitola se zabývá popisem wrapperu DLL, vytvořeného pro účely zprostředkování komunikace mezi LabVIEW a USB DVB-T přijímačem. Z možností, popsaných v kapitole 4 byla tato možnost vybrána jako nejvhodnější. Vytvořený wrapper obsahuje stejné funkce, jako knihovna rtlsdr.dll, ovšem jeho funkce už je možné volat z LabVIEW přímo.

Funkce ovladače Osmocomu lze rozdělit do čtyř skupin: inicializační, konfigurační, funkce pro čtení, ukončovací. Pro správnou funkci zařízení je nutné, aby byla na začátku aplikace zavolána funkce pro inicializaci a před ukončením aplikace ukončovací funkce. Kompletní seznam implementovaných funkcí se stručným popisem činnosti a parametrů je možné nalézt v hlavičkovém souboru rtlsdr_wrap.h na přiloženém CD. Zdrojový kód wrapperu zde nese název rtlsdr_wrap.c.

5.1 Inicializace

Inicializaci zařízení zajišťuje funkce *rtlsdr_open*, která vytvoří pro připojené zařízení strukturu o mnoha proměnných různých datových typech, z nichž některé jsou ještě dále strukturované. Parametrem této funkce je ukazatel na ukazatel na tuto strukturu, ostatní funkce ovladače potom přebírají ukazatel na strukturu. Předávat tuto strukturu do LabVIEW by bylo značně komplikované, proto zde byl použit jiný přístup. Paměťový prostor pro tuto strukturu se alokuje až ve funkci wrapperu a do LabVIEW se vrací pouze ukazatel na tuto strukturu, přetypovaný na integer. Aby zůstala data ve struktuře platná i po ukončení inicializační funkce, byla využita dynamická alokace paměti, čímž jsme se vyhnuli použití globální proměnné. Tělo této funkce wrapperu je ukázáno v následujícím zdrojovém kódu:

```
DLL_EXPORT uintptr_t rtlsdr_open_wrap(void){

    uint32_t index = 0;

    rtlsdr_dev_t *address;

    /* allocate memory for structure */
    address = (rtlsdr_dev_t *)malloc(sizeof(rtlsdr_dev_t));


    if (!address)
        return ERRNOMEM;            // failed to allocate memory

    rtlsdr_open(&address,index);    // call Osmocom function

    if (!address)
        return 0;                   // failed to open session to device

    return (uintptr_t)(address);    // return device handle

}
```

Pro ukončení práce s USB přijímačem slouží funkce *rtlsdr_close_wrap*, která vypne demodulátor a AD převodník, ukončí USB komunikaci a dealokuje paměť.

5.2 Konfigurace

Všechny konfigurační funkce wrapperu přijímají jako parametr handle zařízení, vytvořený funkcí *rtlsdr_open_wrap*, a přetypovávají jej zpět na ukazatel. Funkce wrapperu mimo jiné umožňují:

- naladění na zadanou frekvenci, s možností kompenzace nepřesnosti oscilátoru
- ovládání zisku tuneru R820T
- nastavení dostupné vzorkovací rychlosti
- použití interního AGC AD převodníku

Po ukončení konfigurace USB přijímače je vyžadováno, aby byly resetovány buffery zavoláním funkce *rtlsdr_reset_buffer_wrap*. Teprve poté je možné spustit funkce pro čtení. Pro ilustraci je níže ukázán zdrojový kód funkce wrapperu pro nastavení frekvence lokálního oscilátoru USB DVB-T přijímače. Tato funkce je už přímo volána z LabVIEW pomocí CLFN, jehož parametrem je handle zařízení a hodnota nastavované frekvence v Hz.

```
DLL_EXPORT  uint32_t  rtlsdr_set_center_freq_wrap(uintptr_t  address,
uint32_t freq){

    uint32_t r;
    rtlsdr_dev_t *ptr;

    ptr = (rtlsdr_dev_t *)address; // typecast integer to pointer

    /* Osmocom dll function */
    r=rtlsdr_set_center_freq(ptr, freq);

    return r;

}
```

5.3 Funkce pro čtení

Původní ovladač Osmocomu obsahuje funkce jak pro synchronní, tak i pro asynchronní čtení. Oba způsoby sběru dat byly implementovány do wrapperu a jsou popsány v následující části. Za jistý druh funkcí pro čtení lze také považovat funkce, které slouží ke zjištění stavu zařízení, např. ověření nastavené vzorkovací rychlosti nebo zjištění dostupných zesílení tuneru. Princip těchto funkcí je velmi jednoduchý a lze jej snadno pochopit z přiloženého zdrojového kódu, proto nejsou v tomto textu blíže popsány.

5.3.1 Synchronní čtení

V módu synchronního čtení jsou IQ data ukládána do pole, inicializovaného v LabVIEW na požadovanou délku. Po spuštění funkce synchronního čtení jsou IQ data posílána přes USB zvolenou vzorkovací rychlostí a ukládána do pole. IQ data jsou prokládaná, tzn. jeden vzorek I a poté jeden Q, každý z nich je osmibitový a neznaménkový. Při zpracování signálu je proto nutné odečíst od všech vzorků konstantu 127. Co se týče implementace, synchronní čtení je výrazně jednodušší. Aby

bylo možné číst data z přijímače nepřetržitě, musí být volání této funkce v LabVIEW realizováno ve smyčce. Zřejmou nevýhodou tohoto přístupu je fakt, že v době mezi opakovaným spouštěním funkce synchronního čtení dochází ke krátkým prodlevám, během kterých USB přijímač nezaznamenává signál, což by zřejmě bránilo použití tohoto analyzátoru pro náročnější aplikace.

5.3.2 Asynchronní čtení

Výše uvedené nevýhody synchronního čtení řeší čtení asynchronní, které bylo do wrapperu implemetováno nad rámec zadání. Při tomto způsobu sběru dat lze obecně říci, že nedochází k výpadkům v příjmu, protože přijímaná data jsou interně bufferována. Zpracování takto přijímaných dat může probíhat už během sběru dat (např. ukládání do souboru) nebo později v běhu programu čtením dat z bufferu. Sběr dat může být poté zastaven na základě nějaké vnější události.

Ovladač Osmocomu obsahuje funkci pro asynchronní čtení, jedním z parametrů této funkce je i ukazatel na callback funkci. Samotná callback funkce už v ovladači není, ta musí být implementována až v hostitelské aplikaci. Protože prostředí LabVIEW neumožňuje předat ukazatel na VI pro použití jako callback funkci, musí být callback implementován už ve wrapperu. Úkolem callbacku je ukládat přijatá IQ data do globálního bufferu, dokud není splněna podmínka pro ukončení. Je tedy zřejmé, že v takovém případě by callback funkce zablokovala celý proces a nebylo by možné během sběru dat přijatá data zároveň zpracovávat. Proto byl ve vytvořeném wrapperu proces rozdělen do dvou vláken.

Ačkoliv operace ve vláknech probíhají nezávisle (paralelně), vlákna v procesu sdílejí paměťový prostor, globální proměnné jsou tedy viditelné pro všechna vlákna. Při práci s globálními proměnnými je potřeba zamezit souběhu (race condition). Během souběhu dochází k tomu, že se sdílenými daty najednou operuje více vláken. V případě, že jedno vlákno do sdílené paměti zapisuje a současně z ní druhé vlákno čte, může dojít k chybné interpretaci hodnoty. Kritickou sekci kódu, ve které se pracuje se sdílenou pamětí, je proto nutné chránit proti současnému vícenásobnému přístupu použitím synchronizačních prvků, jako jsou mutexy a semafory.

5.3.3 Implementace asynchronního čtení

Ve wrapperu bylo vytvořeno několik funkcí, které umožňují ovládání asynchronního čtení s ohledem na propojení s LabVIEW.

Asynchronní čtení je zahájeno voláním funkce *rtlsdr_start_async_read*. Funkce nejprve alokuje paměťový prostor pro buffer o velikosti 51,2 Mb (odpovídá 10 vteřinám IQ dat při vzorkovací rychlosti 2,56 MS/s) a inicializuje semafory a mutexy. Poté je potřeba v samostatném vlákně spustit samotné asynchronní čtení (funkce *rtlsdr_read_async_wrap*). Protože použitá knihovna Pthreads umožňuje předat do vlákna jediný parametr, je před spuštěním vlákna provedeno sloučení několika proměnných do struktury *A_param*. Bezprostředně po spuštění vlákna je funkce *rtlsdr_start_async_read* běžící v hlavním vlákně procesu ukončena.

Jak název napovídá, funkce *rtlsdr_read_async_wrap* je wrapem funkce ovladače Osmocomu *rtlsdr_read_async*. Wrap zkonvertuje vstupní parametry a s nimi zavolá funkci *rtlsdr_read_async*. Jedním ze vstupních parametrů je ukazatel na callback

funkci.

V callback funkci *rtlsdr_callback* jsou čtená data průběžně ukládána do globálního bufferu. Tato funkce blokuje vlákno, dokud není ukončena nastavením globálního příznaku *exit_cmd* na hodnotu 1. Pracuje se dvěma proměnnými typu semafor. Semafor *sem_empty_bufs* reprezentuje počet volných zdrojů, tedy kolik je volné paměti v bufferu k zapisování. Ve funkci callbacku je tento počet vždy snížen o počet právě zapsaných prvků (bitů). Druhý semafor *sem_full_bufs* uchovává informaci o tom, kolik prvků bufferu je možné přečíst, tato hodnota je po zápisu dat do bufferu zvýšena. Buffer je implementován jako kruhový, tedy po jeho zaplnění je kurzor pro zápis opět nastaven na začátek bufferu a již přečtená data jsou přepisována. Jelikož s bufferem pracuje také funkce *rtlsdr_get_async_data*, která běží v jiném vlákně, byl navíc použit mutex, který zabraňuje současnému čtení a zápisu v rámci bufferu.

Data z bufferu je možné číst v LabVIEW pomocí funkce *rtlsdr_get_async_data*. Princip práce s pamětí je stejný jako u callback funkce, před čtením z bufferu je snížen počet zdrojů v *sem_full_bufs* a následně je uvolněn stejný počet zdrojů *sem_empty_bufs*. Čtení z bufferu je chráněno stejným mutexem jako zapisování.

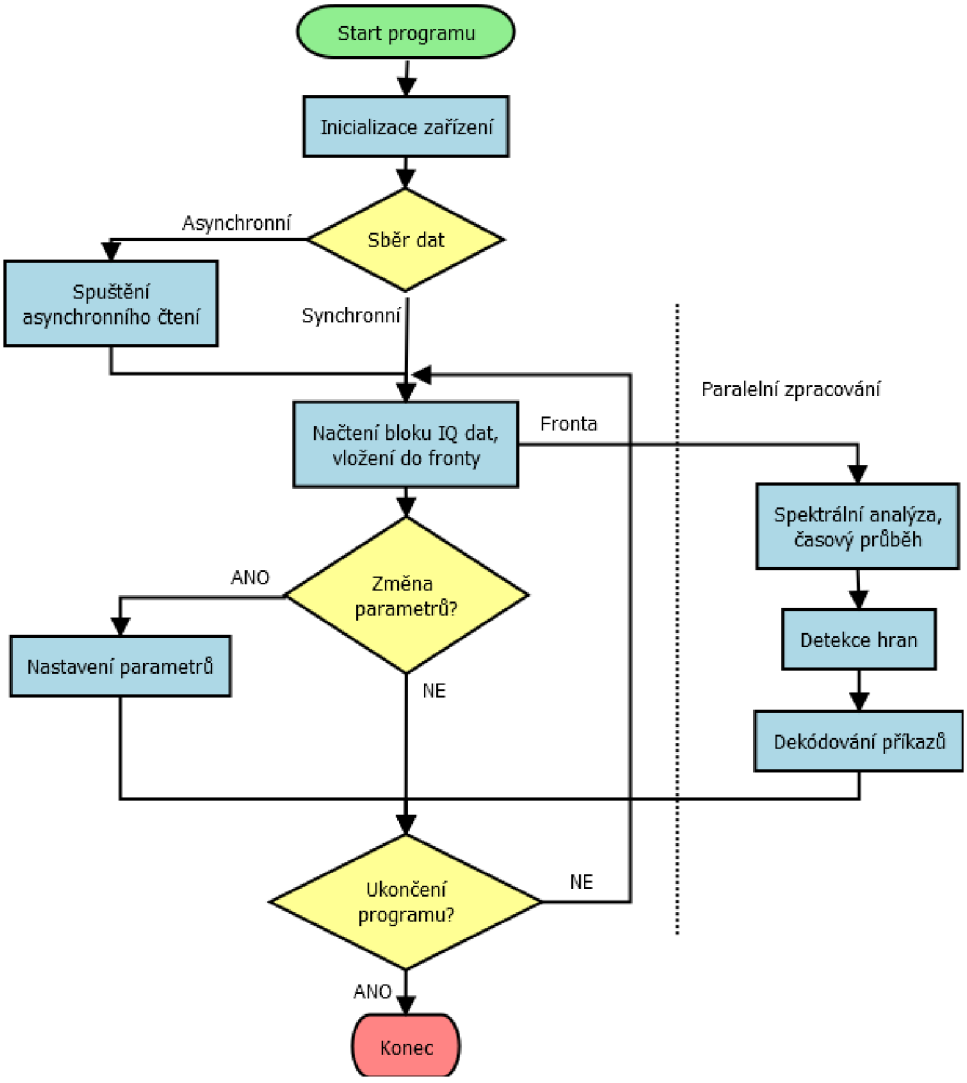
Vlákno načítání dat do bufferu je možné ukončit z LabVIEW zavoláním funkce *rtlsdr_stop_async_read*. V posledním kroku LabVIEW aplikace musí být spuštěna funkce *rtlsdr_free_async_buffer*, čímž se dealokuje paměť bufferu.

5.4 Dosažená funkčnost

Byla úspěšně ověřena správná funkčnost inicializačních, ukončovacích a konfiguračních funkcí a synchronního čtení. Při odlaďování asynchronního čtení v konzoli bylo zjištěno, že funguje jen při empiricky vytvořeném časování pomocí funkce *Sleep* a ruční synchronizaci funkcí pro zapisování a čtení, což jistě nelze považovat za správný programátorský přístup. Přesto byly ve zdrojovém kódu funkce asynchronního čtení ponechány pro budoucí úpravy. I s nimi je vytvořený wrapper DLL použitelný při využití čtení synchronního.

6 APLIKACE V LABVIEW

Hlavním úkolem v rámci této bakalářské práce bylo vytvořit v LabVIEW analyzátor UHF RFID komunikace s využitím USB DVB-T přijímače, popsaného v kapitole 2. Pro potřeby analyzátoru bylo s využitím poznatků z kapitol 4 a 5 vytvořeno komunikační rozhraní pro LabVIEW. Nutno dodat, že cílem práce nebylo vytvořit po všech stránkách dokonalý dekodér RFID komunikace (tím se již zabývala jiná práce, viz [1]), ale spíše otestovat možnosti a omezení použitého hardware v této aplikaci. Následující část práce popisuje princip činnosti analyzátoru a dekódování příkazů čtečky.



Obrázek 6.1 Vývojový diagram analyzátoru

6.1 Popis činnosti programu

Zjednodušený vývojový diagram vytvořené aplikace je na obrázku 6.1. Hlavní část programu tvoří dvě nekonečné smyčky, pracující v režimu producent-konzument. Data

jsou mezi smyčkami sdílena pomocí fronty. Třetí nekonečná smyčka slouží k monitorování změny parametrů, jako je např. změna frekvence nebo vzorkovací rychlosti. Nová konfigurace je do zařízení zapsána okamžitě po načtení jednoho bloku dat. V základním nastavení probíhá čtení v synchronním režimu, pro asynchronní čtení musí být stisknuto příslušné tlačítko ještě před spuštěním sběru dat. Pro práci s USB přijímačem byly vytvořeny subVI, aby některé části programu mohly být použity i v jiných aplikacích.

6.1.1 Start programu

Po spuštění programu jsou nejprve nastaveny ovládací prvky a indikátory na výchozí hodnoty a poté je inicializován připojený hardware. Tím je vytvořen handle zařízení a program přejde ke konfiguraci. V konfiguraci se provede nastavení parametrů USB přijímače na hodnoty, které uživatel zadal ve front panelu. Po konfiguraci přejde program do stavu, kdy čeká na akci uživatele. Touto akcí může být změna v nastavených parametrech zařízení (např. změna vzorkovací rychlosti) nebo pokyn ke spuštění sběru dat, čímž je zahájeno čtení.

6.1.2 Čtení dat

Zatímco u synchronního je ve smyčce spouštěna pouze jediná funkce wrapperu, u asynchronního módu je proces čtení komplikovanější. V prvním kroku je asynchronní čtení spuštěno a poté jsou ve smyčce sbírána data z bufferu. Jakmile je asynchronní čtení ukončeno, je paměť bufferu uvolněna. V obou případech jsou přijatá pole IQ dat vkládána do fronty, ze které jsou vysunována ve smyčce pro zpracování.

6.1.3 Zpracování přijatých dat

Při zpracování signálu se nejprve pole IQ dat ve formátu neznaménkový 8bitový integer (viz kap. 5.2.3) převede na pole komplexních čísel. Následně je odstraněna stejnosměrná složka, kterou program spočítá jako střední hodnotu reálné a imaginární složky v celém poli. Na základě takto upravených dat je zobrazen modul a fáze v samostatných grafech v závislosti na čase a spočítáno a zobrazeno výkonové spektrum.

Pro dekódování příkazů bylo využito subVI z palety Signal Operation pro detekci nástupných hran. Tato subVI vrací pole indexů (pozic), na kterých došlo ve vstupním poli k překročení nastavené prahové hodnoty a na kterých se tudíž vyskytují náběžné hrany. V dalším bloku je na základě známé vzorkovací rychlosti a délky Tari pole indexů hran přepočítáno na jiné pole, jehož prvky odpovídají vzdálenosti sousedících nástupných hran v násobcích Tari.

Vyslaný příkaz čtečky musí obsahovat parametr RTCal, který má délku 3xTari, lze jej tedy v poli násobků Tari snadno nalézt a tím také identifikovat začátek příkazu. Pokud následující hodnota za RTCal je z intervalu 3x – 9x Tari (délka TRCal), jde zřejmě o příkaz začínající preambulí (viz kap. 1.3.2). Bezprostředně za parametrem RTCal resp. TRCal už následuje bitová sekvence, identifikující vysílaný příkaz.

V další fázi dekódování je ze vstupních dat extrahováno prvních osm bitů příkazu. Ve standardu Gen2 platí, že dvojnásobek Tari reprezentuje bitovou hodnotu 1

a jednonásobek znamená nulový bit. Porovnáním se známou bitovou posloupností příkazů (tabulka 6.1) se zjišťuje, o jaký příkaz se jedná. Jelikož některé příkazy jsou uvedeny kódem kratším než osm bitů, provádí se porovnávání nejprve prvních dvou, poté čtyř a až nakonec všech osmi bitů.

Tabulka 6.1 Příkazy čtečky [3], upraveno

Příkaz	Kód	Délka (v bitech)	Příkaz	Kód	Délka (v bitech)
QueryRep	00	4	Read	11000010	> 57
ACK	01	18	Write	11000011	> 58
Query	1000	22	Kill	11000100	59
QueryAdjust	1001	9	Lock	11000101	60
Select	1010	> 44	Access	11000110	56
NAK	11000000	8	BlockWrite	11000111	> 57
Req_RN	11000001	40	BlockErase	11001000	> 57

Ve vysílaných datech nemusí být délky pulsů přesně dodrženy, proto program hodnoty vzdálenosti hran zaokrouhluje na celočíselné násobky Tari v souladu s tolerancemi standardu Gen2. Po úspěšném dekódování příkazu je v clusteru inkrementován dosavadní počet zachycených příkazů daného typu.

6.2 Uživatelské prostředí

Čelní panel vytvořené aplikace je ukázán na obr. 6.2. Kromě ovládacích prvků obsahuje také prvek Tab Control, který umožňuje přepínání mezi zobrazovanými průběhy. Aplikace také umožňuje streamování demodulovaného signálu do TDMS souboru.

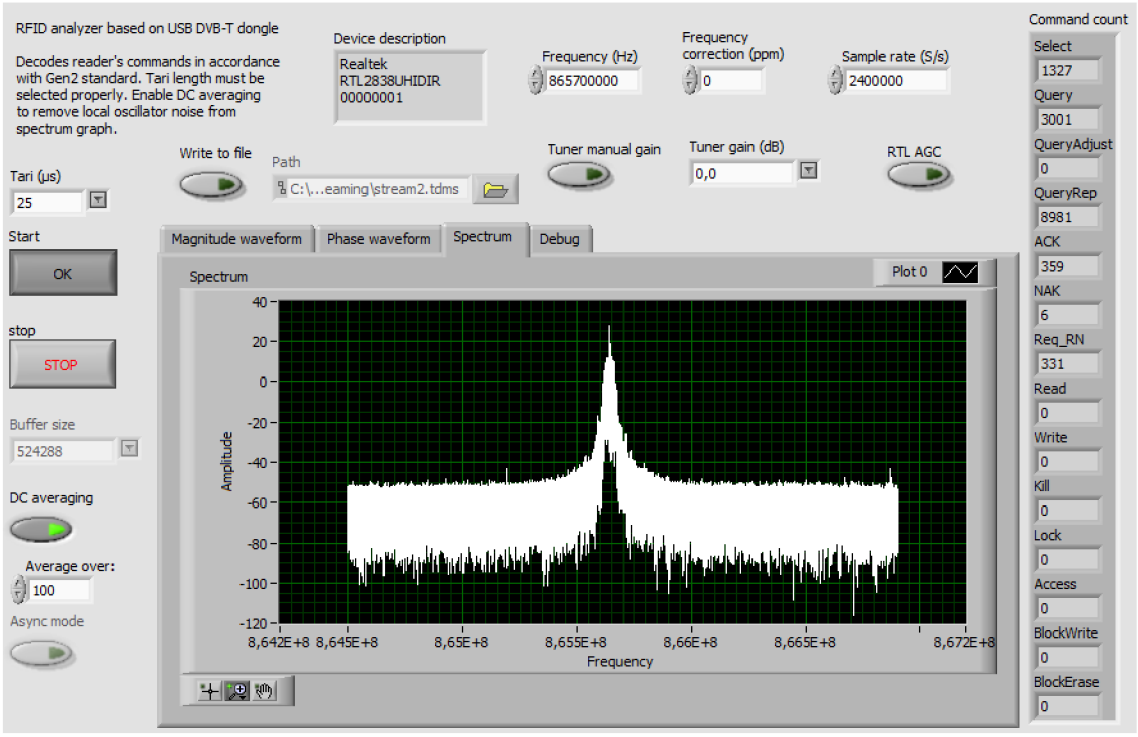
6.2.1 Ovládání programu

V levé části čelního panelu se nacházejí ovládací prvky samotného programu, vpravo nad grafem jsou umístěny prvky pro konfiguraci USB přijímače. Aby byl program příkazy správně dekódovat, je nutné nastavit v levé části panelu používanou hodnotu Tari. Ovládací prvky, které jsou na obr. 6.2 vyvedeny šedě, nemohou být po spuštění aplikace modifikovány. Nad grafem je také zobrazen popis zařízení, který program načte z paměti EEPROM připojeného USB přijímače.

Rozbalovací menu Buffer size určuje délku pole IQ dat, které bude načteno během jednoho cyklu čtení. V levé spodní části čelního panelu je možné ovládat DC průměrování. Při aktivním průměrování je od každého bloku IQ dat odečtena jeho střední hodnota. Pokud je parametr Average over nastaven na vyšší hodnotu než 1, budou do výpočtu střední hodnoty zahrnuty i výsledky předchozích cyklů průměrování.

Program umožňuje naladění přijímače na zvolenou frekvenci a také kompenzovat chybu oscilátoru nastavením korekční hodnoty v ppm. Vzorkovací rychlost je možné nastavit ve vzorcích za sekundu, kde jeden vzorek signálu je reprezentován osmi bity složky I a osmi bity Q. Tlačítkem RTL AGC se aktivuje automatické řízení zisku

v obvodu RTL2832U. Po aktivování ručního nastavení zesílení tuneru (tlačítko Tuner manual gain) lze z rolovací nabídky vybrat hodnotu RF zisku tuneru. Podporované hodnoty zisku se napříč tunery liší, např. R820T umožňuje nastavovat hodnoty z rozmezí 0 – 49,6 dB.



Obrázek 6.2 Čelní panel aplikace

6.2.2 Zobrazované grafy a hodnoty

V příslušných záložkách jsou k dispozici grafy časového průběhu modulu i fáze a také spektrum přijímaného signálu. Poslední záložka Debug je určena pro indikátory použité při odlaďování programu. Sloupec Command count v pravé části čelního panelu slouží pro zobrazení počtu úspěšně dekódovaných příkazů v rámci jednoho spuštění aplikace.

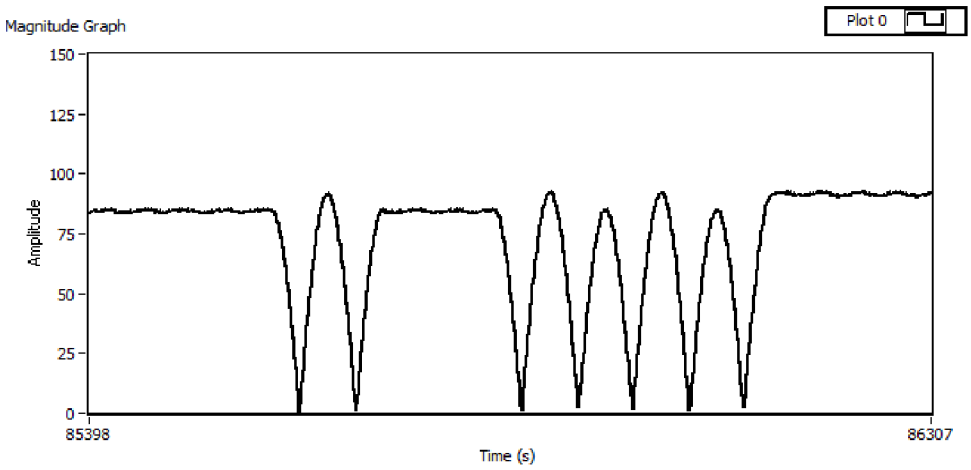
6.3 Ukázky zachycené komunikace

Následující část obsahuje ukázky zachycené RFID komunikace mezi čtečkou Metra RFI21.1 a UHF tagy. Průběhy byly vyexportovány z vytvořené aplikace v LabVIEW v černobílém provedení, které je vhodnější pro tisk. Mřížka grafu je zobrazena jen tam, kde nezhoršovala viditelnost signálu.

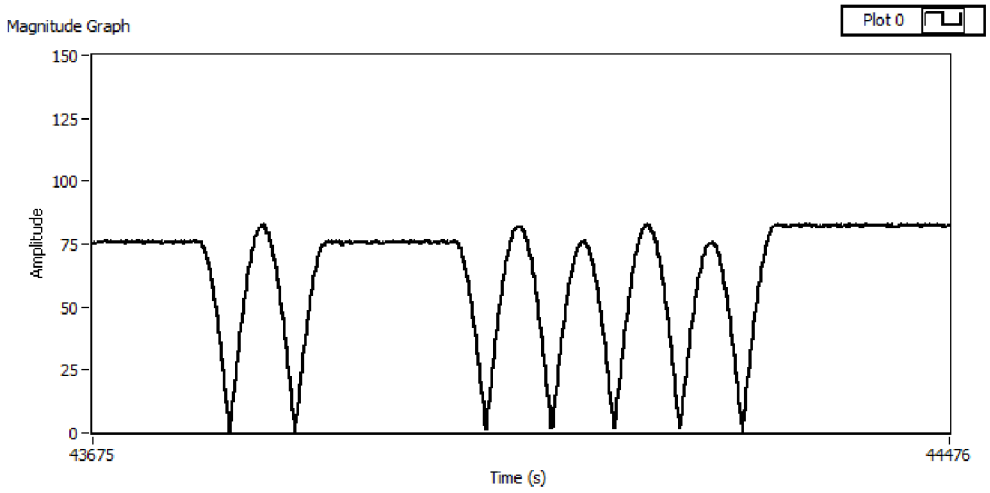
6.3.1 Časové průběhy

Na obrázku 6.3 je ukázán časový průběh demodulovaného signálu, konkrétně byl zachycen příkaz QueryRep. Jsou v něm jasně rozlišitelné všechny součásti – delimiter, referenční Tari, RTCal a datová posloupnost 0000. Obrázek 6.4 ukazuje stejný příkaz

při aktivovaném DC průměrování přes deset iterací. Zvlnění v konstantní části vysílání, které bylo zřetelné na obr. 6.3, je použitím průměrování potlačeno. Při pořizování obrázků dalších průběhů bylo průměrování aktivní. Drobný rozdíl výkonových úrovní není způsoben průměrováním, ale vlastnostmi čtečky Metra, která v průběhu vysílání mění svůj výkon.



Obrázek 6.3 Příkaz QueryRep bez průměrování

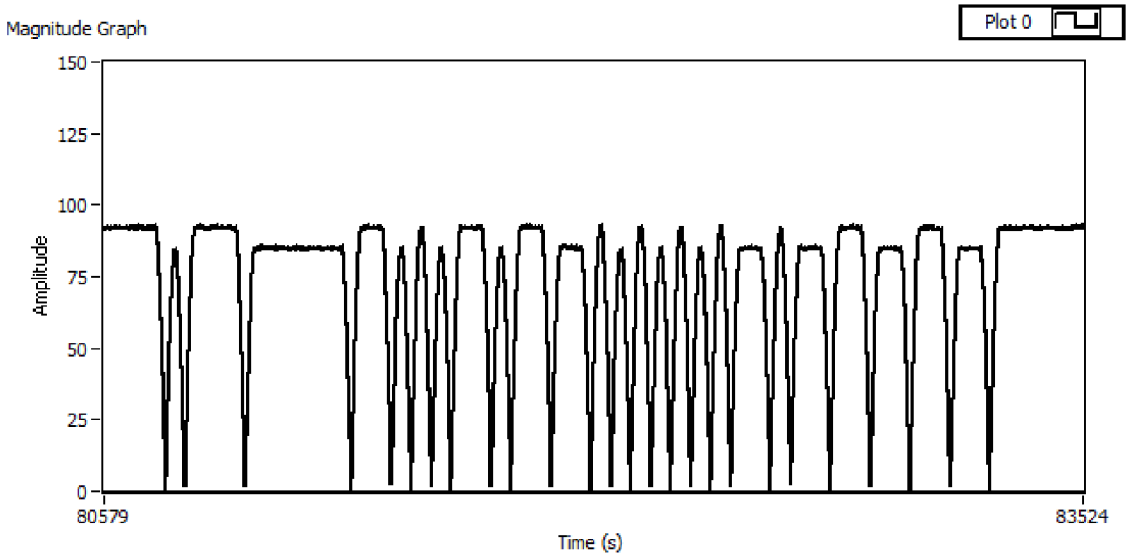


Obrázek 6.4 Příkaz QueryRep s průměrováním

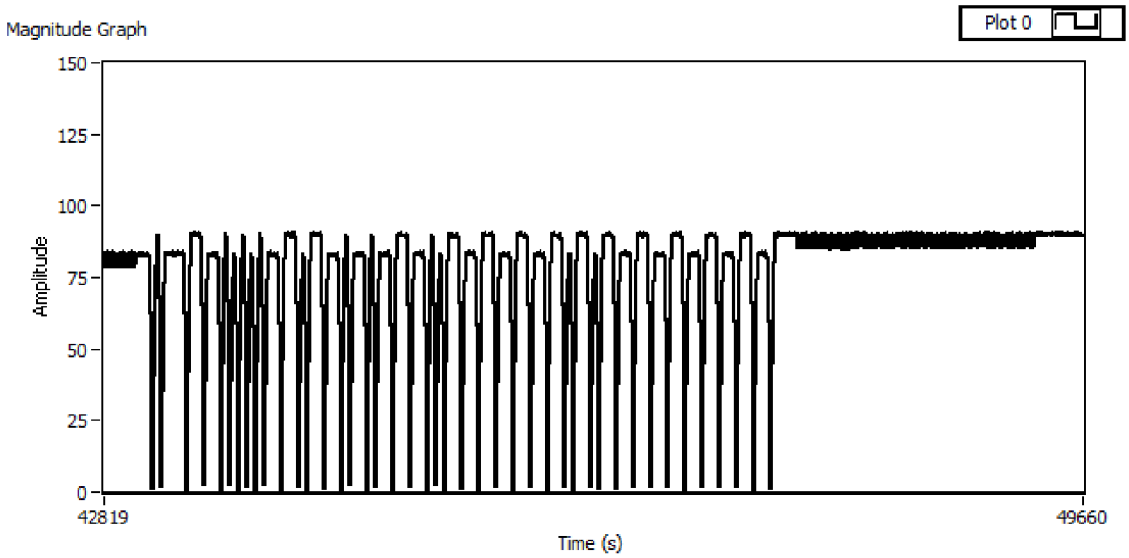
Příkaz Query, obsahující preambuli je zachycen na obr. 6.5. Schodovitý průběh jednotlivých hran je způsoben rozlišením grafu v LabVIEW, kvůli délce příkazu nemohlo být použito větší přiblížení. Je zde také vidět rozdíl ve výkonových úrovních sudých a lichých impulsů. Tento jev je zřejmě způsoben neideálními vlastnostmi rádia.

Navzdory předpokladům se podařilo kromě vysílání čtečky zachytit i odpověď tagu, jak je vidět na obr. 6.6. Tento obrázek zachycuje příkaz Req_RN (kód příkazu 11000001), na který tag odpovídá odvysíláním vygenerovaného náhodného čísla RN16. Detail odpovědi tagu je na obr. 6.7, kvůli lepší viditelnosti není zobrazena celá odpověď RN16, ale pouze její část. V odpovědi tagu bylo použito kódování Miller2 a linkovací frekvence 160 kHz, vzdálenost mezi anténou vysílače a přijímače byla přibližně jeden

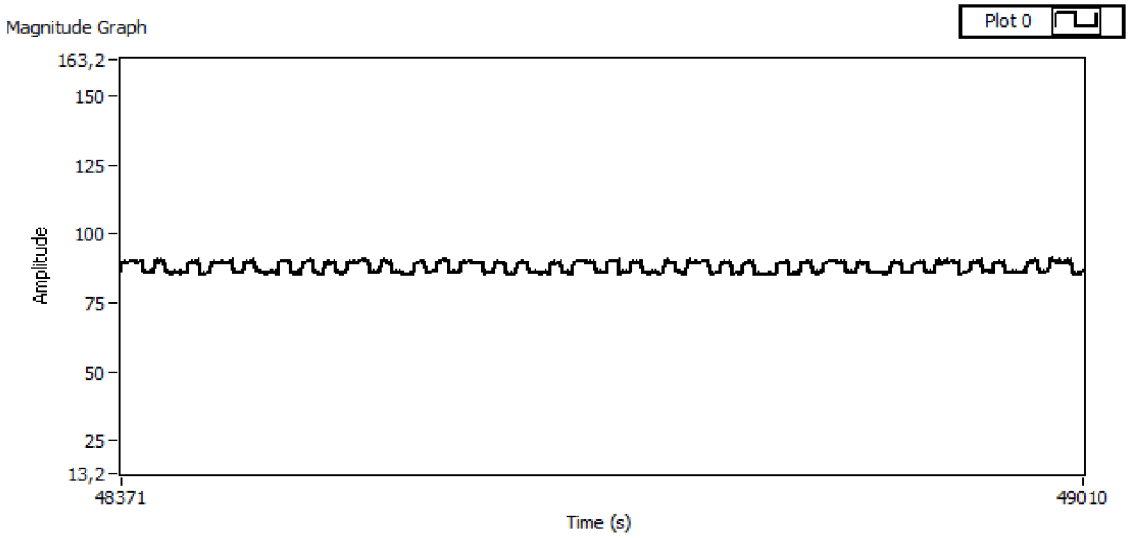
metr. Z obrázku 6.7 je zřejmé, že i při použití nepříliš výkonného hardware, jako je USB DVB-T přijímač, je možné dekódovat odpověď tagu. Implementace dekódování vysílání tagů už je nad rámec této práce, proto nebyla realizována.



Obrázek 6.5 Příkaz Query



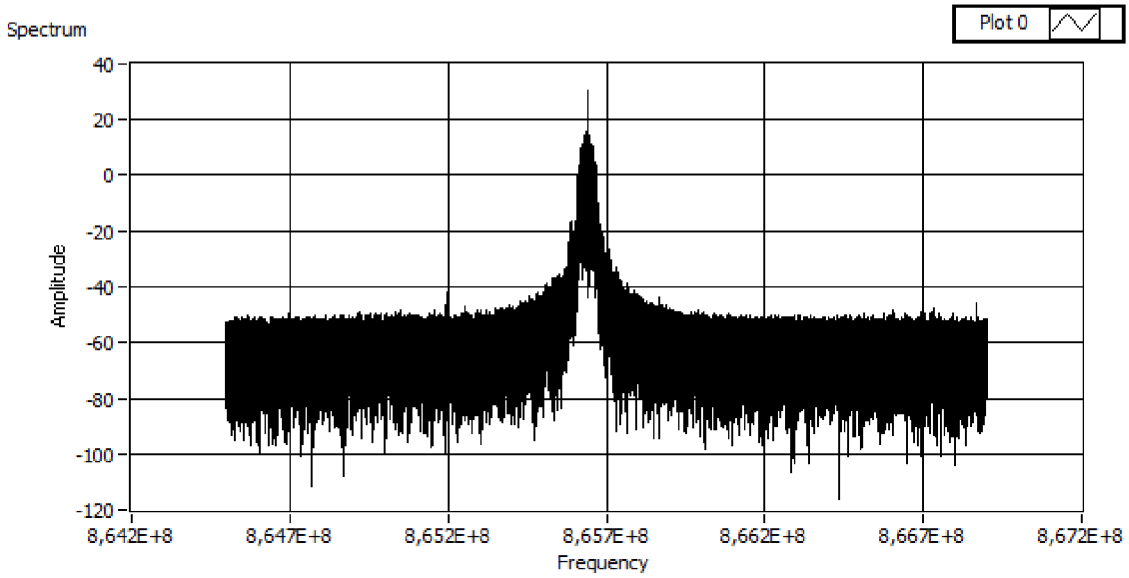
Obrázek 6.6 Příkaz Req_RN s odpovědí tagu



Obrázek 6.7 Část odpovědi tagu - RN16

6.3.2 Spektrální analýza

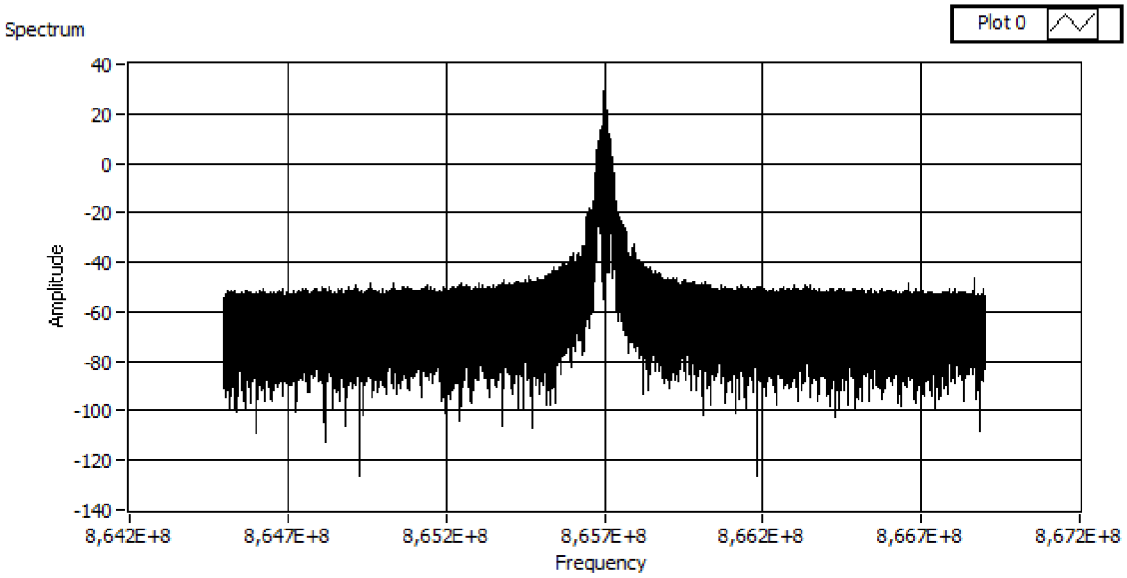
Panel spektrální analýzy umožňuje zobrazení výkonového spektra přijímaného signálu pro váhování Hanningovým oknem. Zobrazována je pouze část spektra ve vzdálenosti odpovídající polovině vzorkovací rychlosti na obě strany od střední frekvence. Na obr. 6.8 je zachyceno spektrum RFID signálu při vysílání čtečky. Čtečka vysílala na kmitočtu 865,7 MHz, vzorkovací rychlost byla nastavena na 2,4 MS/s. Na obr. 6.8 je patrný velký rozdíl v naladění frekvence přijímače a vysílače, spektrální špička se nachází na kmitočtu o cca 100 kHz nižším, než je kmitočet nosné vysílače.



Obrázek 6.8 Výkonové spektrum čtečky

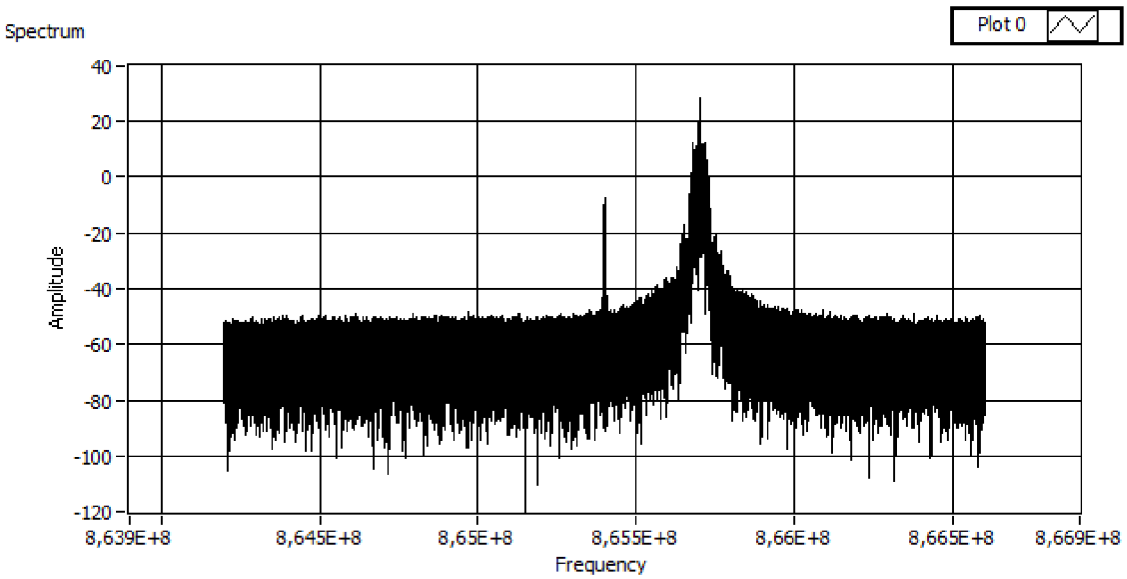
Pro kompenzaci chyby lokálního oscilátoru slouží v aplikaci parametr Frequency correction. Spektrum při nastavení korekční hodnoty na +70 ppm je vidět na

obrázku 6.9. Na frekvenční chybě může mít podíl USB přijímač i čtečka, ale vzhledem k povaze přijímače lze usuzovat, že majoritní podíl na odchylce kmitočtu má lokální oscilátor USB přijímače.



Obrázek 6.9 Spektrum čtečky s korekcí

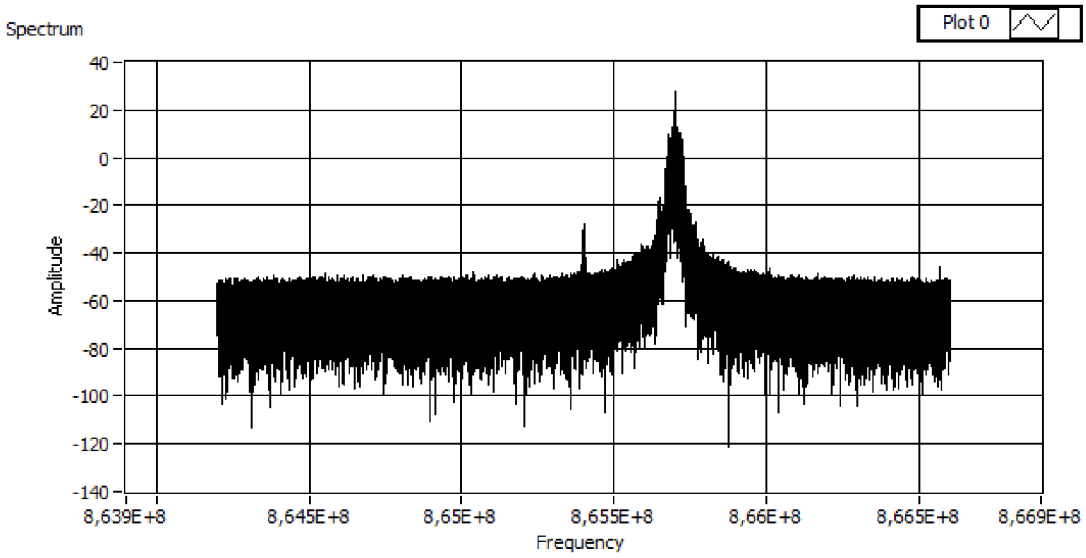
Při odlazení LO mimo vysílání čtečky na frekvenci 865,4 MHz je ve spektru patrná úzká výkonová špička (obr. 6.10), která s RFID signálem (865,7 MHz) zjevně nesouvisí. Protože tato špička ve spektru zůstane i po vypnutí vysílače a odpojení antény, jedná se zřejmě o rušení způsobené oscilátorem v USB přijímači.



Obrázek 6.10 Špička lokálního oscilátoru

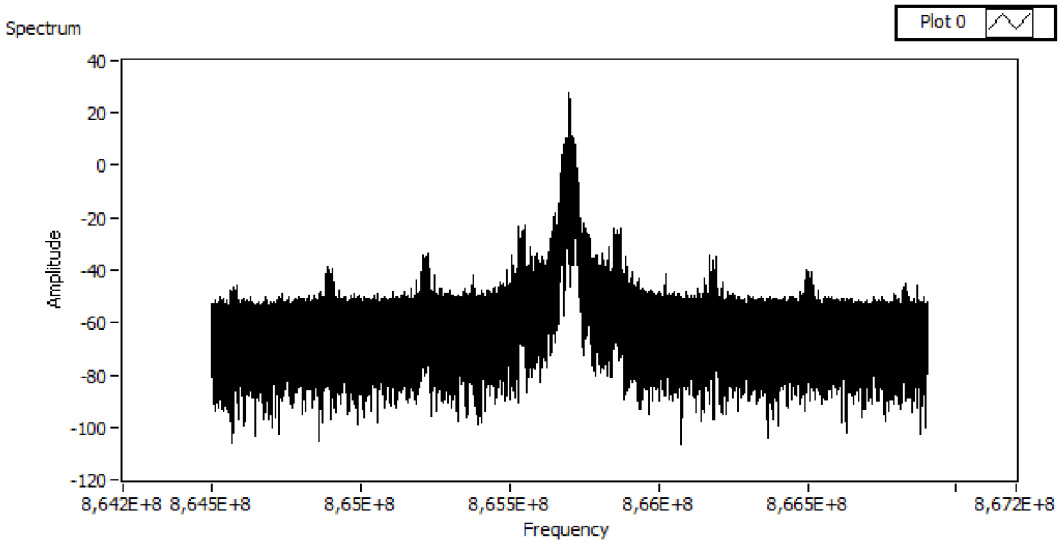
Úroveň signálu lokálního oscilátoru lze snížit použitím průměrování, jak je vidět z obrázku 6.11. Tímto způsobem jej ale úplně odstranit nelze, ani při delším běhu

aplikace a průměrování přes 100 iterací.



Obrázek 6.11 Špička LO při použití průměrování

Spektrum s odpověďmi tagů je zachyceno na obr. 6.12. Protože FFT analýza byla prováděna nad blokem dat čítajícím přes 250 tisíc IQ vzorků, nelze zobrazit spektrum bez signálu čtečky. Na obr. 6.12 je tedy vidět na frekvenci 865,7 MHz vysílání čtečky, menší výkonové špičky jsou způsobeny vysíláním tagu s linkovací frekvencí 160 kHz. Jak bylo popsáno v kapitole 1.3.3, tag moduluje svou odpověď na nosnou čtečky. Odpovědí je obdélníkový signál, jehož bitová perioda je rovna převrácené hodnotě linkovací frekvence. Jelikož se jedná o obdélníkový signál, jsou ve spektru vidět i liché násobky linkovací frekvence 160 kHz, jejichž součtem je obdélníkový signál vytvořen.



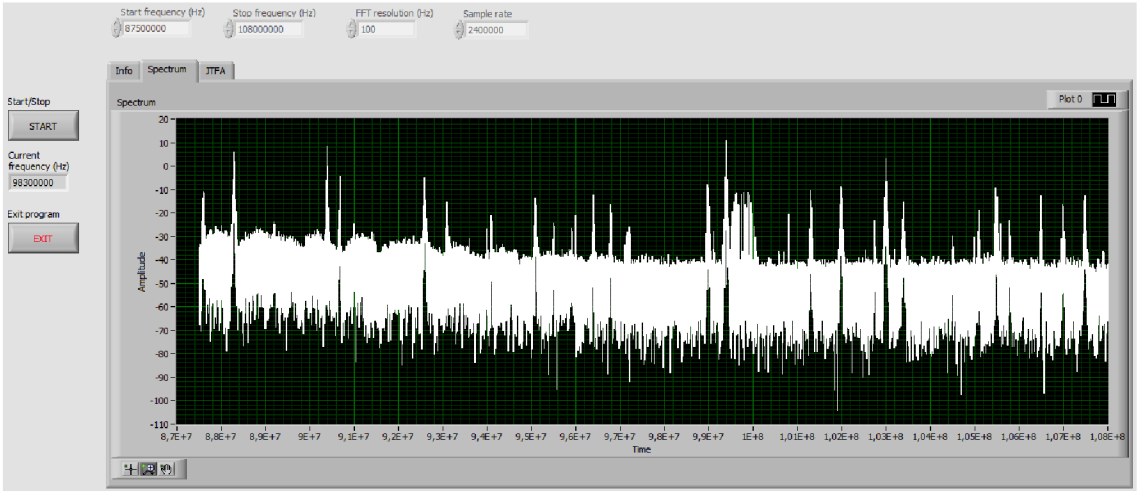
Obrázek 6.12 Odpovědi tagu ve spektru

7 SKENER PÁSMA

Nad rámec požadavků bakalářské práce byla v LabVIEW vytvořena aplikace Band Scanner. Jde o další ukázku praktického využití USB DVB-T přijímače a jeho propojení s LabVIEW. Jedná se v podstatě o zjednodušený spektrální analyzátor, který umožňuje zobrazit spektrum zadané šířky.

7.1 Popis aplikace

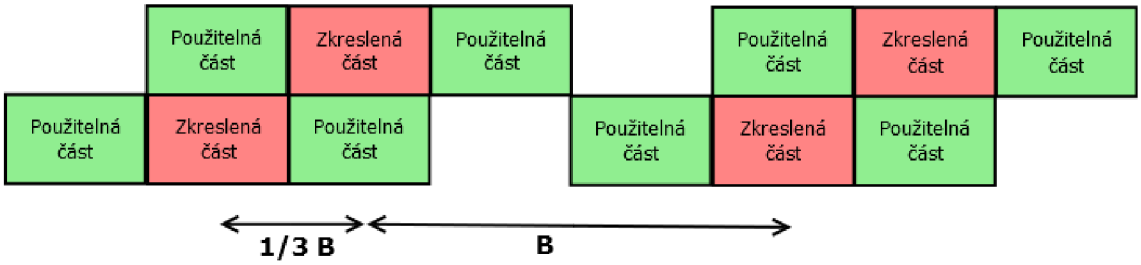
Vytvořená aplikace zobrazuje spektrum v rozmezí podle zadané počáteční a koncové frekvence. Lze tak monitorovat pásmo s šířkou až několik set MHz. Čelní panel aplikace je vidět na obr. 7.1, kde je zachyceno pásmo FM rádií (87,5 – 108 MHz). Kromě frekvenčního rozsahu je zde možné nastavit frekvenční rozlišení FFT a vzorkovací rychlost.



Obrázek 7.1 Čelní panel aplikace skeneru

USB DVB-T přijímač má maximální šířku pásma 2,56 MHz. Pro dosažení větší šířky pásma je nutné přijímač postupně přelaďovat a na každé naladěné frekvenci nějakou dobu sbírat IQ data. Počet přijatých vzorků IQ dat spolu se vzorkovací frekvencí určuje výsledné rozlišení ve spektru.

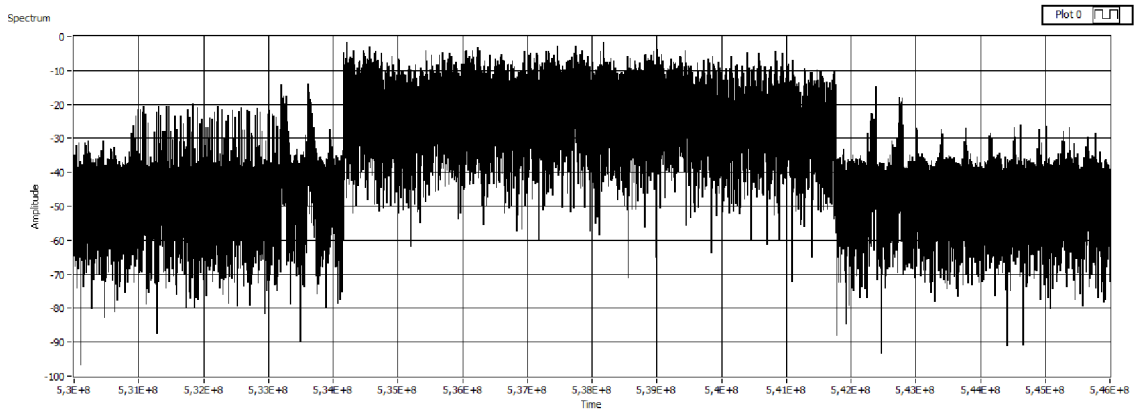
Vzhledem ke zkreslení spektra, které způsobuje šum lokálního oscilátoru USB přijímače (viz kap. 6.3) není možné použít celou dostupnou šířku pásma. Proto nebylo použito jednoduché přelaďování o celou šířku pásma, ale jiný postup. Po naladění na konkrétní frekvenci je získaná část spektra rozdělena na tři bloky o stejné délce. Prostřední blok obsahuje šum LO a není vhodný pro další zpracování. Následně je přijímač přeladěn na frekvenci o třetinu šířky pásma vyšší než předchozí frekvence. Spektrum je opět rozděleno na třetiny a prostřední díl je ignorován. Spojením obou částí jsme získali spektrum bez šumu LO. Postup přelaďování a dělení na bloky je naznačen na obrázku 7.2.



Obrázek 7.2 Postup sestavování výsledného spektra

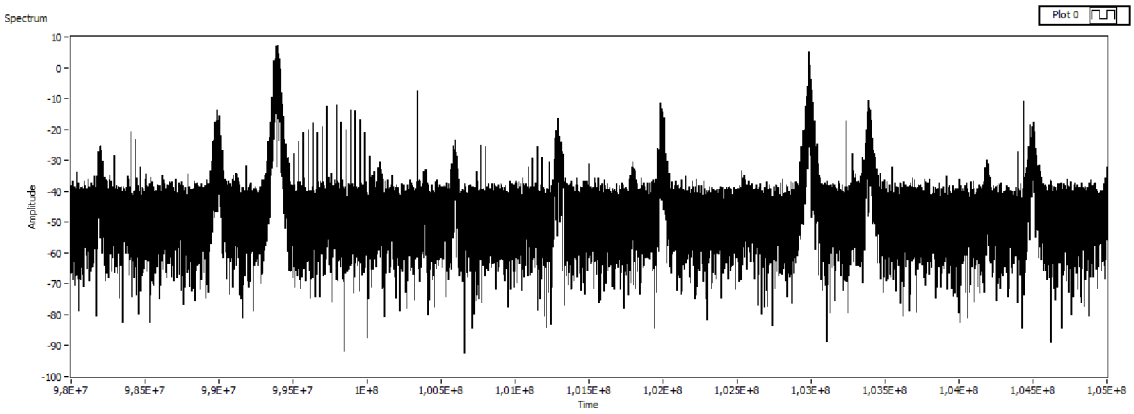
7.2 Ukázky

V této části je uvedeno několik ukázek získaných skenováním pásma. Na obr. 7.3 bylo v pásmu 530 - 546 MHz zachyceno vysílání DVB-T, první veřejnoprávní multiplex, který v Brně vysílá na frekvenci 538 MHz.

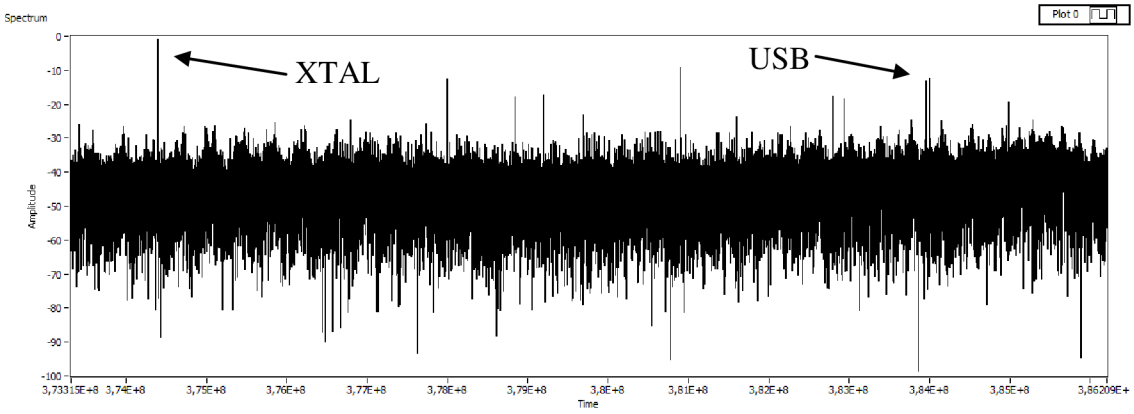


Obrázek 7.3 Vysílání DVB-T

Část spektra FM vysílání je zachycena na obr. 7.4 (98 – 105 MHz). Ve spektru jsou kromě jednotlivých stanic vidět i úzké impulsy, které s FM signálem nesouvisí. Mohou být různého původu, ale z velké části jsou způsobeny rušením z krystalového oscilátoru USB DVB-T přijímače (28,8 MHz) a oscilátorem sběrnice USB (24 nebo 48 MHz). Na obr. 7.5 lze najít impulsy na násobcích základní frekvence oscilátorů. V tomto obrázku XTAL označuje 13. násobek frekvence krystalu USB přijímače (374,4 MHz) a na frekvenci 384 MHz se nachází impuls způsobený oscilátorem USB sběrnice (16x 24 MHz). Kromě celočíselných násobků jednotlivých frekvencí je nutno počítat i s frekvenčními produkty na kmitočtech odpovídajících součtu a rozdílu základních frekvencí a jejich násobků.



Obrázek 7.4 Vysílání FM rozhlasu



Obrázek 7.5 Parazitní impulsy

8 ZÁVĚR

V rámci této práce bylo realizováno softwarové rozhraní pro komunikaci mezi LabVIEW a USB DVB-T přijímačem, které bylo využito pro aplikaci RFID analyzátoru v LabVIEW.

Při použití synchronního čtení funguje komunikační rozhraní bezchybně, s USB přijímačem pracuje v reálném čase a přijatá data jsou relevantní, jak je vidět z ukázek v kapitole 6.3. Asynchronní mód čtení se nepovedlo uvést do provozuschopného stavu vlivem nulových předchozích zkušeností s programováním ve vláknech. Poněvadž wrapper DLL je i tak zkompilovatelný a funkční, byly ve zdrojovém kódu odpovídající části kódu ponechány.

Aplikace RFID analyzátoru v LabVIEW funguje spolehlivě a v reálném čase, hodnoty zobrazované v grafech odpovídají předpokladům. Drobné nedostatky, viditelné především ve spektru jsou způsobeny nedokonalostmi použitého hardwaru. Počet příkazů, které program dekóduje během vysílání čtečky, statisticky odpovídá zastoupení příkazů v RFID komunikaci při daných parametrech. Použitý USB DVB-T přijímač pracuje až nad očekávání dobře. Zejména je potřeba zdůraznit, že dokáže zachytit i odpověď tagu. Její dekódování by bylo záležitostí další softwarové úpravy programu.

Analyzátor byl vytvořen spíše pro demonstrační účely, přesto může nalézt uplatnění např. ve výuce hlavně díky cenově dostupnému hardwaru. Samozřejmě by bylo možné jej po mnoha stránkách zdokonalit, např. dynamickým nastavováním prahu, propracovanějším dekódováním, automatickou detekcí délky Tari atd. Vzhledem k diplomové práci [1] to ovšem postrádá hlubší smysl. Hlavní přínos této práce je především ve vytvořeném komunikačním rozhraní, které otvírá další možnosti využití USB DVB-T přijímače. Příkladem využití může být skener pásma, popsaný v kapitole 7.

Zdrojové kódy knihovny v C, stejně jako DLL soubory nezbytné pro funkčnost programu jsou uloženy na přiloženém kompaktním disku spolu se zdrojovými soubory LabVIEW aplikací. Zdrojový kód wrapperu DLL zde má název rtlsdr_wrap.c.

LITERATURA

[1] Vychodil, J. *Analyzátor UHF RFID komunikace založený na SDR*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2013. 57 s. Diplomová práce. Vedoucí práce: Ing. Aleš Povalač

[2] Metra Blansko a. s. *UHF RFID Compact Reader 865 – 868 MHz, 902 – 928 MHz* [online]. 2010, [cit. 16. 12. 2014]. Dostupné z URL: http://www.asicentrum.cz/data/pdf/datasheet_rfi21.pdf

[3] EPCglobal Inc. *UHF Air Interface Protocol Standard "Gen2v2". Version 2.0.0* [online]. [cit. 16. 12. 2014]. Dostupné z URL: http://www.gs1.org/sites/default/files/docs/uhfc1g2/uhfc1g2_2_0_0_standard_20131101.pdf

[4] Software-defined radio. *In: Wikipedia, the free encyclopedia* [online]. 2014 [cit. 2014-12-17]. Dostupné z: http://en.wikipedia.org/wiki/Software-defined_radio

[5] REED, J.H. *Software radio: a modern approach to radio engineering*. Upper Saddle River: Prentice Hall PTR, c2002, 567 s. ISBN 01-308-1158-0.

[6] RTL-SDR compatibility list v.2: RTLSDR. *In: RTLSDR - inexpensive USB software defined receiver (RTL2832 SDR) community* [online]. April 16th 2014 [cit. 2014-11-11]. Dostupné z: www.reddit.com/r/RTLSDR/comments/s6ddo/rtlsdr_compatibility_list_v2_work_in_progress/

[7] LAUFER, Carl. RTL-SDR.COM. *The Hobbyist's Guide to the RTL-SDR: Really Cheap Software Defined Radio* [elektronická kniha]. 2014 [cit. 2014-12-16].

[8] Rtl-sdr and GNU Radio w/Realtek RTL2832U, E4000 and R820T. *In: Superkuh.com: doing more with less until I can do everything with nothing* [online]. 2014 [cit. 2014-11-11]. Dostupné z: http://superkuh.com/rtlsdr.html

[9] OSMOCOM. *Rtl-sdr – OsmoSDR* [online]. [cit. 2014-12-16]. Dostupné z: http://sdr.osmocom.org/trac/wiki/rtl-sdr

[10] Rtl-sdr.com. *RTL-SDR (RTL2832U) and software defined radio news and projects. Also featuring Airspy, HackRF, FCD and more.* [online]. [cit. 2014-12-16]. Dostupné z: http://www.rtl-sdr.com/

[11] ECE 4670 Spring 2014 Lab 6 Software Defined Radio and the RTL-SDR USB Dongle. *In: EAS Home - College of Engineering and Applied Science: University of Colorado Colorado Springs* [online]. [cit. 2014-12-16]. Dostupné z: http://www.eas.uccs.edu/wickert/ece4670/lecture_notes/Lab6.pdf

[12] LYONS, Richard G. *Understanding digital signal processing*. 3rd ed. Upper Saddle River: Prentice Hall, 2011, xxiii, 954 s. ISBN 978-0-13-702741-5.

[13] KUISMA, Mikael Q. I/Q Data for Dummies. [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://whiteboard.ping.se/SDR/IQ

[14] USB Instrument Control Tutorial - National Instruments. *In: National Instruments* [online]. 2014 [cit. 2014-12-16]. Dostupné z: http://www.ni.com/white-paper/4478/en/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

λ	Vlnová délka
f	Frekvence
B	Šířka pásma
φ	Fáze
I	Reálná část komplexního čísla
Q	Imaginární část komplexního čísla
P	Výkon
DR	Dynamický rozsah

RFID	Radio Frequency Identification, identifikace na rádiové frekvenci
UHF	Ultra High Frequency, ultra krátké vlny
USB	Universal Serial Bus, univerzální sériová sběrnice
DVB-T	Digital Video Broadcasting - Terrestrial, pozemní digitální televize
EPC	Electronic Product Code, elektronický kód produktu
ACK	Acknowledged, potvrzeno
ASK	Amplitude Shift Keying, klíčování amplitudovým posuvem
PSK	Phase Shift Keying, klíčování fázovým posuvem
DSB	Double Side Band, oboustranný
SSB	Single Side Band, jednostranný
PIE	Pulse Interval Encoding, pulsně intervalové kódování
MSB	Most Significant Bit, nejvýznamnější bit
LSB	Least Significant Bit, nejméně významný bit
SDR	Software-defined Radio, softwarově definované rádio
AD	Analog-digital, analogově-digitální
RF	Radio Frequency, rádiová frekvence
FM	Frequency-modulated, frekvenční modulace
DAB	Digital Audio Broadcasting, digitální rozhlas
LNA	Low Noise Amplifier, nízkošumový zesilovač
AGC	Automatic Gain Control, automatická kontrola zisku
DLL	Dynamic-linked Library, dynamicky linkovaná knihovna

TCP	Transmission Control Protocol, primární přenosový protokol
JTFA	Joint Time-Frequency Analysis, časově-frekvenční analýza
DFT	Discrete Fourier Transformation, diskrétní Fourierova transformace
FFT	Fast Fourier Transformation, rychlá Fourierova transformace
DR	Dynamic Range, dynamický rozsah
CLFN	Call Library Function Node
FM	Frekvenční modulace
XTAL	Crystal, krystalový oscilátor