VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Ľubomír Jagoš

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY
DEPARTMENT OF RADIO ELECTRONICS

GENEROVÁNÍ OFDM SIGNÁLŮ POMOCÍ GENERÁTORU LIBOVOLNÝCH PRŮBĚHŮ LW410
GENERATION OF OFDM SIGNALS USING ARBITRARY WAVEFORM GENERATOR LW410

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Ľubomír Jagoš

VEDOUCÍ PRÁCE
SUPERVISOR
prof. Ing. Aleš Prokeš, Ph.D.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Elektronika a sdělovací technika**
Ústav radioelektroniky

Student: Bc. Ľubomír Jagoš ***ID:*** 140232
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Generování OFDM signálů pomocí generátoru libovolných průběhů LW410

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s principem činnosti a s metodami programování generátoru libovolných průběhů (tzv. arbitrary waveform generator) LeCroy LW410 pomocí PC. Vytvořte sadu funkcí v Matlabu pro přenos dat do generátoru a pro nastavování základních parametrů generovaného signálu (amplituda, kmitočet) a ověřte jejich správnou funkci. V prostředí Matlab vytvořte model OFDM systému a pomocí generovaného signálu ověřte jeho funkčnost.

S využitím grafického rozhraní (GUI) případně Simulinku vytvořte vhodné uživatelské prostředí pro snadné ovládání generátoru. Je požadována možnost nastavení základních parametrů OFDM signálu jako je počet nosných vln, modulační formát nosných vln, struktura modulačních dat, délka ochranného intervalu apod. Rovněž je požadována možnost simulace mnohacestného šíření a rušení aditivním šumem v přenosovém kanále.

DOPORUČENÁ LITERATURA:

[1] LeCroy WaveStation LW4001LW400A Series AWG Remote Programmer's Manual [online]. Chestnut Ridge: LeCroy Corporation 1996 - [cit. 20. května 2010]. Dostupné na www: http://www.lecroy.com/tm/library/manuals/download.asp?id=800.

[2] FAZEL, S., KAISER, S. Multi-Carrier and Spread Spectrum Systems. Chichester: John Wiley & Sons 2003.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 16.5.2017

Vedoucí práce: prof. Ing. Aleš Prokeš, Ph.D.
Konzultant:

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda oborové rady

UPOZORNĚNÍ:
Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Práca popisuje základné prvky OFDM modulátoru, venuje sa ich implementácií v prostredí Matlab, taktiež sa sústreďuje na použitie USRP softvérového rádia spolu s Matlabom a vytvorenie funkčného OFDM modulátoru. Pozornosť je venovaná predovšetkým praktickej implementácií na reálnom hardware.

KĽÚČOVÉ SLOVÁ

OFDM signál, generátor LeCroy LW410, Matlab, GUI, ifft, fft, GnuRadio, kmitočtový alokátor, ISI, ICI,synchronizácia

ABSTRACT

This thesis describing basic concepts and parts of OFDM modulator, showing their implementation in Matlaba. It's also focusing on how to use USRP with Matlab and creating functional OFDM modulator. OFDM signal, waveform generator LeCroy LW410, Matlab, GUI, ifft, fft, GnuRadio, carrier allocator, ISI, ICI

KEYWORDS

OFDM signal, waveform generator LeCroy LW410, Matlab, GUI, ifft, fft, GnuRadio, carrier allocator, ISI, ICI, synchronizácia

JAGOŠ, Ľubomír *Generování OFDM signálů pomocí generátoru libovolných průběhů LW410*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2017. 57 s. Vedúci práce bol prof. Ing. Aleš Prokeš, Ph.D.

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Generování OFDM signálů pomocí generátoru libovolných průběhů LW410“ vypracoval(a) samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi prof. Ing. Alešovi Prokešovi, Ph.D. za odborné vedenie, konzultáce a tiež pánovi Ing. Radimovi Čížovi, Ph.D. a doc. Ing. Romanovi Maršálkovi, Ph.D. a nakoniec Ing. Martinovi Pospíšilovi za užitočné podnety a rady ktoré pomohli pri riešení problémov vtedy keď som to najviac potreboval a nevedel sa pohnúť z miesta.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Aj keď princíp OFDM je známy už dlho, až v súčasnosti sa začína masovo využívať v dôsledku pokroku doby a dostupnosti rýchlych technológií umožňujúcim jeho nasadenie do praktických oblastí života .

Rôzne matematické popisy a nové spracovania signálov predtým nerealizovateľné v dôledku chýbajúceho technologického vybavenia sú dnes uskutočniteľné s omnoho menším úsilím. OFDM patí k základným typom digitálnych modulácií. Tvorí súčasť mnohých ďalších štandardov ako WiMAX, LTE, DVB, DVB-T, DRM, ... a to pre svoje dobré vlastnosti a jednoduchosť implementácie.

Práca sa snaží vytvoriť základnú implementáciu tejto modulačnej metódy v Matlabe tak aby bolo možné meniť jednotlivé parametre a skúmať ich prejavy na výstupe modulátoru. Tiež demonštruje funkčnosť vytvoreného modulátoru demodulovaním jeho signálu pri použití dostupného OFDM demodulátoru.

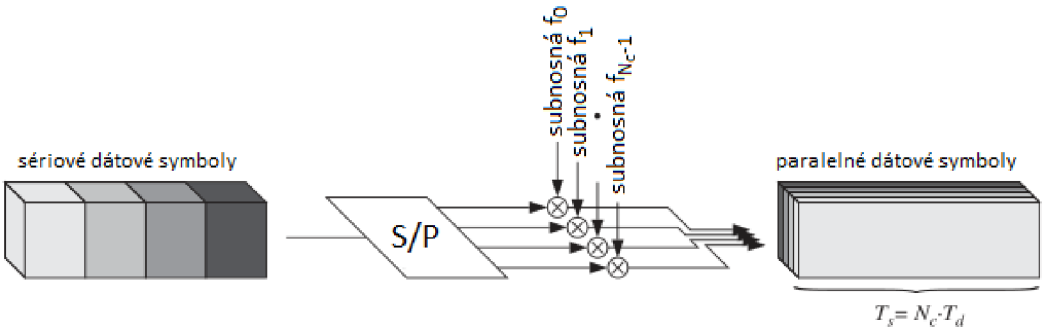
Okrem základných funkcií implementuje jednoduchý paketového systému aby mohla demonštrovať ukážku rádiového spojenia s protistanicou a tým pádom poskytnúť reťazec, ktorý môže poskytnúť priestor pre ďalšie bádanie a zlepšovanie jeho vlastností.

1 TEÓRIA

1.1 Systémy s viacerými nosými

Základným princípom systémov s viacerými nosnými je konverzia rýchleho dátového toku na viacero pomalších paralelných dátových tokov. Každý jeden z týchto paralelných dátových tokov je modulovaný na inú nosnú frekvenciu. Pretože symbolová rýchlosť každej nosnej frekvencie je omnoho nižšia než pôvodná symbolová rýchlosť sériového dátového toku, dôjde k značnému obmedzeniu medzisymbolových preslúchov čím sa znížia nároky na ekvalizáciu kanálu. OFDM je je pomerne jednoduchá technika použitá na efektívne modulovanie viacerých nosných použitím základných algoritmov na spracovanie signálu.

Príklad takejto modulácie so 4 nosnými frekvenciami je znázornený na 1.5 Jednotlivé bloky na obrázku reprezentujú signál v troch dimenziách - čase, frekvenčnom spektre a výkone. Výhodou systémov založených na OFDM modulácií v mobilnom rádiovom kanáli je, že frekvenčná charakteristika kanálu môže byť v okolí jednej nosnej považovaná za konštantnú. Tým pádom dĺžka trvania jedného symbolu by mala byť kratšia než doba zmeny impulznej odozvy kanálu. Ak sa podarí splniť tieto základné podmienky, umožnia nám jednoduchú implementáciu prijímačov.

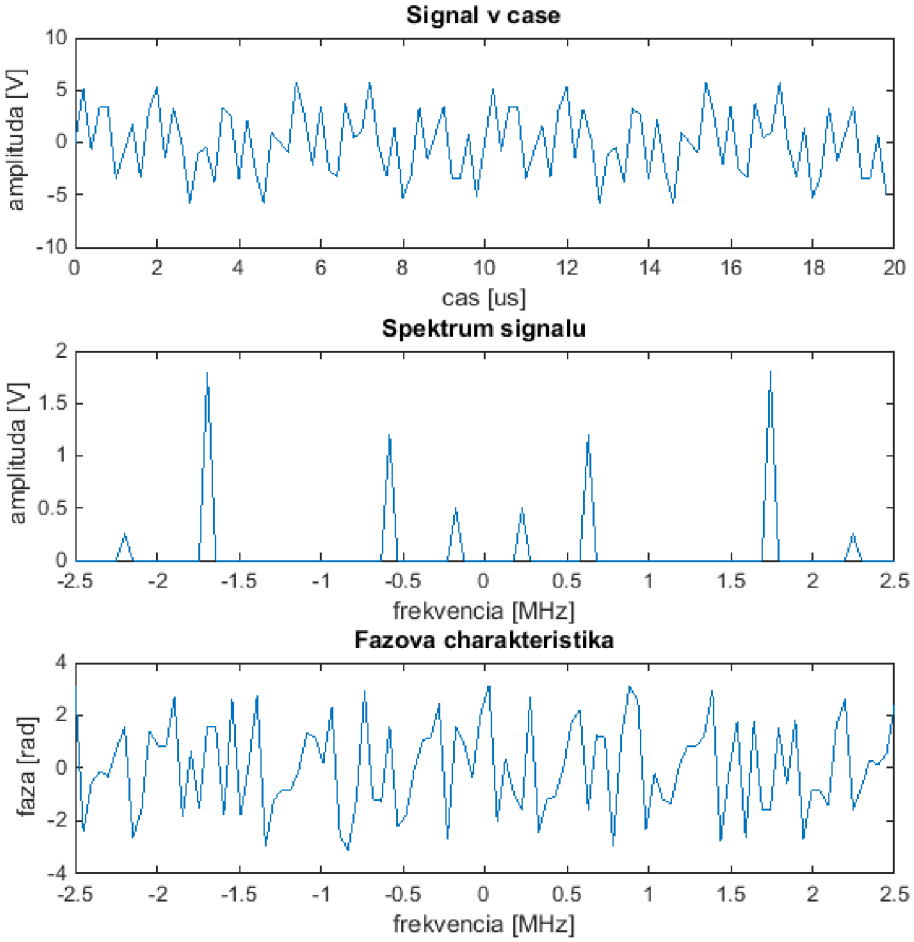


Obr. 1.1: Ukážka modulácie so 4 nosnými frekvenciami **??**

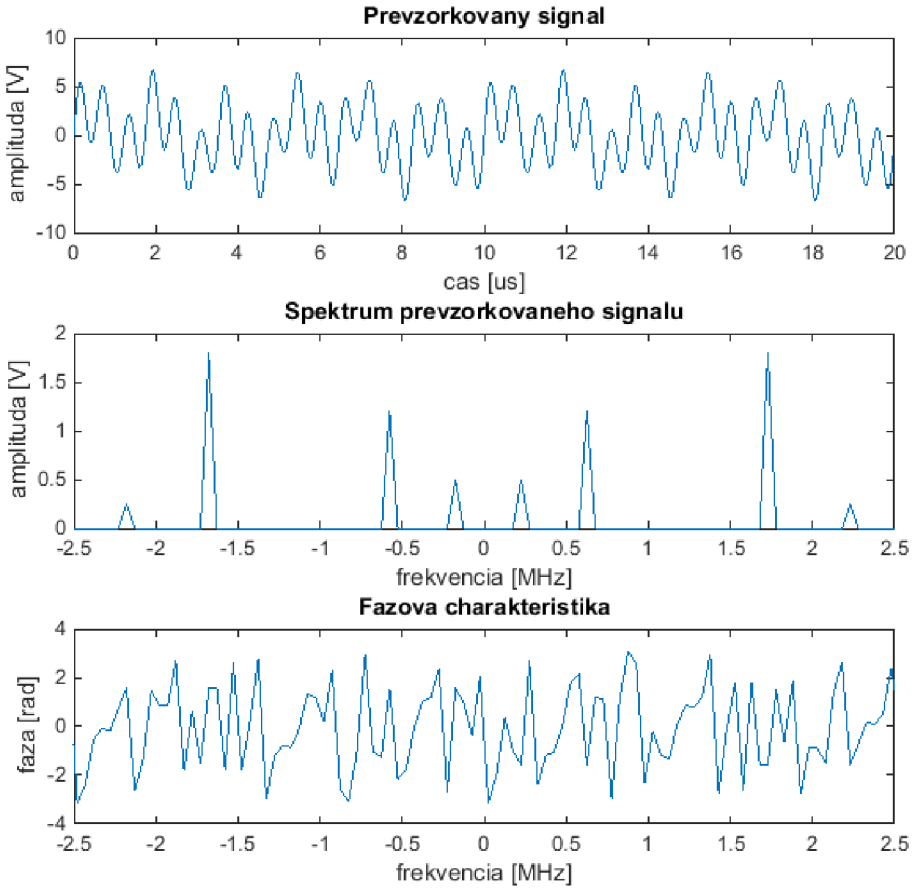
1.2 Interpolácia použitím *ifft()*

Na interpoláciu signálu, tj. prevzorkovanie signálu s násobkom pôvodnej vzorkovacej frekvencie možno s výhodou použiť *ifft()*, pridaním núl do stredu spektra umelo zvýšime vzorkovací kmitočet. Tvárime sa akoby sme za rovnaký čas nazbierali väčší počet bodov čo je vo výsledku použitie väčšej vzorkovacej frekvencie.

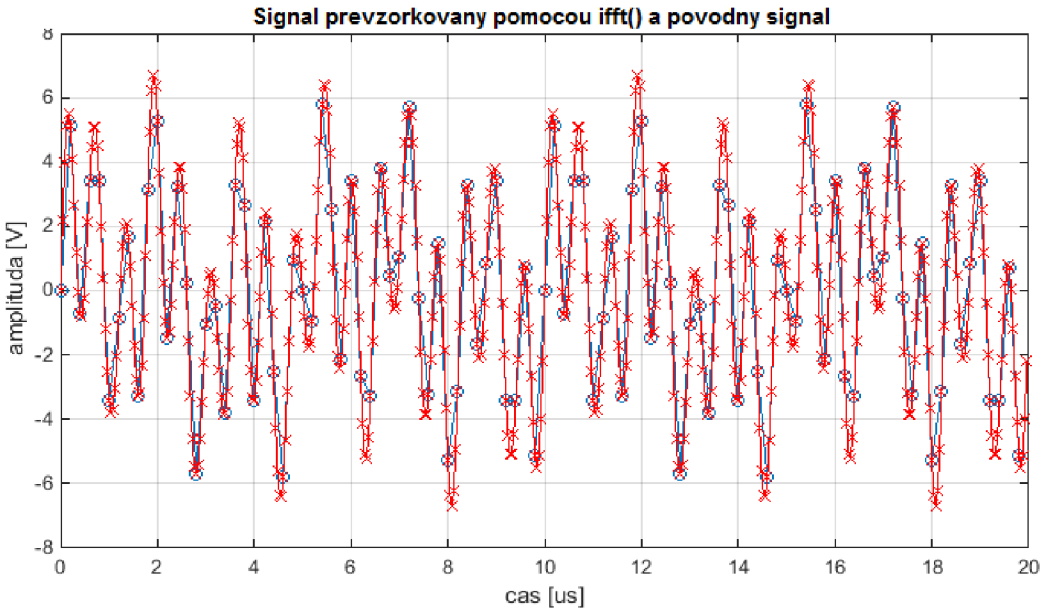
Celý proces možno vidieť v skripte *ifft_upsampling.m* za použitia signálu $y = sin(2pi0.2e6t) + 2.4sin(2pi0.6e6t) + 3.6sin(2pi1.7e6t) + 0.5.sin(2\pi2.2e6t)$



Obr. 1.2: Pôvodný signál v čase a jeho spektrum vzorkované fs = 5MHz



Obr. 1.3: Päťkrát prevzorkovaný signál v čase a jeho spektrum



Obr. 1.4: Pôvodný a päťkrát prevzorkovaný signál

1.3 Základy OFDM modulácie

Komunikačný systém s viacerými nosnými vysiela N_c komplexných symbolov $S_n, n = 0, \ldots N_c - 1$ paralelne na N_c nosných kmitočtoch. Označme dobu trvania jedného sériového symbolu T_d, po konverzií sériového dátového na paralelný sa zmení nasledovne:

$$T_s = N_c T_d \tag{1.1}$$

Princípom OFDM je modulácia N_c dátových tokov na nosné kmitočty s roztečou

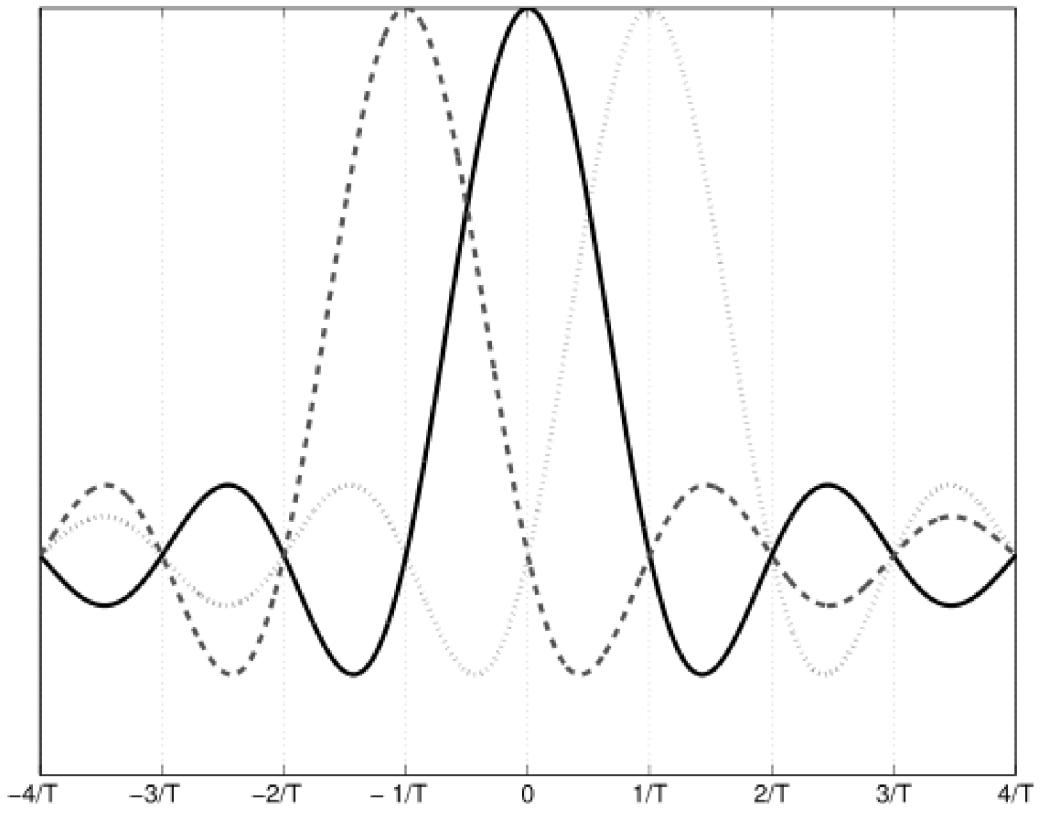
$$F_s = \frac{1}{T_s} \tag{1.2}$$

za účelom zachovania ortogonality medzi jednotlivými vetvami **??**.

Výsledný signál v čase bude podľa [**?**]:

$$s(t) = \sum_{n=-inf}^{inf} \sum_{m=0}^{M-1} A_{m,n} Rect_{Ts}(t - nT_s) e^{j\pi m \frac{t}{Ts}} \tag{1.3}$$

m ... číslo príslušného nosného kmitočtu
n ... poradie symbolu
$A_{m,n}$ ... n-tý symbol vysielaný v poradí na m-tej nosnej
$Rect_T$ ... pravouhlé okno s dobou trvania T jedného OFDM symbolu



Obr. 1.5: Ortogonalita subnosnych v OFDM [**?**]

Ako sa ďalej uvádza v [3] minimálny vzorkovací kmitočet nám určí nosná s najvyššou frekvenciou $f_{vz} = \frac{M}{T}$ a pri vzorkovaní s F_s dostaneme pre nultý symbol

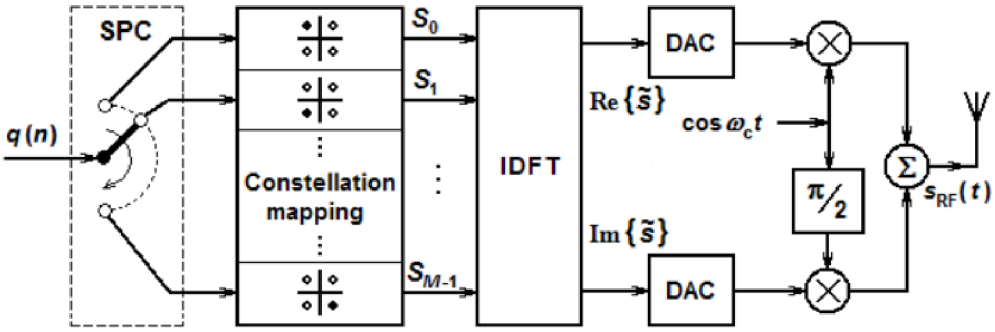
$$s_0(i\frac{T}{M}) = \sum_{m=0}^{M-1} A_{m,0} e^{j2\pi m \frac{i}{M}} \tag{1.4}$$

čo je predpis IDFT. Rovnaký vzťah platí aj pre ostatné symboly. Demodulácia OFDM signálu tým pádom spočíva výpočtom IDFT z prijatého signálu.

1.4 OFDM modulácia

Na obr. 1.6 vidieť kompletný OFDM modulátor, na vytvorenie OFDM signálu je nutné spraviť nasledujúce operácie:

1. Načítanie vstupných dát
2. Prevedenie sériových dát na paralelné, tj. vstupné pole vzorkov sa prevedie na maticu m x n
3. Modulácia jednotlivých postupností zvoleným typom modulácie



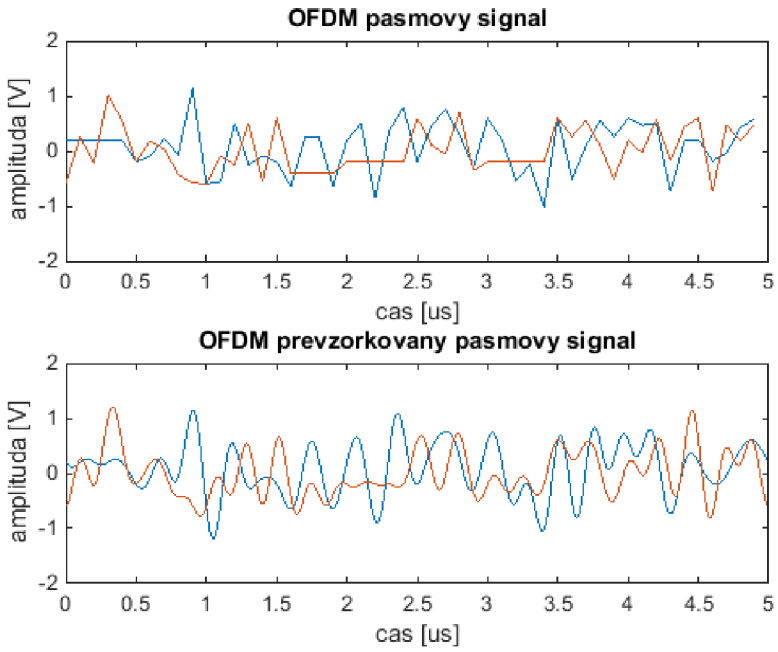
Obr. 1.6: Štruktúra OFDM modulátoru [2]

4. Nad získanou maticou m x n kde jednotlivé riadky predstavujú signál vysielaný na príslušných frekvenciách previesť IDFT a následná serializácia vzorkov m x $n \rightarrow 1$ x $(n*m)$
5. IQ modulácia na získanie pásmového signálu

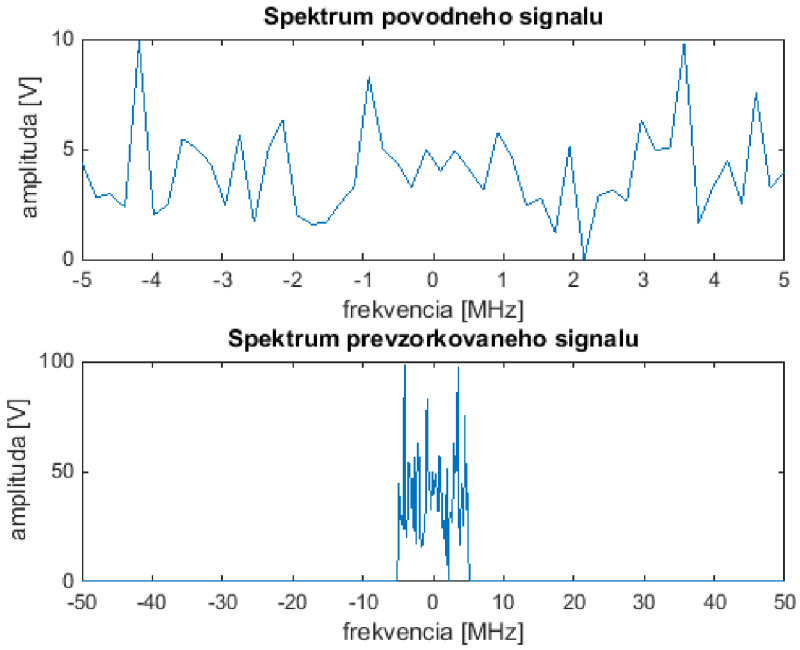
Jednoduchý skript na OFDM moduláciu by teda vyzeral nasledovne:

```
fs = 10e6;      %vzorkovacia frekvencia
dt = 1/fs;

M = 4;
hMod = comm.QPSKModulator('PhaseOffset', 1/4*pi, 'BitInput', true);

%   Vytvorenie vstupnej datovej postupnosti.
%       Vstupne data su vygenerovane ako stlpcovy vektor.
nSymbols = 5;
nCarriers = 10;
dataIn = randi([0 1], log2(M)*nSymbols*nCarriers, 1);

%   Prevod seriovych dat na paralelne.
%       Pocet nosnych = 10  ==> kazda vetva obsahuje nSymbols/nCarriers
%       symbolov v nasom pripade 100/10 = 10 symbolov na nosnu
dataIn = reshape(dataIn, log2(M)*nSymbols, nCarriers);

%   Modulacia dat pre jednotlive nosne kmitocty. Jednotlive stlpce
%   predstavuju signal pre nosne kmitocty.
for k = 1:size(dataIn,2)
    modData(:,k) = hMod.step(dataIn(:,k)) * sqrt(2);
end

%   IFFT
modData = ifft(ifftshift(modData));

%   Serializacia dat.
dataOut = reshape(modData, nSymbols*nCarriers, 1);

%   Prevzorkovanie signalu pomocou IFFT.
upSampFactor = 10;
fs2 = upSampFactor*fs;
dt2 = 1/fs2;

dataOut2 = fft(dataOut);
dataOut2 = [dataOut2(1:end/2); zeros(nSymbols*nCarriers*(upSampFactor
    -1),1); dataOut2(end/2+1:end)];
dataOut2 = ifft(dataOut2) * upSampFactor;    %nasobenie koli zachovaniu
     energie

%   IQ modulacia signalu
fc = 12e6;                                  %frekvencia oscilatoru
tc = [0:upSampFactor*nSymbols*nCarriers-1]' * dt2;
dataOut2 = dataOut2 .* exp(1j*2*pi*fc*tc);  %zmiesavanie
```

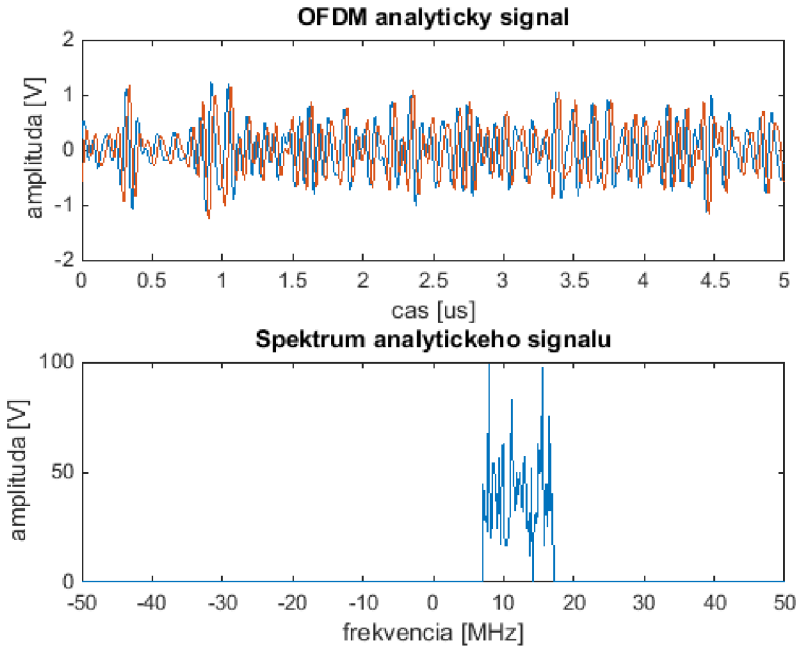
Signál v priebehu OFDM modulácie zachytávajú 1.7 1.8 1.9



Obr. 1.7: OFDM modulácia, vytvorenie pásmového signálu

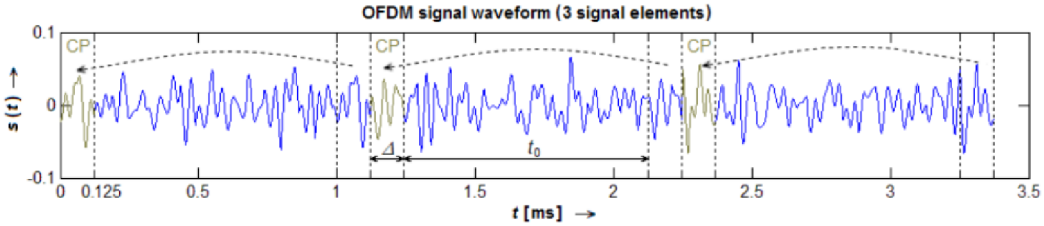


Obr. 1.8: OFDM modulácia, porovnanie spektra pôvodného pásmového signálu a prevzorkovaného signálu



Obr. 1.9: OFDM modulácia, signál po IQ modulácií s fc = 12MHz

1.5 Ochranný interval, cyklický prefix



Obr. 1.10: Vytvorenie cyklického prefixu [2]

Ak uvažujeme premenlivý rádiový kanál v čase ktorý sa mení s τ_{max} a dĺžku trvania symbolu T_s podľa 1.1 tak pokiaľ bude doba trvania jedného symbolu väčšia v porovnaní s časovou zmenou impulznej odozvy kanálu, nebude zmena kanálu ovplyvňovať jednotlivé symboly a vplyv kanálu na ISI a ICI sa značne zredukuje. Za týmto účelom sa v časovej oblasti používa pred každým symbolom tzv. ochranný interval alebo cyklický prefix s dĺžkou trvania T_g, pričom väčšinou platí:

$$T_g \geq \tau_{max} \tag{1.5}$$

Ochranný interval interval znamená že pred každý OFDM symbol je vložená postupnosť núl danej dĺžky, zatiaľ čo pri použití cyklického prefixu vložíme pred symbol kópiu konca daného symbolu ako vidieť na 1.10. Tým pádom dôjde k predĺženiu dĺžky trvania symbolu a novú dĺžka symbolu v časovej oblasti bude:

$$T'_s = T_g + T_s \tag{1.6}$$

Ako sa uvádza [1] aby sa zamedzilo ISI dĺžka ochranného intervalu by mala byť dlhá nasledujúci počet vzorkov:

$$L_g \geq \lceil \frac{\tau_{max} * N_c}{T_s} \rceil \tag{1.7}$$

Tento rozšírený symbol o ochranný interval je predstavuje v časovej oblasti:

$$x_v = \frac{1}{N_c} \sum_{n=0}^{N_c-1} S_n e^{\frac{j2\pi nv}{N_c}}, v = -L_g, \ldots, N_c - 1 \tag{1.8}$$

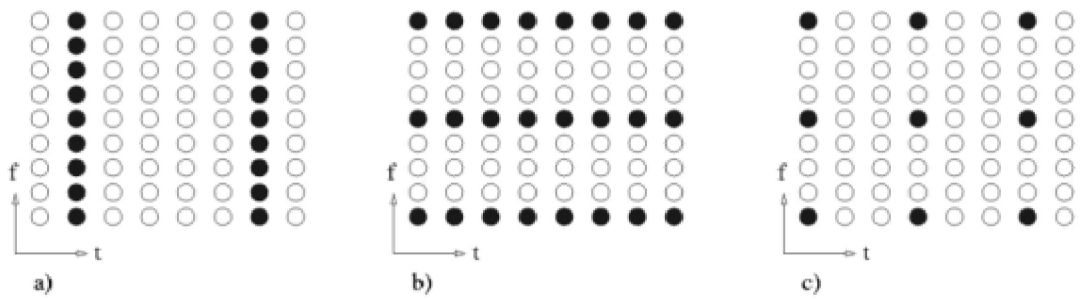
Časová odozva kanálu je tým pádom zasahuje len ochranný interval podľa 1.5 a ten je na strane prijímača odstránený a užitočný signál ostáva bezo zmeny.

1.6 Ekvalizácia OFDM

Prijatý symbol Z_m na danom nosnom kmitočte m na strane prijímača predstavuje súčin vyslaného symbolu A_m a zoslabenia H_m daného kanálom [3]:

$$Z_m = H_m A_m, \forall m = 0 \ldots M - 1 \tag{1.9}$$

Aby mohol prijímač stanoviť koeficienty pre vyrovnávač, musí vedieť odhadnúť prenosovú funkciu kanálu. Za týmto účelom sa niektoré nosné kmitočty obsadzujú tzv. *pilotnými tónmi* ktoré sú známe na strane vysielača aj prijímača. Ak poznáme v prijímači hodnoty pilotných tónov, môžme výpočtom určuť hodnoty H_m. Príklad rozmiestnenia pilotov je znázornený na **??**



Obr. 1.11: Rôzne spôsoby rozmiestnenia pilotov v OFDM signále, **??**

1.7 Výhody a nevýhody OFDM

Zhrňme si teda výhody a nevýhody OFDM systému:

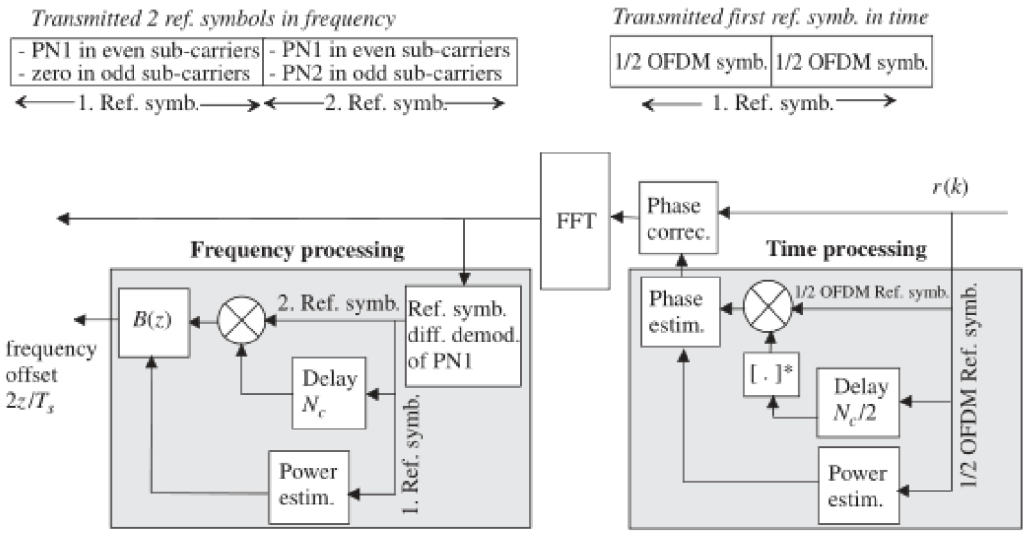
+ Vysoká spektrálna účinnosť s takmer obdĺžnikovým spektrom pri použití dostatočného množstva nosných kmitočtov
+ Jednoduchá realizácia pri použití FFT algoritmu
+ Jednoduchosť prijímača vzhľadom k absencii ISI a ICI pri použití dostatočne dlhého ochranného intervalu
+ Možnosť použitia rozličných modulačných schém pre individuálne nosné kmitočty
- Signál vykazuje vysoký pomer PAPR čo zaťažuje koncový zosilovač vysielača
- Strata spektrálnej účinnosti pri použití ochranného intervalu
- Požiadavka na presnú časovu a frekvenčnú synchronizáciu
- Fázový šum vo vysielači a prijímači negatívne ovplyvňuje výkonnosť systému

1.8 Synchronizácia

Tvorí základný kameň digitálnych komunikácií v systémoch s viacerými nosným kmitočtami. Pri nesprávnej synchronizácií dochádza v systéme OFDM k ICI a ISI a značnému zhoršeniu SNR a tým pádom k zvýšenej chybovosti pri demodulácií dát. Medzi jedny z najznámejších metód odsahujúcu dobré výsledky čo sa týka časového odhadu začiatku symbolov a výpočtu frekvenčného offsetu patrí metóda ktorú uviedli do praxe Schmidl a Cox.

1.8.1 Schmidl a Cox

Princíp synchronizácie podľa je zobrazený na 1.12. Táto synchronizácia umožňuje detekovať fekvenčný posun presahujúci vzdialenosť väčšiu než len medzi dvoma susednými nosnými kmitočtami.



Obr. 1.12: Synchronizácia podľa Schmidl a Cox **??**

Princíp je založený na použití dvoch OFDM symbolov s rozličnou moduláciou. Prvý symbol pozostáva v časovej oblasti z dvoch zhodných symbolov a jemožné ho vygenerovať ako náhodnú PN sekvenciu vo frekvenčnej oblasti umiestnenú na párne subnosné kmitočty a umiestnením núl na nepárne subnosné kmitočty. Výber PN sekvencie nijak výrazne neovplyvňuje výkonnosť synchronizácie. Obsadením len párnych subnosných vo frekvenčnej oblasti predstavuje po prevední IFFT symbol, ktorý je symetrický v časovej oblasti. Druhý symbol je modulovaný rozdielnou moduláciou a obshauje jednu PN sekvenciu na nepárnych subnosných a ďalšiu PN sekvenciu na párnych subnosných kmitočtoch. Tieto dva symboly sú umiestnené pred každý OFDM rámec.

Vytvorenie PN sekvencí je implementované vo funkciách *generateSync1.m* a *generateSync2.m*. Existuje tiež funkcia *generateSync.m* ktorá vygeneruje obe slová naraz. Postup je nasledovný:

- Zadajú sa obsadené nosné a pilotné tóny a vygeneruje sa prázdne slovo dĺžky ako je dĺžka IFFT
- V prvom slove sa na obsadené nepárne nosné umiesntni náhodná postupnosť generovaná napr. randsrc()
- V druhom slove sa na všetky obsadené kmitočty (okrem DC zložky) vygeneruje náhodná postupnosť

1.9 Generátor signálu

Zadanie úlohy hovorí o použití generátoru LeCroy LW410, ktorý bol v počiatkoch riešenia použitý, ale pretože zadávanie vstupnej sekvencie bolo zdĺhavé z dôvodu prepojenia generátoru s počítačom cez GPIB zbernicu rýchlosť nahrávania vzoriek pomalá bol nahradený súčasným USRP softwarovým rádiom, ktoré poskytuje väčšiu voľnosť čo sa týka prepojenia s počítačom a nahrávania dát.

1.10 USRP rádio

Exitujú viaceré verzie USRP N200 rádia:

- r2 netestované, pravdepodobne áno
- r3, funkčné
- r4, neexistuje možnosť zadania typu n200_r4 aby sa nainštaloval obraz

Podarilo sa doteraz sfunkčniť verziu FPGA image 3.9 r2 na USRP N200 r3, firmware r3 sa nahral ale nekomunikoval (alebo skôr nevykazoval známky odosielania signálu a nastavovania parametrov USRP rádia).

Druhé USRP rádio bolo pripojené na Ubuntu LiveCD s gnuradiom a vysielalo sa DPSK signál o šírke pásma 200kHz.

1.10.1 USRP rádio, posielanie vzoriek

Vzorky generované na PC sa odosielajú do rádia kde sú spracované a odoslané na výstup. Počítač môže komunikovať s rádiom viacerými cestami:

- céčkový UHD driver
- univerzálny FPGA image od Ettusu + UHD driver
- Matlab, Simulink ——> práve pomocou UHD driveru

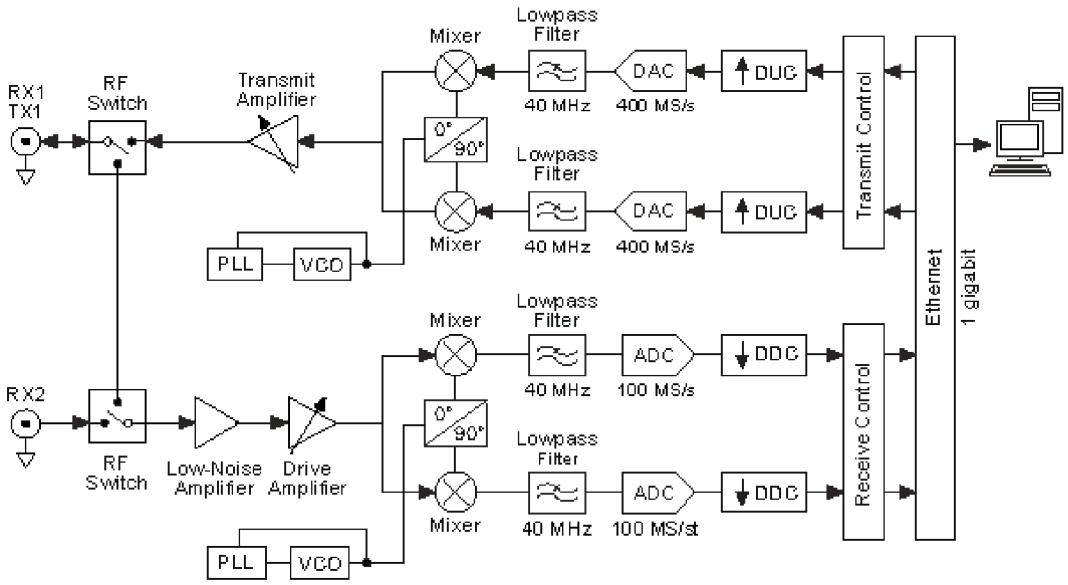
1.10.2 USRP architektúra

1.11 Rodiny zariadení USRP

Od svojho vzniku bolo toto zariadenie postupne vylepšované a vznikli ďalšie verzie poskytujúce rýchlejšie prepojenie s počítačom. Nasledujúca tabuľkla zhrňuje súčasné modely USRP zariadení a ich základné parametre.

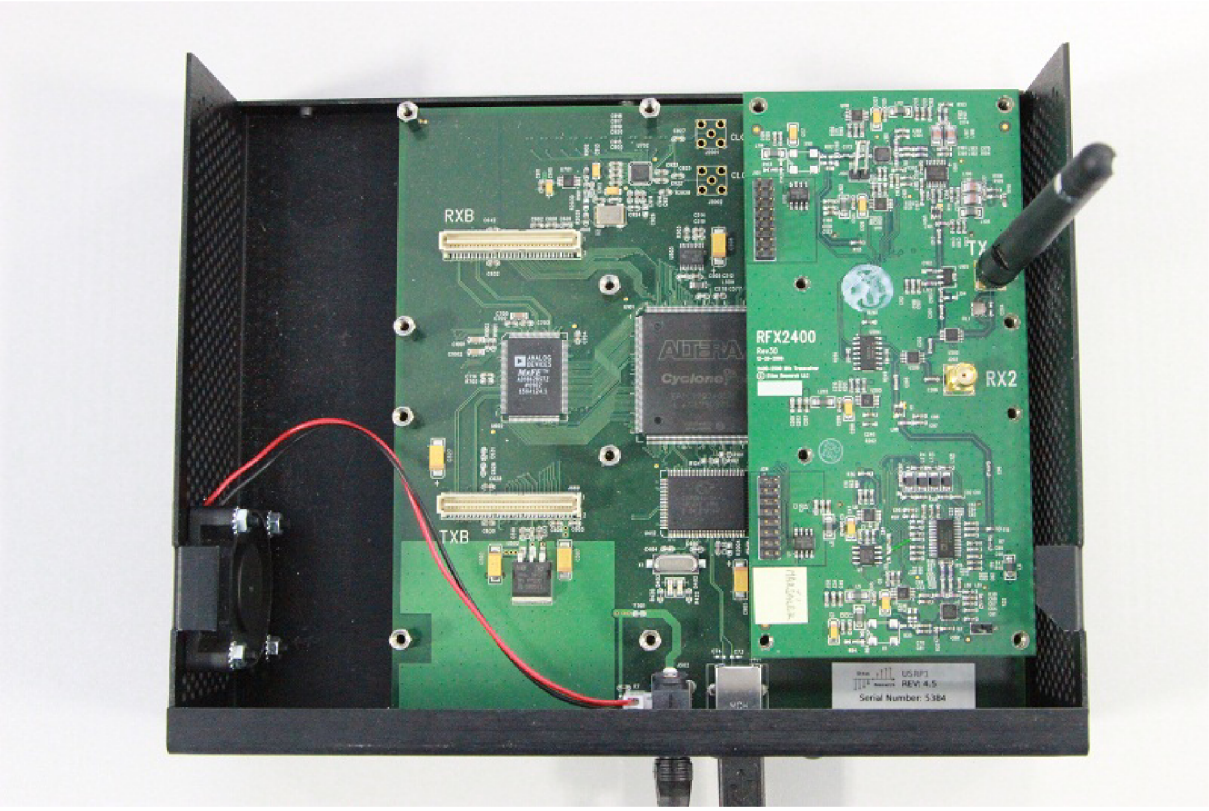
USRP rádia boli pôvodne vyvinuté firmout Ettus Research, ktorú v roku 2010 odkúpila firma National Instruments. Preto novšie USRP zariadenia nesú označenie firmy National Instruments.

H]

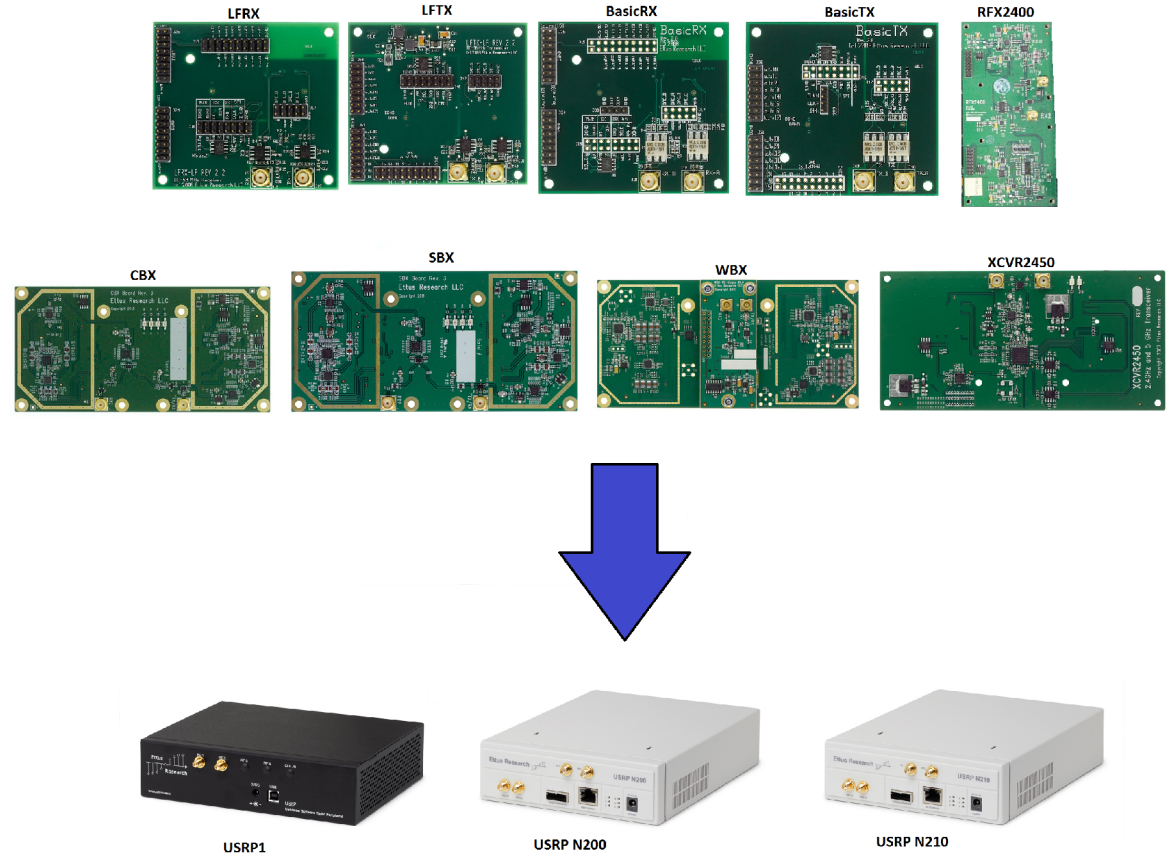


Obr. 1.13: USRP architektúra

Základná doska sa stará o generovanie hodinového kmitočtu a synchronizáciu, obsluhu FPGA, ADC a DAC prevodníkov, reguláciu napájania a obsluhu rozhrania pre pripojenie s počítačom alebo iným kompatibilným zariadením. Za účelom ďalšieho analógového spracovania ako je prevzorkovanie alebo podvzorkovanie signálu, jeho filtrácia a iné sa starajú prídavné moduly, ktoré sa do základnej dosky pripájajú pomocou konektorov ako vidieť na 1.15. FPGA na základnej doske prevádza preklad prijatého analógového signálu z modulu prijímača na komplexný pásmový signál a ten odosiela do pripojeného počítača. V prípade vysielania prijíma vzorky pásmového signálu a stará sa o ich odoslanie na vstup DAC prevodníku.



Obr. 1.14: Pripojenie modulu pre vysielanie/príjem do USRP



Obr. 1.15: Vybraté USRP zariadenia a prídavné moduly

Na obr. 1.15 možno vidieť základné typy USRP zariadení. Prvá generácia týchto rádií používala na prepojenie s počítačom USB rozhranie, v nasledujúcich generáciach bolo USB rozhranie nahradené ethernetovým rozhraním, ktoré poskytuje väčšiu prenosovú rýchlosť a tým pádom umožňuje generovať pásmový signál s väčšou šírkou pásma.

V nasledujúcej tabuľke z [7] je uvedená analógová šírka pásma vybraných modulov, ktoré je možné pripojiť k USRP rádiu.

typ modulu	frekvenčné pokrytie	šírka pásma
WBX-120	50 MHz - 2.2 GHz	120 MHz
SBX-120	400 MHz - 4.4 GHz	120 MHz
CBX-120	1.2 GHz - 6 GHz	120 MHz
UBX-160	10 MHz - 6 GHz	160 MHz
WBX	50 MHz - 2.2 GHz	40 MHz
SBX	400 MHz - 4.4 GHz	40 MHz
CBX	1.2 GHz - 6 GHz	40 MHz
BasicRX/BasicTX	1 – 250 MHz	určené AD-C/DAC vzorko-vaním
LFRX/LFTX	DC-30 MHz	30 MHz
XCVR2450	2,4 - 2,5 GHz	cca 20MHz
RF2000	2,4GHz a 5GHz	—

FPGA sa stará o príjem a vysielanie vzoriek signálu do ADC, DAC prevodníkov. Rýchlosť s akou spracúva vzorkky určuje šírku pásma signálu, v tabuľke z [7] sú uvedené rýchlosti pre jednotlivé modely rádií.

USRP Model	ADC rýchlosť (MS/s)	DAC rýchlosť (MS/s)
USRP B200mini/USRP B205mini	61.44 MS/s (simplex)	61.44 MS/s (simplex)
USRP B200/USRP B210	61.44 MS/s (simplex)	61.44 MS/s (simplex)
USRP E310/USRP E312	61.44 MS/s (simplex)	61.44 MS/s (simplex)
USRP N200/USRP N210	100 MS/s	400 MS/s
USRP X300/USRP X310	200 MS/s	800 MS/s

A nakoniec šírka pásma použitého rozhrania na pripojenie s počítačom. Tabuľka čerpaná z zobrazuje maximálnu možnú rýchlosť pri prenose 16 bitových vzoriek medzi počítačom a USRP zariadením.

Rozhranie	USRP model	PC rozhranie (MS/s@16-bit I/Q)	Half/Full Duplex
USB 2.0	USRP1, B100	8	Half Duplex
USB 3.0	B200, B210, B200mini, B205mini	61.44	Half Duplex
Gigabit Ethernet	N200/N210	25	Full Duplex
10 Gigabit Ethernet	X300/X310	200	Full Duplex
PCI-Express (4-lane PCIe Card)	X300/X310	200	Full Duplex
PCI-Express (1-lane ExpressCard)	X300/X310	50	Full Duplex

Ak chceme zistiť maximálnu možnú šírku pásma pre danú konfiguráciu postu-

pujeme nasledovne:

- Systém s USRP N200 pripojený k počítaču pomocou 1Gbps ethernetu s programom používajúcim 16b vzorky a modul so šírkou pásma 40MHz poskytuje šírku pásma cca 20MHz.
- Limitujúcim faktorom v tomto prípade je ethernetové rozhranie ktoré poskytuje prenos len 25MS/s čo je približne 20MHz šírka pásma.

Za účelom generovania OFDM signálu boli použité momentálne dostupné USRP1 a USRP2 N200 zariadenia spolu s modulmi XCVR2450 a RF2000 ktoré dokážu generovať signál v pásme *2 - 5 GHz*.

1.11.1 Maximálna šírka pásma

Aby sme vedeli určiť základný parameter našej zostavy, je otázne aká je možná maximálne dosažiteľná šírka pásma generovaného signálu. Vzorky komplexného signálu sú do USRP zariadenia odsielané ako *float16* čo predstavuje *2x16b* na jednu vzorku, spolu *32b*. ADC prevodníky USRP rádia vzorkujú s fixnou frekvenciou *100MHz*. Pri využití celej šírky pásma prevodníkov musíme generovať signál s rovnakou vzorkovacou frekvenciou *100MHz* a každá vzorka obsahuje *32b*, výsledná dátová rýchlosť je

$$R_gen = 100MHz.32b = 3200Mbps = 3.2GHz$$

Je zrejmé že najužšie hrdlo predstavuje rozhranie, ktorým je USRP pripojené k počítaču. USRP1 sa pripájajú k počítaču pomocou USB2 ktoré má maximálnu prenosovú rýchlosť 60MHz. USRP N200 je už pripojené 1Gbps ethernetom čo umožňuje generovať signál o šírke pásma:

$$f_BW = \frac{R_{gen}}{32b} = 30MHz$$

Gigabitový ethernet umožňuje generovať signál s maximálnou šírkou pásma *30MHz*. Pri pokusoch keď boli do USRP cyklicky odosielané predpripravené vzorky bieleho šumu sa podarilo generovať signál so šírkou pásma 10MHz.

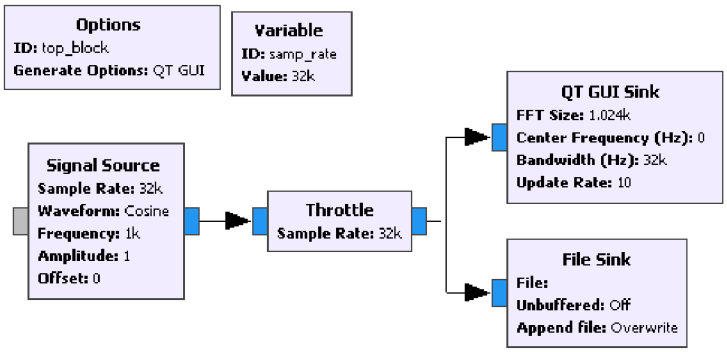
Pri cyklickom odosielaní vzoriek do USRP sme tým pádom limitovaný jeho rozhraním, ak by sme vzorky generovali v reálnom čase začneme byť obmedzovaný rýchlosťou procesoru počítača, ktorý na dnešných počítačoch je taktovaný do približne 3GHz a výpočtovou náročnosťou použitých algoritmov.

V prípade generovania skutočne širokopásmových signálov by sme museli použiť odlišný prístup a metódy, v tejto oblasti sú dnes používané hradlové polia FPGA, ktoré poskytujú maximálnu hardwarovú rýchlosť čo súvisí s ich architektúrou.

Výsledný program si nekladie za cieľ dosiahnuť maximálnej šírky pásma ale jednoduchú modifikovateľnosť a prehľadnosť, 30MHz je tým pádom horná hranica šírky pásma.

1.12 GnuRadio

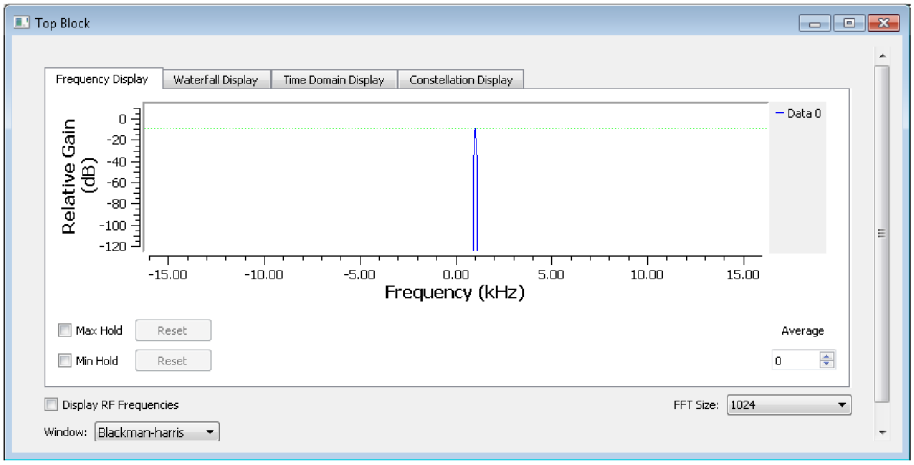
Jedná sa o signálový framework vyvíjaný primárne v opreačnom systéme Linux, používajúci mnohé knižnice vyvíjané primárne pre tento systém. Časovo kritické bloky pracujúce so signálom sú implementované v C++ knižniciach a ako obaľujúci jazyk je použitý Python. Gnuradio zabezpečuje časovanie a načítavanie dát ich buffrovanie pri presune medzi jednotlivými blokmi ktoré spracujú signál. Na implementáciu algoritmov v prostredí gnuradio môžeme písať kód priamo v Pythone alebo C++ a skompilovať ako normálnu aplikáciu alebo môžeme použiť vývojové prostredie *gnuradio-companion*, ktoré poskytujé grafické programovanie, teda každá funkcia (blok) implementovaný v Pythone/C++ je tu zobrazená ako blok so vstupmi a výstupmi. Smer toku dát medzi blokmi zobrazujú šípky medzi nimi ako možno vidieť na nasldujúcom obrázku:



Obr. 1.16: Ukážka zostavenia programu v prostredí Gnuradio Companions

Framework na pozadí sa stará o generovanie dát a ich ukladanie do súboru a zobrazovanie v grafe, na to slúžia bloky *QT GUI Sink* a *File Sink*.

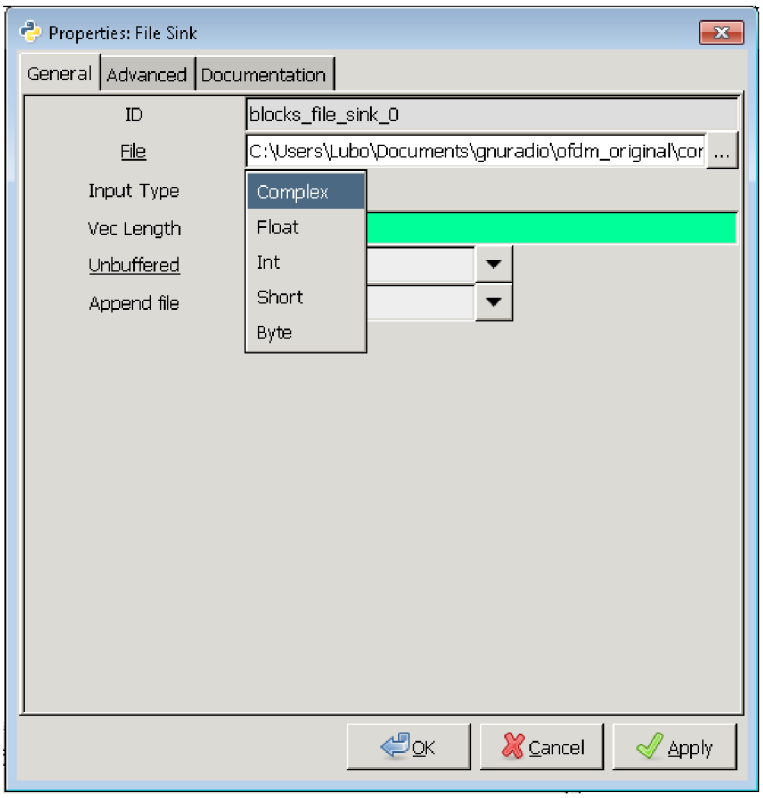
Grafický výstup z tohto programu vidieť nižšie na obr. **??**



Obr. 1.17: Grafický výstup jednoduchého programu v GnuRadiu

1.12.1 Výstup dát z gnuradia

Pri hľadaní chýb v gnuradiu, keďže sa jedná väčšinou o programy spracujúce signál používať výstupy do súborov a ich následnú analýzu. V prípade OFDM signálu sa jedná o komplexný signál a preto je dobré vedieť ako tento signál vyčítať a zobraziť v Matlabe. Zoberme si jednoduchý program generujúci komplexný sínusový signál. Zostavíme jednoduchý program, ktorý bude generovať tento signál a ukladať jednotlivé vzorky do súboru. Blok *Throttle* v programe 1.16 zabezpečuje aby rýchlosť programu neprekročila stanovenú medz, bez neho by mohol nastať nekontrlovateľne rýchly zápis do súboru čo by pravdepodobne skončilo zablokovaním ostatných procesov a pádom celého operačného systému ak by program pokračoval dostatočne dlho. Dvojklikom na blok súboru sa nám zobrazí dialógové okno s možnosťami nastavenia 1.18.



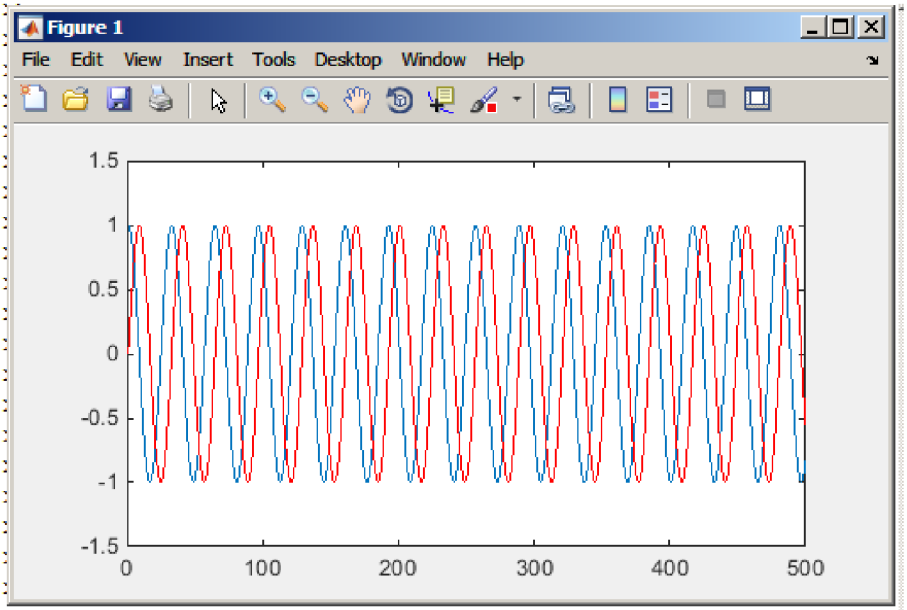
Obr. 1.18: Nastavenie výstupu komplexného signálu do súboru

V lokálnej zložke sa po po skončení programu vytvorí zadaný súbor *complexSinewave.dat* ktorý obsahuje reálne a imaginárne čísla uložené v binárnom formáte ako 2 *float32* uložené za sebou, prvý predstavuje reálnu zložku a druhé číslo imaginárnu zložku.

Pretože vyčítavanie komplexných dát je pomerne častá operácia, vytvoríme si za týmto účelom funkciu *arrayToComplex* ktorej vstupom je vektor dát zo súboru a výstupom komplexný vektor.

```
function y = arrayToComplex(inArray)
    inArray = reshape(inArray,2,length(inArray)/2);
    y = inArray(1,:) + 1i*inArray(2,:);
end
```

Komplexný sínusový signál si môžme teraz zobraziť nasledovným spôsobom 1.19




```
>>
>> f = fopen('complexSinewave.dat');
>> samples = fread(f,2*500,'float32');
>> samples = arrayToComplex(samples);
>> figure;
>> plot(real(samples));
>> hold on;
>> plot(imag(samples),'-r');
>> hold off
```

Obr. 1.19: Zobrazenie komplexného sínusového signálu zo súboru v prostredí Matlab

1.13 Meranie časovej náročnosti programu v Matlabe

Počas experimentovania so spracovaním signálu je namieste pýtať sa aká je časová náročnosť danej metódy alebo algoritmu. Najlepšie je zmerať priamo čas potrebný na vykonanie daných operácí nad množinou dát. Pre tento účel je možné použiť funkcie Matlabu *tic* a *toc*. Funkcia *tic* zaznamená aktuálny čas a následné použitie *toc* vráti čas v milisekundách od použitia poslednej funkcie *tic*. Tieto dve funkcie umiestnime do programu na zvolené miesta programu a zistíme tak čas ktorý potrebuje časť nášho programu v Matlabe na výpočet:

```
tic;
for k = 1:length(someArray)
    do some stuff...
end
```

```
benchmark = toc;
```

1.14 Princíp výpočtu CRC

Výsledok výpočtu CRC je určený vstupnou postupnosťou bitov (polynóm *M(x)*) a zvoleným kľúčom *G(x), tiež postupnosť bitov*. Dostaneme ho ako zvyšok po delení polynómu *M(x)* polynómom *G(x)*. Tento výsledok interpretujeme ako bitovú postupnosť.

CRC má nasledujúce vlastnosti:

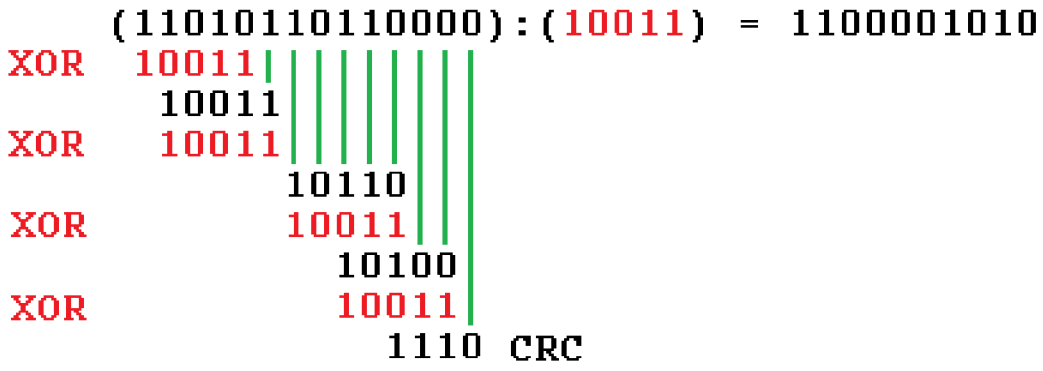
- Schopnosť detekcie chýb záleží na voľbe kľúča, dlhší kľúč umožňuje odhaliť viac chýb
- Existuje viac algoritmov na výpočet CRC, z historických dôvodov dochádza v niektorých algoritmov k zámene jednotlivých bajtov, otočeniu poradia bitov alebo pridávaniu bitov na začiatok či koniec vstupnej bitovej postupnosti
- Pretože CRC je založené na delení vstupnej dátovej postupnosti polynómom, nedokáže rozoznať nuly na začiatku vstupnej postupnosti a tým pádom sa pridáva na začiatok 1.

V CRC aritmetike sú operácie sčítania, odčítania definované nasledovne, nezáleží na poradí argumentov:

operácia	A	B	výsledok
+	0	0	0
+	1	0	1
+	1	1	0 bez prenosu 1 na vyšší bit
-	0	0	0
-	1	0	1 bez prenosu na vyšší bit
-	1	1	0

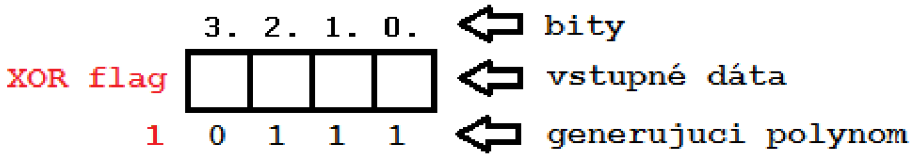
Delenie v CRC aritmetike predstavuje funkcia *XOR*.

Tu je ukážka jednoduchého výpočtu CRC.



Obr. 1.20: Výpočet CRC zo vstupnej bitovej postupnosti

Implementácia CRC v reálnom svete používa posuvný register do ktorého sú načítané vstupné bity a prevádza sa XOR s hodnotami zvoleného generujúceho polynómu. Tento proces je zobrazený na 1.21 Postup výpočtu podľa obrázku 1.21 je



Obr. 1.21: Výpočet CRC za pomoci posuvného registra

nasledovný:

1. Vzorky sa na čítajú do posuvného registra
2. Vypočíta sa XOR medzi hodnotami v posuvnom registri a generujúcim polynómom
3. Nastane posun registra o jeden bit a výstupný bit určí či sa v ďalšom krkou prevedie operácia XOR
4. Toto sa opakuje kým sa nenačíta celá dátová postupnosť

Pri výpočte CRC z dlhej vstupnej postupnosti by tento spôsob zabral mnoho času. Preto sa v praxi používa riešenie s tabuľkou, v ktorej sú uložené hodnoty CRC vypočítané dopredu a CRC sa následne počíta nad blokmi vstupnej postupnosti, pričom dĺžka bloku je vopred zvolená.

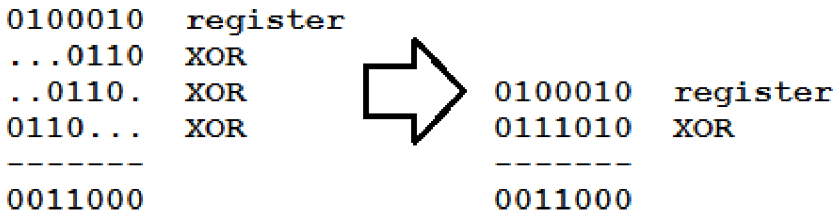
Pri použití metódy s tabuľkou sa využívajú viaceré vlastnosti funkcie XOR.

- Ak použijeme XOR funkciu na výpočet CRC pričom generujúci polynóm je nižšieho rádu ako dĺžka bloku pre ktorý počítame CRC nemusíme deliť vstupnú postupnosť po krokoch a posúvať register ale môžeme priamo spočítať XOR

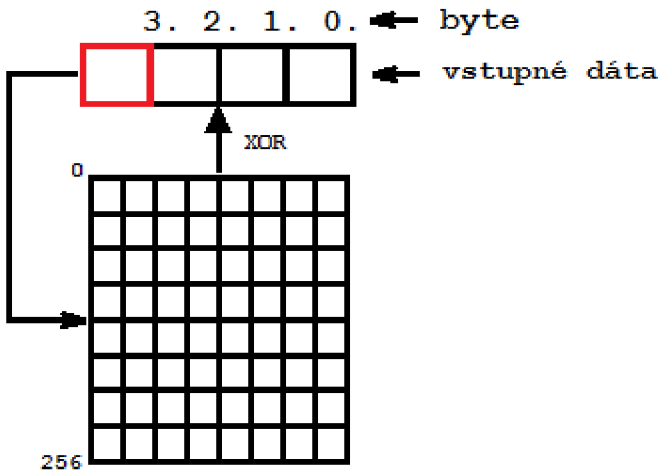
medzi blokom bitov a hodnotou rovnajúcou sa súčtu posunutého polynómu v jednotlivých krokoch ako je vidieť na 1.22

- Hodnota najvýznamnejšieho bitu v posuvnom registri po k iteráciách môže byť vypočítaná z horných k bitov.

Pri uplatnení týchto vlastností môžeme použiť vrchný byte v posuvnom registri ako index do tabuľky s pripravenými hodnotami pre výpočet funkcie CRC a tým pádom počítať kontrolný súčet po blokoch čo znázorňuje 1.23. Podrobnejšie informácie ohľadom postupu výpočtu CRC je možno čerpať z [5].



Obr. 1.22: Nahradenie krokov pri XOR vstupnej postupnosti jedným krokom



Obr. 1.23: Výpočet CRC po blokoch, horny byte posuvného registru predstavuje index do tabuľky s predpripravenými hodnotami CRC

Implementácia výpočtu CRC8 a CRC32 pre hlavičku a dáta paketu je v súboroch *gnuradioCRC8.m* a *gnuradioCRC32.m*.

2 VÝSLEDKY ŠTUDENTSKEJ PFRÁCE

2.1 Prerekvizity

2.1.1 Matlab a USRP rádio

Tento program bol vytvorený za použitia nasledujúcich nástrojov:

- Matlab 2014, v škole na počítačoch Matlab 2013
- Communications System Toolbox, Version 5.3 (R2012b) 20-Jul-2012 (minimálne verzie 5.0)
- DSP System Toolbox
- USRP® Support from Communications System Toolbox
- USRP N200, LFRX, LFTX frontend

2.2 Inštalácia

Rozhranie UHD vyvíjané firmou Ettus zabezpečujce komunikáciu medzi Matlabom a USRP rádiom je dostupná v balíčkoch, ktoré môžeme nainštalovať priamo z prostredia Matlabu. Za týmto účelom sa musíme registrovať na stránke Mathworks, balíček je dostupný zdarma, pretože využíva UHD driver písaný v jazyku C, ktorý je licencovaný aj pod GPLv3, takže driver k rádiu je zdarma. Musíme nainštalovať nasledujúce toolkity:

- Matlab 2014 alebo minimálne 2013
- Communications System Toolbox, version 5.7
- Signal Processing Toolbox, version 6.22
- USRP® Support from Communications System Toolbox

Použité oficiálne dostupné verzie súborov pre FPGA s Matlabom 2013:

- 003.005.001-vendor
- 003.007.001-vendor

Pre Matlab 2014:

- 003.007.001-vendor

Najlepšie je nahrať firmware do USRP priamo príkazom z Matlabu:

```
sdruload('Device', 'n200r3');   % n200r3 n200r2
```

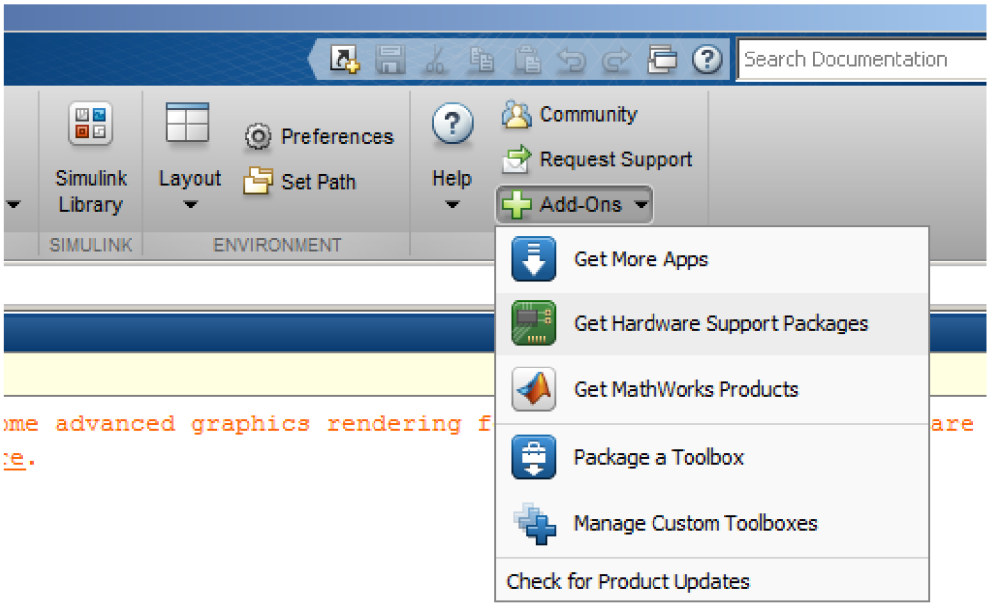
Ďalšie užitočné príkazy pri práci s USRP v Matlabe

```
probesdru();
findsdru();
```

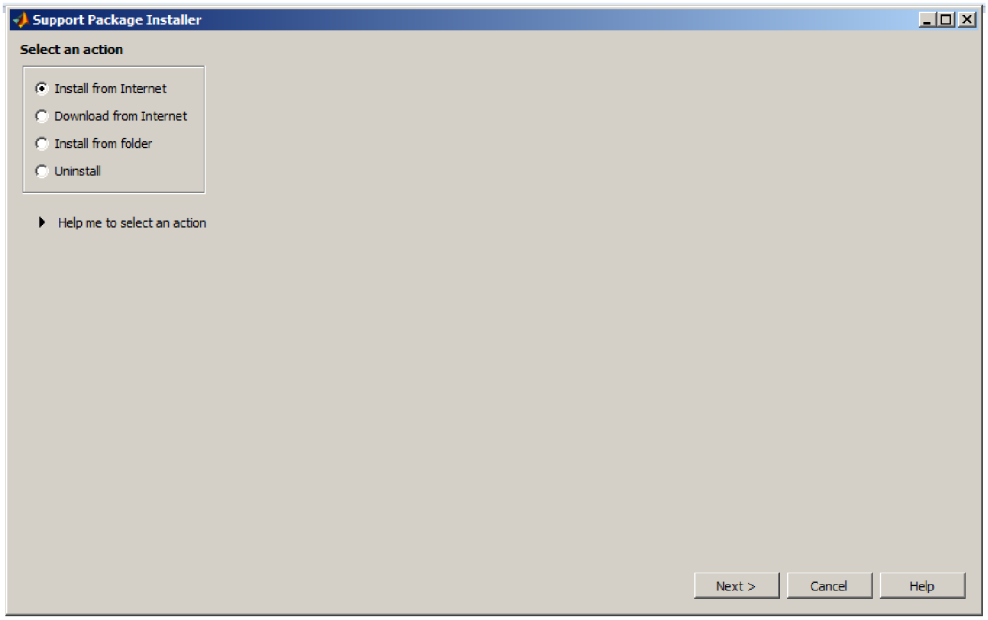
Tu je nutné poznamenať že aj napriek tomu že Matlab píše v nápovede dostupnosť aj pre n200r4 nepodarilo sa rozchodiť túto variantu a podpora n200r4 je sporná. Je dôležité mať správnu revíziu zariadenia, pretože medzi revíziou 3 a 4 sú pridané a pozmenené signály v FPGA a tým pádom FPGA pracuje s nesprávnym imageom a pracuje nespoľahlivo.

Pri realizácií bolo použité USRP N200r3 s modulmi RF2400 a XCVR2450 a RFX2400.

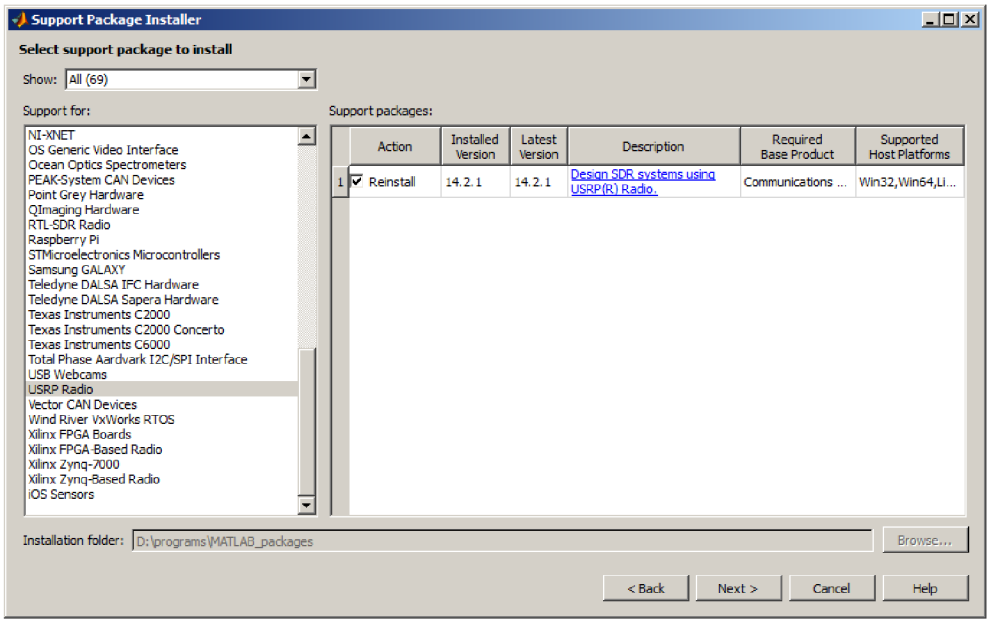
Toolkit pre USRP sa inštaluje priamo z prostredia Matlabu, je nutná registrácia na účet v MathWorks. Kroky na inštláciu vidieť na 2.1,2.2,2.3, 2.4



Obr. 2.1: Inštalácia USRP toolkitu, krok 1

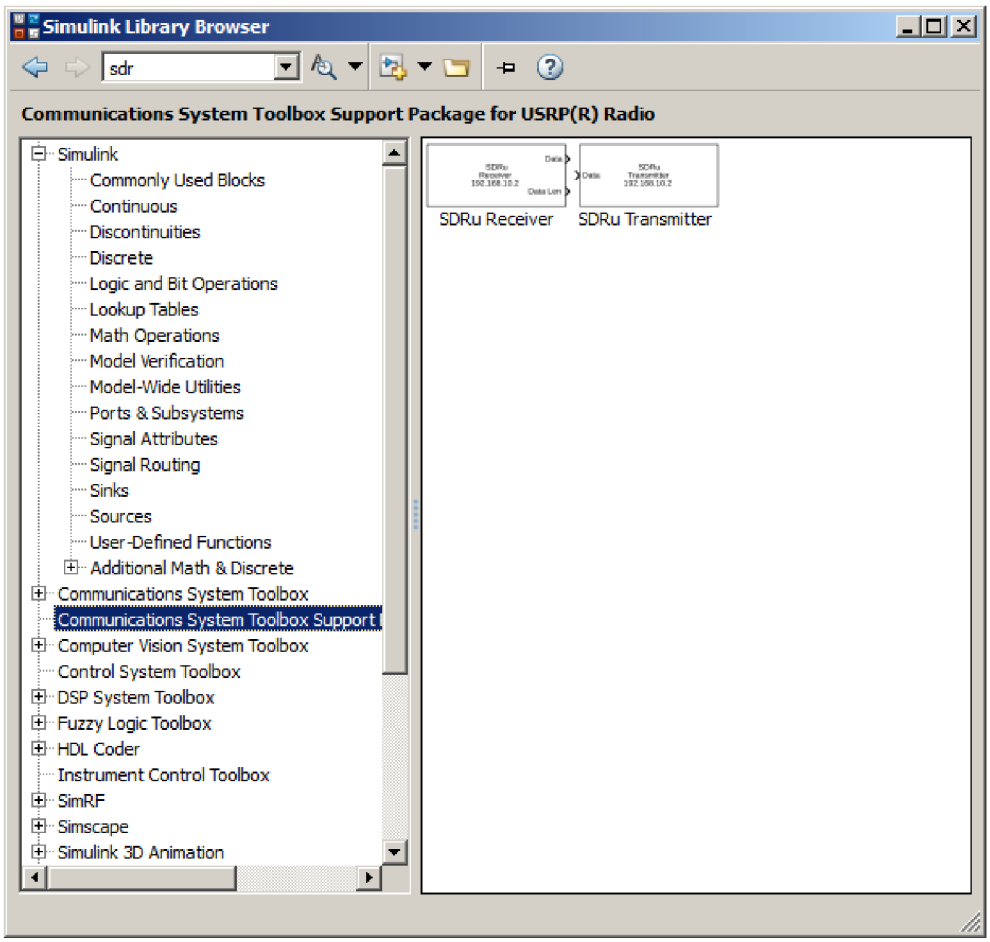


Obr. 2.2: Inštalácia USRP toolkitu, krok 2



Obr. 2.3: Inštalácia USRP toolkitu, krok 3

Po inštalácií sa zobrazí blok pre USRP rádio v knižnici.

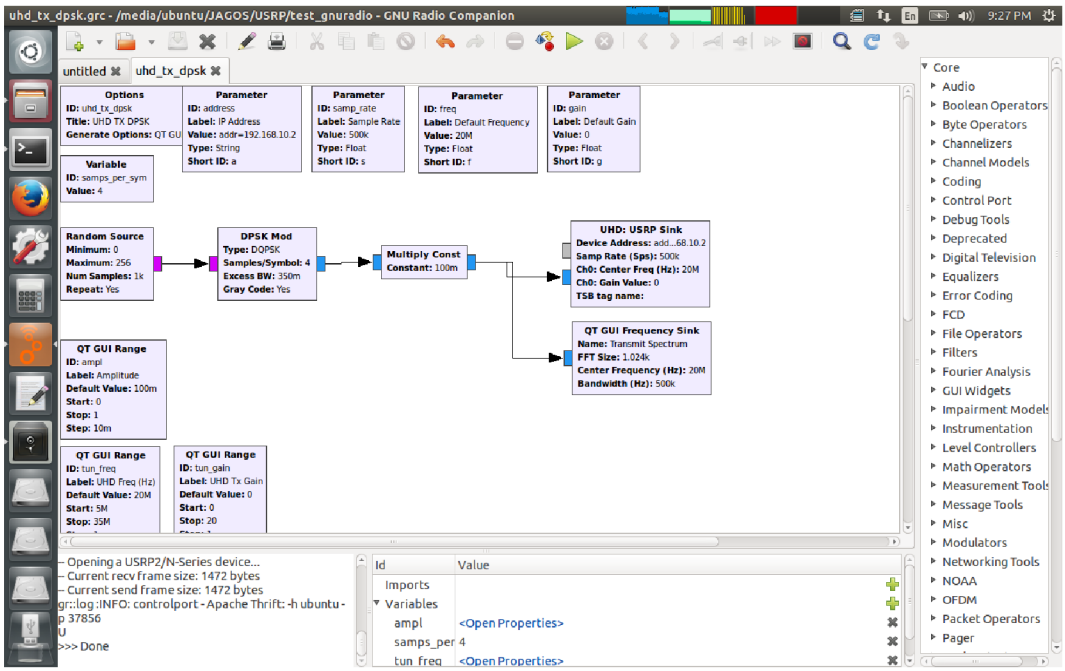


Obr. 2.4: Inštalácia USRP toolkitu, krok 4

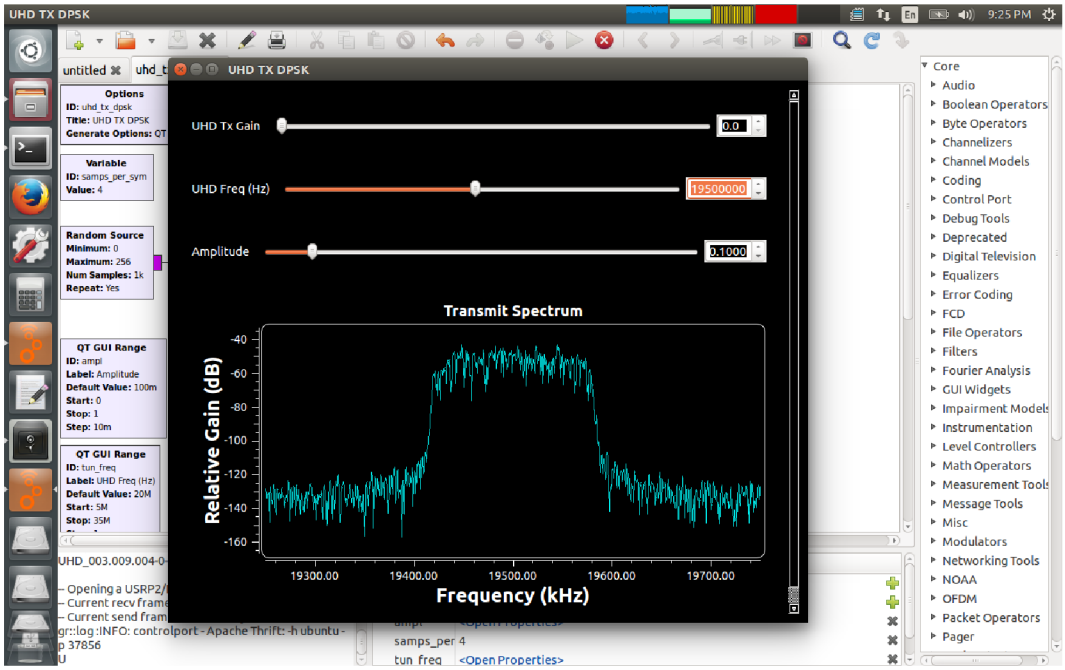
2.2.1 Gnuradio a USRP rádio

Gnuradio je možné stiahnuť ako LiveUSB SDR a nahrať na USB flash disk a použiť v ľubovoľnom 64 bitovom počítači. LiveSDR je Gnuradio nainštalované na Ubuntu 14.04, toto riešenie prináša značné výhody, nemusíme nič inštalovať a na vyhradenej veľkosti na disku môžme vytvárať programy.

Tiež bola vyskúšaná funkčnosť s gnuradiom a vysielanie pomocou OFDM transceiver bloku. Nastavená šírka pásma *cca 400kHz*



Obr. 2.5: Schéma OFDM TX v gnuradio



Obr. 2.6: Nastavenie parametrov pre vysielanie v gnuradio

2.2.2 Vzorkovanie signálu, USRP rádio

Výstupné ADC prevodníky na 1.13 vzorkujú na pevne danej frekvencii 100Ms/s a k tejto frekvencií sa vzťahuje parameter interpolation pri nastavovaní parametrov USRP rádia. V programe sa zadáva vzorkovacia frekvencia symbolov (samplerat) a interpolate sa vypočíta ako $interp. = \frac{100MHz}{f_s}$ Je dobré prevzorkovať aj v stupný signál z Matlabu do USRP, aby signál nezaberal celé základné pásmo a nedosahoval k medznej frekvencií výstupného filtru, za týmto účelom slúži v parametroch USRP rádia parameter *oversampling*.

Vzorkovaciu frekvenciu hodín USRP získame vieme zistiť pomocou príkazu *get(Tx,'MasterClockRate')*, pričom Tx je deklarovaný ako:

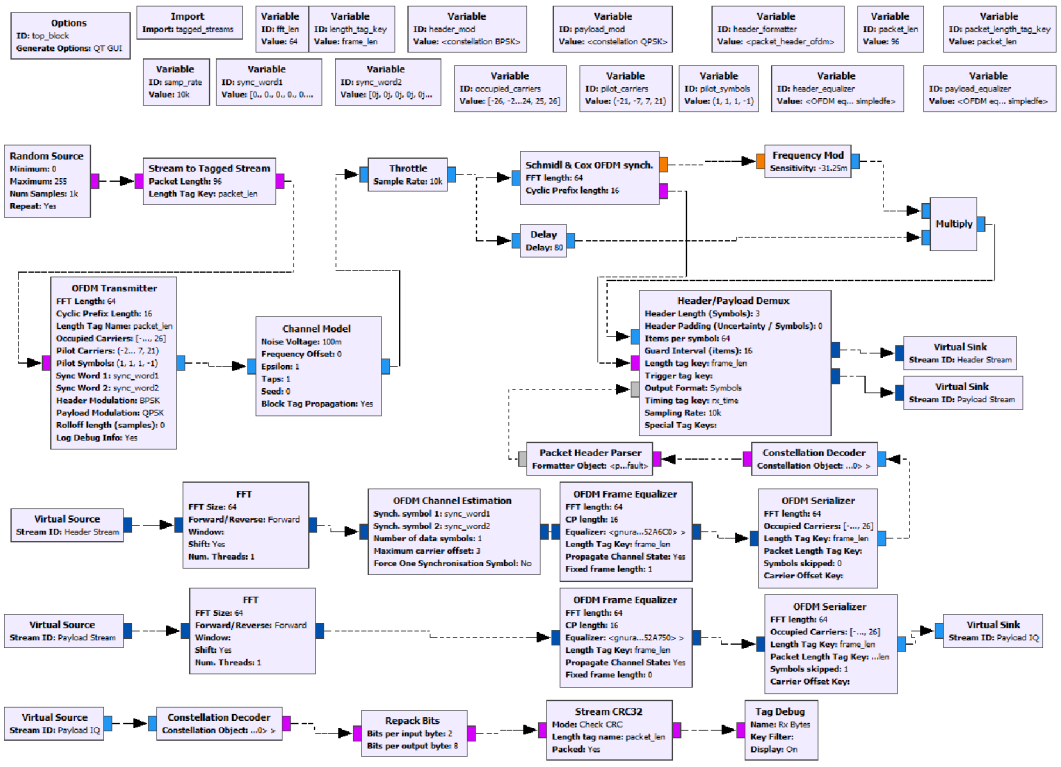
$$Tx = comm.SDRuTransmitter()$$

2.3 Dátový prenos pomocou OFDM modulácie

OFDM moulácia sama o sebe poskytuje len fyzickú vrstvu na prenos dát teda rieši vzorkovanie dát, modulácie a demodulovanie signálu, ale nešpecifikuje akým spôsobom tieto dáta máme prenášať. Aby sme mohli preniesť dáta použitím OFDM modulátoru, musíme si nad touto fyzickou vrstvou vytvoriť prenosovú vrstvu, ktorá zabezpečí delenie dát do paketov, ich príjem v správnom poradí a kontrolu integrity dát.

Gnuradio obsahuje funkčný príklad OFDM 2.7, ktorý má implementovanú synchronizáciu pomocou Schmidl a Cox algoritmu, realizuje odhad kanálu a ekvalizáciu rámcov a tiež obsahuje implementovanú základnú prenosovú vrstvu. Ak teda dodržíme tento samodohodnutý protokol a budeme dáta modulovať podľa neho, musíme na výstupe dostať rovnakú postupnosť. Riešenie presnej synchronizácie pre digitálny systém vlastnými silami je pomerne náročná úloha a preto použijeme tento softwareový demodulátor.

Gnuradio a Matlab majú rozdielnu filozofiu, gnuradio pracuje s tokom dát, pričom sa jednotlivé toky dodatočne označujú tagmi a na základe týchto tagov sú spracvané po častiach v blokoch implementovaných v Pythone alebo C++. Oba použité jazyky sú objektové. Matlab pracuje s maticami .



Obr. 2.7: OFDM demodulátor v GnuRadiu

2.3.1 Python vs Matlab, indexovanie polí

Pri prezeraní blokov v GnuRadiu netreba zabúdať na rozdiely indexov polí v Matlabe a Pythone. V matlabe prvý prvok vektoru leží na indexe 1, zatiaľ čo v Pythone na indexe 0. Tiež je dobré vedieť, že *range(5,9)* (nezahŕňa pravý okraj) v Pythone znamená čísla 5,6,7 a 8 zatiaľ čo v Matlabe 5:9 vyjadruje postupnosť 5,6,7,8,9 (zahŕňa aj pravý okraj). Pri použití FFT o N bodoch zodpovedá kladný index kmitočtu v gnuradiu zväčšenému indexu o jedna v Matlabe (pretože 0 index v Pythone je v Matlabe 1) a 0 alebo záporný index kmitočtu zas zodpovedá v Matlabe kmitočtu s indexom $i + N + 1$, pretože Matlab neumožňuje indexovanie vektoru odzadu použitím záporných indexov (pre N = 64 a i = -3 v Pythone zodpovedá i = 62 v Matlabe).

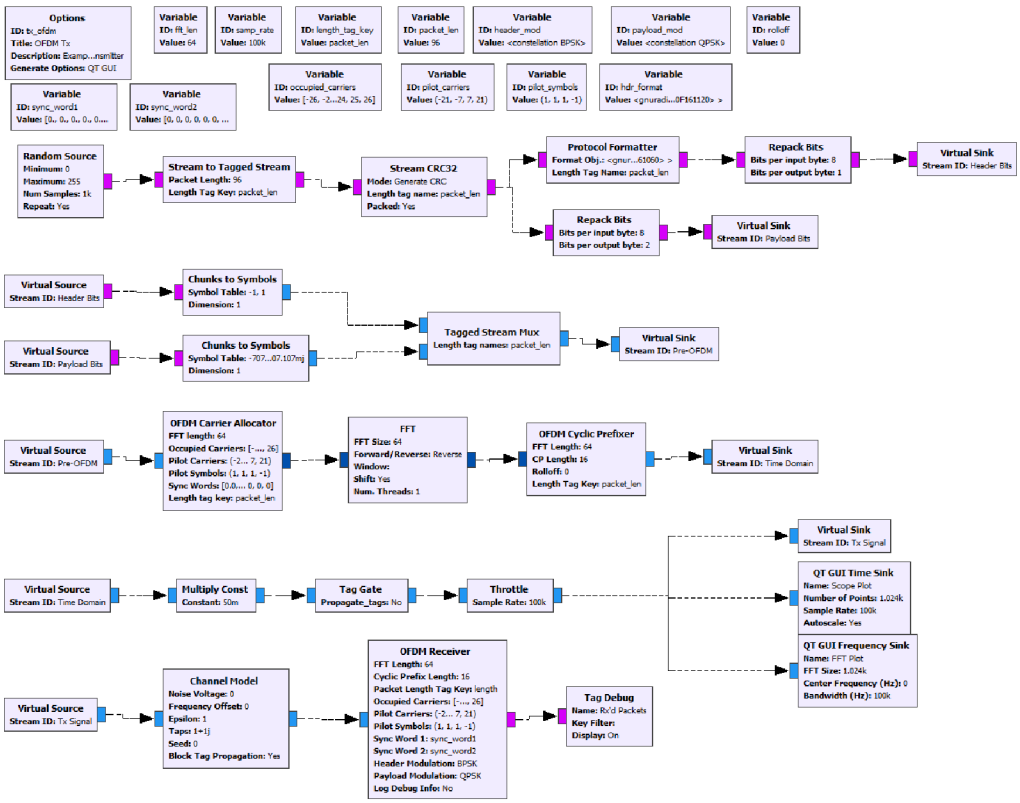
2.3.2 Usporiadanie spektra

Aby sme zachovali konzistentnosť s demodulátorom, spektrum má naturálne usporiadanie, teda jeho rozsah je v rozmedzí: $-\frac{f_s}{2} - \frac{fs}{2}$ pričom nula sa nachádza v strede

spektra.

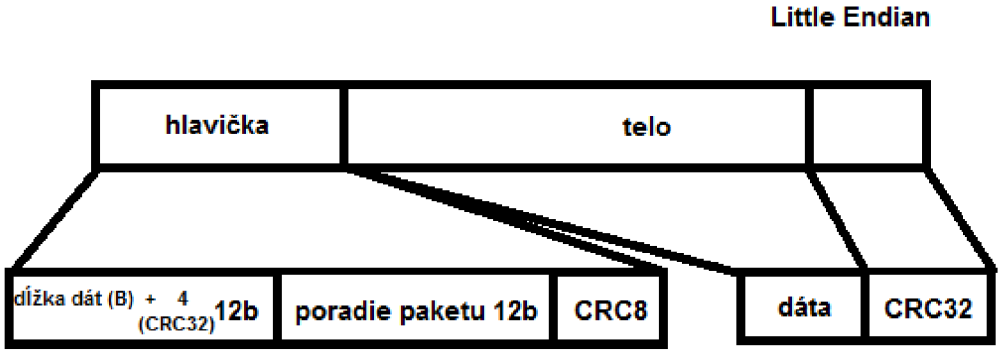
2.3.3 Šturktúra a vytváranie paketov

Spôsob akým sa pakety vytvárajú, zistíme v modulátore 2.8 a ďalšie informácie musíme nájsť v dokumentácií.



Obr. 2.8: OFDM modulátor v GnuRadiu

Z dokumentácie na internete sa dozvieme že paket v OFDM v Gnuradiu má nasledovnú štruktúru:



Obr. 2.9: Štruktúra paketu

Je dôležité ešte raz pripomenúť, že usporiadanie bitov je Little Endian, tým pádom je LSB uložené na začiatku súboru. Tento fakt je dôležitý pri analýze výstupov z jednotlivých blokov v OFDM. Bohužiaľ nikde nie je uvedený generujúci polynóm pre CRC8 a CRC32, tie musíme nájsť až v zdrojových súboroch GnuRadia a to konkrétne:

- *header_format_crc.cc*
- *header_format_crc.h*

V nich nájdeme jednotlivé riadky deklarujúce objekt na výpočet CRC a s týmito informáciami už sme schopný zostaviť si algoritmus pre jednotlivé CRC:

- pre CRC32 je to riadok: $boost :: crc_optimal < 32, 0x04c11db7, 0xFFFFFFFF, 0xFFFFFFFF, true, true >$
- pre CRC8: $boot :: crc_optimal < 8, 0x07, 0xFF, 0x00, false, false >$

Jedná sa o objekt z knižnice boost, ich význam je dĺžka CRC, generujúci polynóm, iniciačná hodnota CRC, hodnota pre záverečný XOR, otočenie poradia bitov vstupnej postupnosti, otočenie poradia výstupnej postupnosti.

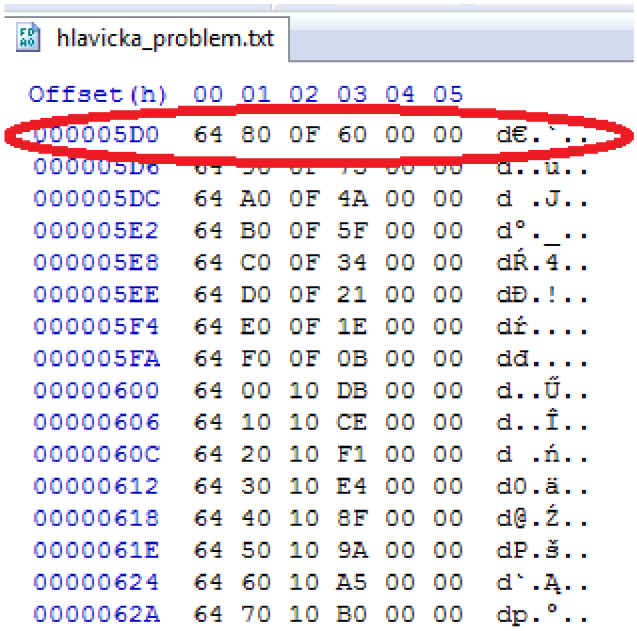
2.3.4 Hlavička paketu

Pri skúmaní štruktúry hlavičky bolo zistené skúmaním tvorby paketov v OFDM modulátore že hlavička paketu nemá vždy konštantnú dĺžku, ale je niekedy doplnená nulami ale jej dĺžka je závislá od počtu použitých nosných, empiricky overené pravidlo pri použití FFT do dĺžky 128:

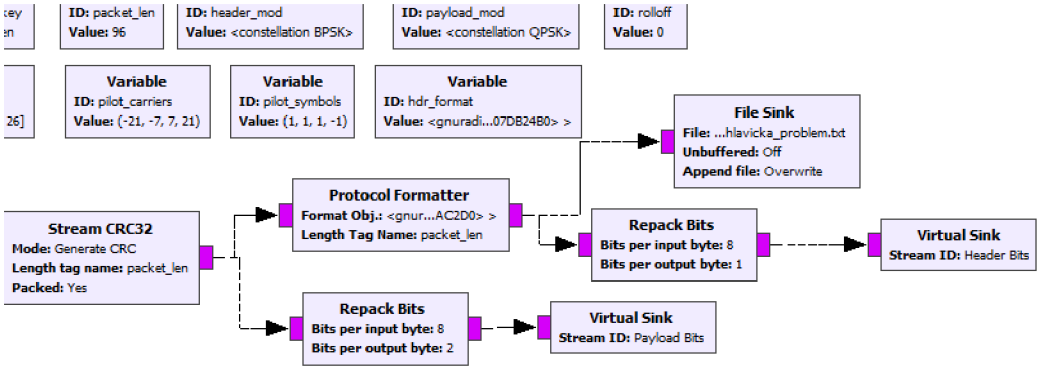
$$headerLen = 8 * floor(nCarriers/8); \tag{2.1}$$

Pri použití FFT 256 za namiesto *floor()* musíme použiť *ceil()*. Tento problém nebol vyriešený ani úplne dohľadaný, najviac bolo používané FFT do 128 bodov. Spôsob

akým je hlavička uložená v súbore je na 2.10, prvé číslo je dĺžka paketu v tomto prípade 100 (96B dáta + 4B CRC) a druhé poradie paketu, obe dlhé 12b a nasleduje CRC8.



Obr. 2.10: Funkčná chyba generovania hlavičky pre OFDM rámec v GnuRadiu



Obr. 2.11: Zdroj chyby generovania hlavičky pre OFDM rámec v GnuRadiu

Túto chybu 2.10 2.11 čo je vlastne doplnenie nulami musíme zaniesť aj do nášeho programu ak chceme aby fungoval s demodulátorom, pretože v inom prípade budeme na vstupy modulátorov pre hlavičku privádzať kratšiu postupnosť a po začlenení pred dáta a vykonaním IFFT sa bude chyba len kopiť a kopiť. Vo výsledku bude náš paket zlý a v prijímači zahodený.

2.3.5 Alokátor kmitočtov

Ďalšia potrebná súčasť reťazca, ktorú musíme implementovať je alokátor kmitočtov. Alokátor sa stará a zostavenie OFDM rámcu ktorý pozostáva:

- z preambuly tvorenej dvoma synchronizačnými slovami pred každým rámcom za účelom synchronizácie v demodulátore
- jednotlivých symbolov o dĺžke N (ktorá je daná počtom bodov IFFT) tvorených modulovanou dátovou postupnosťou (v našej implementácií BPSK, QPSK alebo PSK8) a pilotných tónov

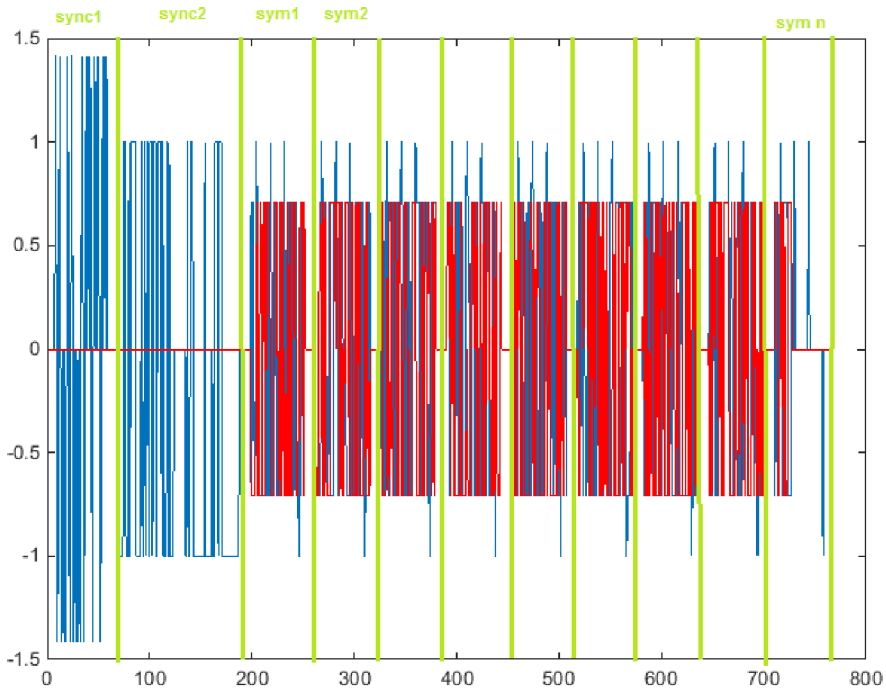
Alokátor najprv zoberie vstupné synchronizačné slová a zaradí ich na začiatok výstupného vektoru. Potom zoberie polohy dátových kmitočtov a začne na ich pozície radiť modulované dáta. Toto vykoná N-krát a ak stále nevyčerpal všetky modulované dáta, zaradí na koniec nový symbol a opäť začne priradzovať dáta od najnižšieho nosného kmitočtu (v tomto prípade zľava) Ak už priradil všetky modulované dáta zvyšok nosných doplní nulami. Následne priradí na príslušné kmmitočty pilotné symboly. Ak nepriradil všetky modulované signály na nosné započne nový symbol. Dĺžka rámca je teda závisla od počtu dodaných vzorkov, ale keďže pracujeme stále s tou istou dĺžkou paketov, mal by byť konštantný.

Alokátor nosných kmitočtov je implmentovaný vo funkcií *allocateCarriers.m*, tu je jej ukážka, funkcia očakáva na svojom vstupe indexy nosných kmitočtov a pilotov v Matlabovskom spôsobe (DC zložka na indexe 1, f_{max} na index N):

```
function frame = allocateCarriers(  ...
    data,                           ...
    fftLen,                         ...
    packetLen,                      ...
    nProcessPackets,                ...
    occupiedCarriers,               ...
    pilotCarriers,                  ...
    pilotSymbols,                   ...
    sync1,                          ...
    sync2)

nCarriers = length(occupiedCarriers);

frame = [];
readIndex = 1;
dataEnd = false;
while (nProcessPackets > 0 && ~dataEnd)
    frame = [frame sync1 sync2];
    needToRead = packetLen;
    while (needToRead > 0 && ~dataEnd)
```

```
            stopIndex = 0;
            if (needToRead > nCarriers)
                stopIndex = readIndex + nCarriers - 1;
                count = nCarriers;
            else
                stopIndex = readIndex + needToRead - 1;
                count = needToRead;
            end
            if (stopIndex <= length(data))
                dataIn = data(readIndex:stopIndex);
                readIndex = stopIndex + 1;
            else
                dataIn = data(readIndex:end);
                count = length(data) - readIndex;

                dataEnd = true;
                needToRead = 0;
            end

            if (count == nCarriers)
                needToRead = needToRead - count;
            else
                dataIn = [dataIn zeros(1,nCarriers-count)];
                needToRead = 0;
            end
            fftFrame = zeros(1,fftLen);
            fftFrame(occupiedCarriers) = dataIn;
            if (~isempty(pilotCarriers))
                fftFrame(pilotCarriers) = pilotSymbols;
            end
            fftFrame = circshift(fftFrame', floor(fftLen/2))';
            frame = [frame fftFrame];
        end
        nProcessPackets = nProcessPackets-1;
    end
    end
```

Výstup z alokátoru je na 2.12. Pretože jednosmerná zložka je umiestnená v strede symbolu (spektrum je prirodzene usporiadané - má záporné a kladné kmitočty), zapĺňame rámec vždy ako pole zľava a po zaplnení toto pole zrotujeme o $\frac{N_{frame}}{2}$ Na zaičiatku možno vidieť pridané synchronizačné slová prvé modulované QPSK a druhé BPSK moduláciou. Za nimi nasledujú jednotlivé OFDM symboly, pričom kmitočet s DC zložka sa nachádza uprostred nich a nasledujúci nosný kmitočet sa nachádza smerom vpravo.



Obr. 2.12: Výstupný rámec z alokátoru kmitočtov s vyznačenými synchronizačnými slovami a jednotlivými symbolmi.

2.3.6 IFFT

IFFT blok v GnuRadiu funguje ako IFFT škálované jeho dĺžkou. VV našej implementácií tiež výsledok IFFT násobíme jeho dĺžkou.

2.3.7 Zistenie mapovania modulátorov

Jedna z posledných vecí ktorá môže robiť problém zistiť je správne mapovanie modulátorov. Na zistenie ako sú usporiadané konštelačné body jednotlivých modulátorov, je najjednoduchšie otvoriť nejaký top blok programu ktorý používa danú moduláciu v Python IDE (alebo príkazovej riadke) a spustiť krokovanie programu a v momente keď modulátor bude načítaný v pamäti, prezrieť si jeho obsah alebo si nechať zobraziť jeho konštelačné body ako idú v rade za sebou.

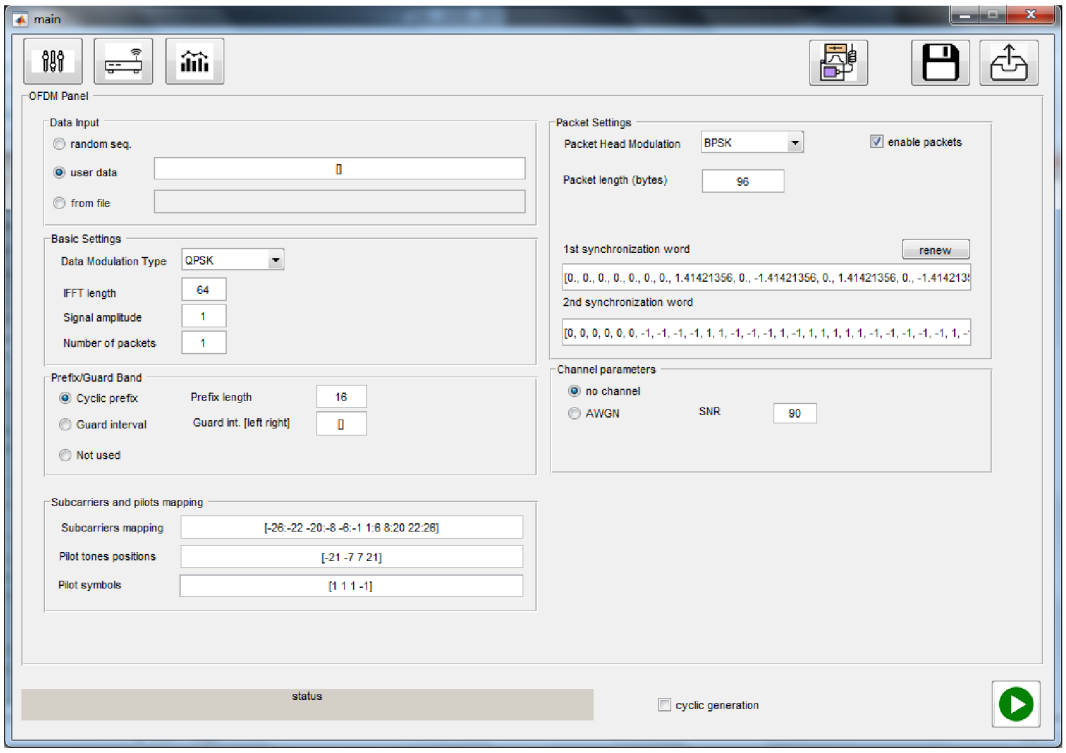
2.4 Ukážka výsledného OFDM generátoru

Na nasledujúcich obrázkoch 2.13 2.14 2.15 vidieť kompletný program. Do jednotlivých panelov sa dá dostať kliknutím na tlačidlá v ľavej hornej časti. Zľava doprava

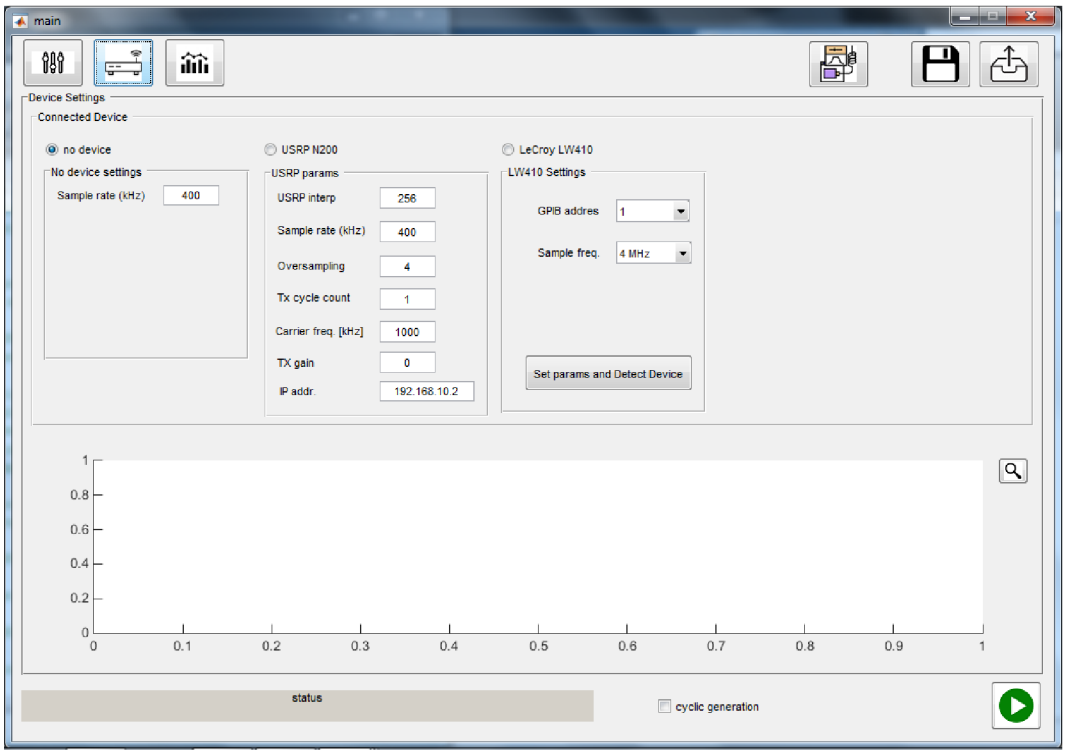
to sú:

- Nastavenie parametrov OFDM signálu
- Nastavenie výstupného zariadenia
- Zobrazenie výsledkov

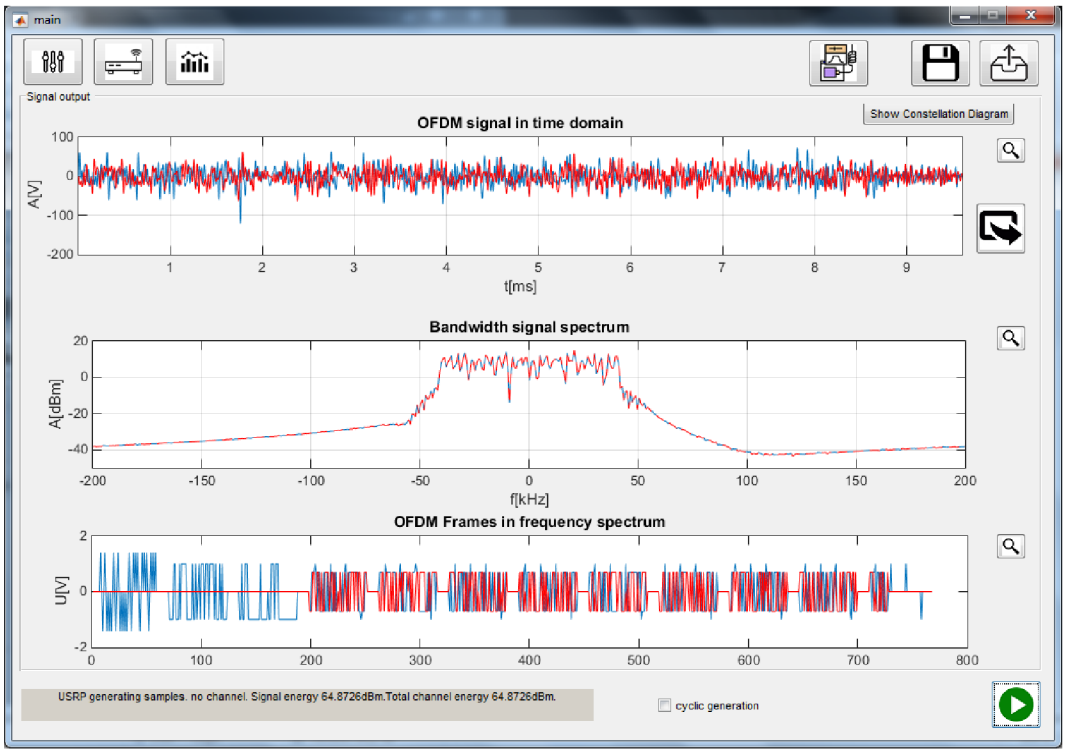
Tlačidlá v pravej hornej časti slúžia na uloženie aktuálne nastavených hodnôt a ich načítanie z užívateľom zvoleného súboru. Na spodku okna vidieť statusbar, do ktorého program vypisuje informácie o generovanom signále a informácie pre užívateľa. Generovanie sa spustí stlačením šípky v zelenom kruhu.



Obr. 2.13: Nastavenie OFDM parametrov



Obr. 2.14: Nastavenie výstupného zariadenia

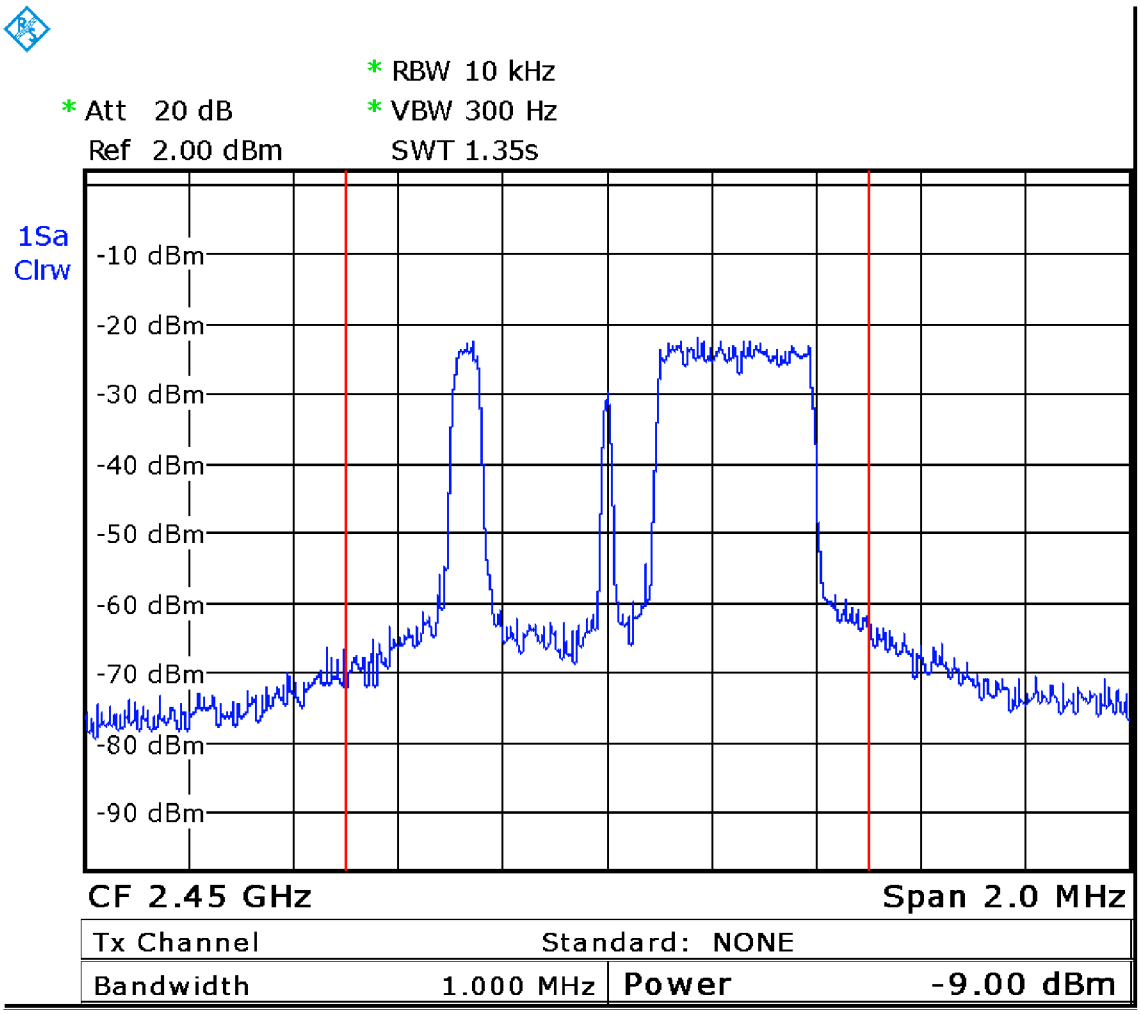


Obr. 2.15: Zobrazenie výstupu

V hornej časti je ešte jedno nespomenuté tlačidlo. Toto tlačidlo vygeneruje GnuRadio demodulátor s našimi aktuálnymi nastaveniami a ten stačí nahrať na počítač s GnuRadiom a prenášať pakety aj bezdrôtovo ak máme k dispozící moduly s príslušným frekvenčným rozsahom.

Obr. 2.16: Bezdrôtové vysielanie dát pomocou vytvoreného programu na demonštráciu OFDM

Na záver bolo ešte prevedené meranie na spektrálnom analyzátor FSL od Firmy Rhode&Schwarze aby sme sa uistili že hodnoty zobrazované vo vytvorenom programe sú totožné s generovanými. Za týmto účelom sa vysielal OFDM signál bez paketového prenosu s 1024 bodovým IFFT a obsadenými 50 nosnými v pravej časti obrazovky a 150 v ľavej časti obrazovky.



Obr. 2.17: Bezdrôtové vysielanie dát pomocou vytvoreného programu na demonštráciu OFDM

3 ZÁVER

Zadanie hovorí o vytvorení základných funkcí na prenos dát do generátoru LW410 a overení ich funkčnosti. Daný generátor sme použili v počiatkoch riešenia úlohy a neskôr sme ho vymenili za iné zariadenie - USRP rádio, pretože viac vyhovuje požiadavkam úlohy a komunikácia medzi počítačom a týmto zariadením poskytuje viac možností ako dané zariadenie využiť pri riešení úlohy. Grafický program na obsluhu a generovanie signálu bol nakoniec implementovaný v prostredí Matlab najmä z dôvodu že toto prostredie sa vyučuje na univerzite a tým pádom väčšina ľudí už s ním má skúsenosti. Výsledný program umožňuje meniť požadované parametre, ktoré boli v zadaní úlohy. Za účelom overenia funkčnosti systému bola implementovaná paketová vrstva a sfunkčnené posielanie správ medzi dvoma USRP zariadeniami, čo predstavuje krok k simuláciam mnohacestného šírenia medzi vysielačom a prijímačom. Po vzájomnej dohode s vedúcim nebolo riešené rušenie aditívnym šumom, pretože použité zariadenie nie je širokopásmové a tým pádom šum generovaný funkciou, je úzkopásmový, vyfiltrovaný výstupným filtrom zariadenia, táto funkcia bola ponechaná v programe len ako ukážka. Správne fungovanie generátoru OFDM signálu bolo preverené použitím demodulátoru slúžiaceho na príjem paketov a ich zápis do súboru a tiež overením že signál má vo frekvenčnej oblasti správny tvar pomocou spektrálneho analyzátoru.

LITERATÚRA

[1] Multi-carrier and spread spectrum systems: from OFDM and MC-CDMA to LTE and WiMAX – 2nd ed. K. Fazel, S. Kaiser. ISBN 978-0-470-99821-2

[2] *THEORY OF COMMUNICATION* Elektronické prednášky Ing. Radim Číž, Ph.D Brno: FEKT VUT v Brně 2013.

[3] *TEORIE RáDIOVé KOMUNIKACE* ISBN 978-80-214-4503-1 Doc. Ing. Roman Maršálek, Ph.D Brno: FEKT VUT v Brně 2012.

[4] *RÁDIOVÉ PŘIJÍMAČE A VYSÍLAČE* ISBN 80-214-2263-7 Ing. Aleš Prokeš, Ph.D Brno: FEKT VUT v Brně 2005.

[5] Zlib [online]. [cit. 2017-05-16]. Dostupné z: www.zlib.net/crc_v3.txt

[6] Gnuradio Packet Data API [online]. [cit. 2017-05-16]. Dostupné z: https://gnuradio.org/doc/doxygen/page_packet_data.html#packet_data_structure

[7] Ettus Knowledgebase [online]. [cit. 2017-05-16]. Dostupné z: https://kb.ettus.com/Knowledge_Base

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

USRP	Universal Software Radio Peripheral
SNR	Signal to noise ratio
PN	Pseudorandom Noise
CRC	Cyclic Redundancy Check
ISI	Inter Symbol Interference
ICI	Inter Carrier Interference
GPIB	General Purpose Interface Bus
DFT	Discrete Fourier Transform
IDFT	Inverse Discrete Fourier Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
USB	Universal Serial Bus
OFDM	Orthogonal Frequency Division Multiplex
BER	Bit Error Ratio
DLL	Delay Locked Loop
FPGA	Field Programmable Gate Logic
QPSK	Quadrature Phase Shift Keying
DC	Direct Current
IDE	Integrated Development Environment