

Použití softwarově definovaného rádia v leteckých navigačních aplikacích.

Příklad implementace digitálního rádiového vysílače a přijímače

Autor: Petr BOJDA

Dne: 31. listopadu 2017

Verze dokumentu: 2.2

1 Úvod

Hlavní cíl zprávy Zpráva vzniká na podnět výzkumného týmu Katedry Leteckých elektrotechnických systémů k podpoře výzkumného úkolu v oblasti terestriálních navigačních systémů s rozprostřeným spektrem. Cílem je nalézt vhodnou strukturu a parametry signálu tak, aby co nejlépe vyhověl požadavkům na celkový systém. V této fázi bude důraz kladen zejména na co nejlepší autokorelační vlastnosti signálu tak, aby bylo dosaženo co nejpřesnější synchronizace, resp. měření časového zpoždění.

Sekundární cíl zprávy Dalším důvodem ke vzniku této zprávy je snaha demonstrovat, případně i pomoci s přípravou výuky v oblasti digitálních rádiových systémů. Pozornost je věnována hlavně moderním přístupům k návrhu, analýze, verifikaci a implementaci jednotlivých částí digitálního rádiového systému na bázi softwarově definovaného rádia (SDR). Nedílnou součástí zprávy jsou ucelené bloky softwaru (knihovny a aplikace) vytvořené tak, aby jednak plnily požadovanou funkci v rámci návrhu systému, ale zároveň aby dostatečně srozumitelně demonstrovaly způsob psaní daného typu programu a jeho použití.

Použité prostředky Zpráva vznikla jako komentovaný popis a dokumentace návrhu digitálního rádiového systému. Zahrnuje základní kroky návrhu - simulaci, testování (verifikaci) a nakonec i finální implementaci. Cílovou platformou je SDR, tudíž lze předpokládat, že funkce rádiového systému bude popsána, definována a implementována formou bloků software. Veškeré softwarové nástroje a prostředky použité při zpracování této zprávy jsou v kategorii "Open-Source Software License", zejména BSD licence v případě simulačních nástrojů a GNU GPL v případě implementačních nástrojů. Textová část je editována v systému LaTeX. Jednotlivé práce byly odvedeny na počítačích s operačním systémem Linux, distribuce Ubuntu. Nebyly použity žádné softwarové nástroje podléhající komerční licenci.

K simulaci je využit systém knihoven v jazyce Python, zejména pak knihovny NumPy, Matplotlib a SciPy. S využitím funkcí těchto knihoven byly vytvořeny knihovny vlastních funkcí ke generování signálů, jejich modulace v základním pásmu (baseband), jejich konverze do vyššího kmitočtového pásma (up-converter), filtrace, tvarování impulzu s ohledem na minimalizaci mezisymbolové interference při přenosu signálu kanálem s omezeným kmitočtovým pásmem (ISI - Inter-Symbol Interference) a další. Jak bylo uvedeno výše, všechny simulace jsou psány v programovacím jazyce Python.

Prvky digitálního rádiového systému jsou implementovány do SDR E310 firmy Ettus a/nebo s pomocí levného USB přijímače DVB-TV signálu s čipem RTL2832U. Implementace je ve většině případů provedena pomocí softwarového balíku GNU Radio. Některé části jsou pak implementovány přímo za použití knihovních funkcí výrobce rádia E310, tzv. API funkcí v jazyce C++.

2 Struktura digitálního rádiového systému

Tato práce se zabývá obecným digitálním rádiovým systémem. V leteckých aplikacích jsou rádiové systémy použity ve třech základních oblastech: komunikační, navigační a přehledové. Podle toho, jaký je primární účel systému, mění se i jeho složení a nastavení parametrů. Společná je jejich základní struktura:

[Složení systému](#)

- Vysílač
- Přenosový kanál
- Přijímač

V dalším budou analyzovány tyto tři komponenty oprostěné od funkcí specifických pro dané oblasti použití. To znamená, že nebude zahrnuto například různé kódování přenášené bitové posloupnosti (zdrojové či kanálové) a podobně. Vstupem vysílače je logický signál. Ten je modulován některou z mnoha digitálních modulací na nosný harmonický signál a anténou vyslán do prostoru. Přenosový kanál reprezentuje souhrnně vliv prostředí (přenosového média) na přenášený signál. Jako příklad je možno jmenovat: přidaný šum, útlum, vícecestné šíření, dopplerův posun, interference, omezení šířky pásma přenášeného signálu a jiné. Zkreslený signál je poté detekován a zpracován pomocí přijímače.

2.1 Vysílač

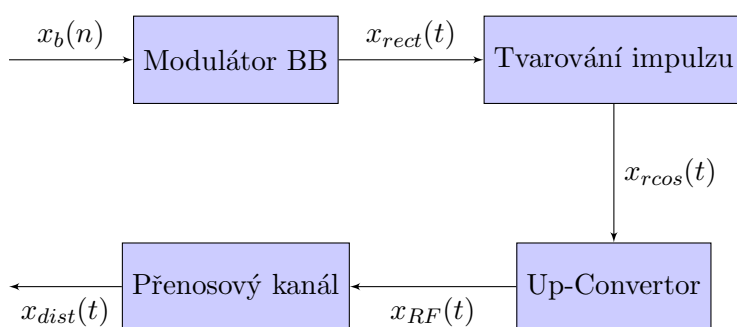
Struktura vysílače, jak bude v následném textu chápána a analyzována je na obr. 1. Vstupní signál $x_b(n)$ je bitovou posloupností nesoucí požadovanou informaci. Blok *Modulátor BB* přemění – moduluje vstupní bitovou posloupnost na komplexní signál v základním pásmu (angl. *baseband*). Signál v základním pásmu $x_{rect}(t)$ pomocí aktuální úrovně své reálné a imaginární složky precizně definuje amplitudu a fázi nosné složky budoucího výstupního signálu vysílače $x_{RF}(t)$. To je již signál s harmonickou nosnou, na kterou vybranou digitální modulací namodulována bitová posloupnost $x_b(n)$.

[Popis struktury vysílače](#)

SDR (ať už simulované, nebo reálně využívané v této zprávě) používá k transformaci (konverzi) ze základního kmitočtového pásma do pásma nosného kmitočtu (angl. *passband*) obvod kmitočtové přeměny (tzv. směšovač) *Up-Convertor* typu kvadrurní modulátor. Tento typ obvodu vyžaduje komplexní vstupní signál v základním pásmu. Komplexní signál v základním pásmu je tvořen reálnou (soufázní, angl. *In-Phase*) a imaginární (kvadrurní, angl. *Quadrature*) složkou¹. Blok *Modulátor BB* tvaruje ze vstupní posloupnosti $x_b(n)$

[Baseband modulace – signál v základním pásmu](#)

¹Soufázní složka signálu bývá v technické praxi často označována jako signál I, kvadrurní pak jako Q.



Obrázek 1: Základní struktura digitálního rádiového vysílače.

komplexní $x_{rect}(t)$ signál, který je charakterizovaný svou tzv. bitovou (symbolovou) rychlostí (angl. *bitrate*)² a amplitudou. "Modulátor BB" je již navržen konkrétně pro daný typ výsledné digitální modulace signálu na výstupu vysílače $x_{RF}(t)$. Přicházející bity vstupní posloupnosti $x_b(n)$ jsou nejprve seskupovány podle typu výsledné modulace, viz.:

- dvoustavová: 1 symbol signálu $x_{rect}(t)$ odpovídá 1 bitu posloupnosti $x_b(n)$,
- čtyřstavová: 1 symbol signálu $x_{rect}(t)$ odpovídá 2 bitům posloupnosti $x_b(n)$,
- osmistavová: 1 symbol signálu $x_{rect}(t)$ odpovídá 3 bitům posloupnosti $x_b(n)$, ...

Poté jsou seskupené bity tvořící m -tice (angl. *tuples*) mapovány na patřičnou reprezentaci pomocí komplexního signálu $x_{rect}(t)$. Pro každou kombinaci bitů symbolu je definována i kombinace úrovní³ soufázni a kvadrurní složky signálu $x_{rect}(t)$. Jedním ze základních parametrů digitálních rádiových systémů je i požadovaná přenosová *bitová* (angl. *bitrate*) a z ní odvozená *symbolová* (angl. *symbol-rate*) rychlost. Symbolová rychlost definuje dobu trvání jednoho symbolu (impulzu) T_S .

Tvarování pulzu

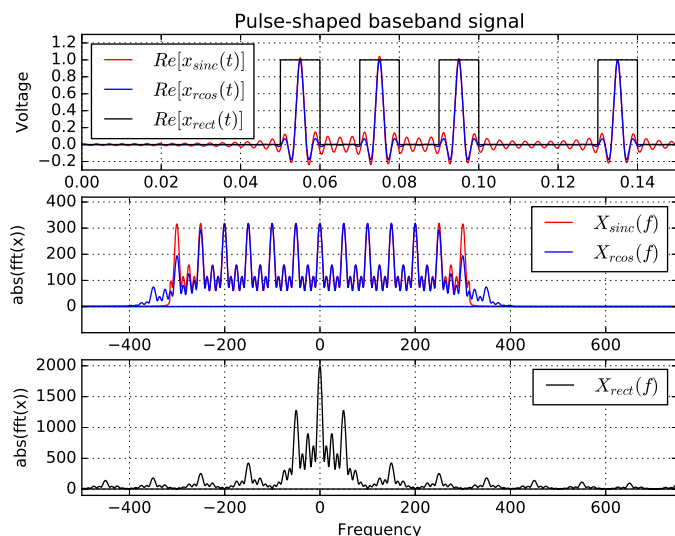
Za blokem *Modulátoru BB* jsou přechody mezi jednotlivými symboly signálu $x_{rect}(t)$ ⁴ ideálně strmé, viz obr. 2. Signál v základním pásmu (*baseband* signál) má v tuto chvíli podobu ideálních obdélníkových impulsů. Obdélníkový průběh je v reálných podmínkách nevýhodný. Skutečné přenosové kanály značně omezují kmitočtové spektrum přenášených signálů a potlačují některé kmitočty. Zjednodušeně lze jejich chování modelovat filtry typu dolní nebo pásmová propust⁵. Pro soudobé systémy pracující s vysokými bitovými

²V případě m -stavové modulace je poměr bitové rychlosti k rychlosti symbolové dán počtem bitů přenesených v jednom symbolu.

³kvantizační úroveň může být dána přímo napětím, nebo vyjádřena jen číselně. To závisí na aktuální podobě obvodu Up-konvertoru. Někdy je vyžadován vstupní signál v základním pásmu, tzv. I a Q v analogové podobě. Moderní integrované obvody provádějící konverzi však často obsahují na vstupu dva DA převodníky a signál v základním pásmu je na ně přiváděn ve formě diskretních I a Q vzorků.

⁴U komplexního signálu v základním pásmu se už nehovoří o "bitech", ale o "symbolech". Přestože v tuto chvíli pojem "symbolu" ještě není precizně definován, je kvůli správnosti výkladu použit. Nicméně, v případě dvoustavových modulací obsahuje symbol jeden datový bit. Proto je možné setkat se někdy v technické literatuře s pojmem "bit" i v případě *baseband* signálu.

⁵V angl. psané literatuře jsou označovány jako "*bandlimited channels*"



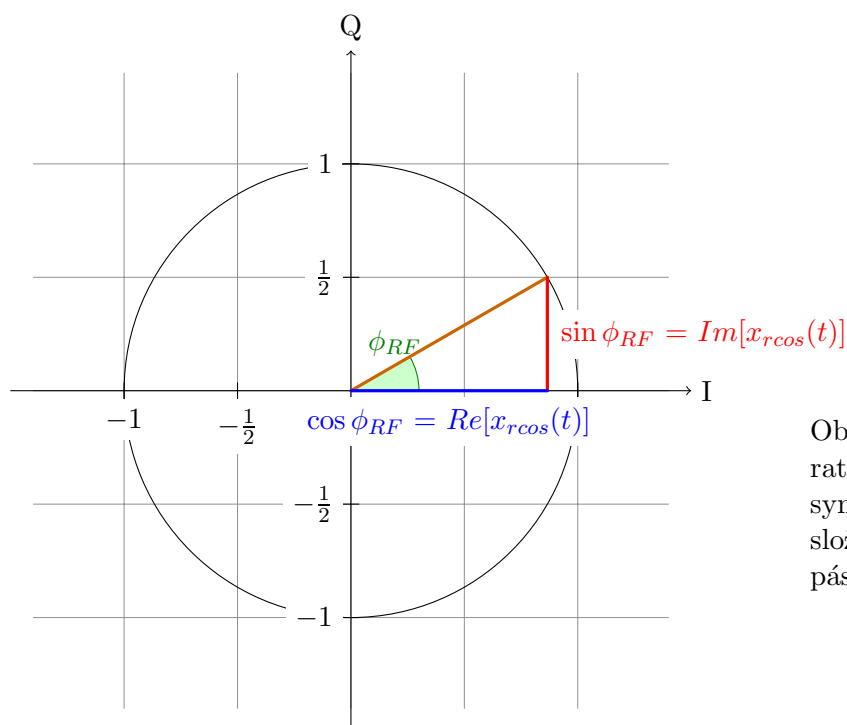
Obrázek 2: Příklad tvarování impulsu signálu v základním pásmu. Kvůli jednoduchosti je znázorněná pouze jedna (uvažujme například, že reálná - symfázní) složka komplexního signálu.

(symbolovými) rychlostmi by nepredikovatelné zkreslení impulsu jednotlivých symbolů představovalo vážný problém při detekci a následném zpracování signálu v přijímači. Proto blok *Tvarování impulsu* kontrolovaně omezí pásmo signálu $x_{rect}(t)$ tím, že tvaruje jednotlivé impulsy symbolů podle vhodných matematických funkcí. Teoreticky je jich používáno více (*gauss*, *sinc*, ...), prakticky jsou používány funkce *raised-cosine* (ve zkratce *rcos*). V podstatě se jedná o "utlumenou" funkci *sinc* (angl. *damped sinc*). Výstupem bloku "Tvarování impulsu" je tedy signál $x_{rcos}(t)$, jehož kmitočtové spektrum je na rozdíl od signálu $x_{rect}(t)$ ohraničené a definované už v době návrhu systému. Signál $x_{rect}(t)$ s ideálně obdélníkovým průběhem, signál $x_{rcos}(t)$, kde je impuls tvarován funkcí *raised-cosine* a nakonec i signál $x_{sinc}(t)$ s impulzem ve tvaru funkce *sinc* spolu se svými kmitočtovými spektry jsou prezentovány na obrázku 2.

Komplexní signál $x_{rcos}(t)$ je připraven k transformaci (ke konverzi) do kmitočtového pásma vhodného pro daný systém. Většina SDR dnes využívá obvod *kvadrurní modulátor*. Tento typ modulátoru (často pojmenovávaného anglicky *Up-Convertor*) umožňuje obrovskou flexibilitu celého rádia. Je jedním z hlavních stavebních prvků SDR a příčinou "softwarového" přístupu k tvarování signálu rádia. Jde o to, že nezměněná hardwarová struktura může vygenerovat signál $x_{RF}(t)$ s takřka libovolnou digitální modulací. Záleží jen na tom, jak bude vypadat komplexní signál v základním pásmu $x_{rcos}(t)$. Ten, pomocí své reálné složky $Re[x_{rcos}(t)]$ a imaginární složky $Im[x_{rcos}(t)]$ přesně určuje fázi $\phi_{RF}(t)$ a amplitudu výsledného signálu $x_{RF}(t)$. Obrázek 3 demonstruje tento princip. Zde uvažujeme pouze digitální modulace s konstantní amplitudou nosného signálu. U nich je manipulováno

Up-Convertor,
Transformace do
pásma VF

jen s fází $\phi_{RF}(t)$ ⁶ nosného harmonického signálu.



Obrázek 3: Princip kvadraturní modulace. Význam symfázní a kvadraturní složky signálu základního pásma.

Díky nízkým kmitočetům signálů základního pásma je možné k jejich tvarování využít procesorů (často obsahujících podpůrné struktury pro číslicové zpracování signálů) a tím pádem lze jejich strukturu i parametry určovat softwarem v procesoru. Přeneseně tedy software procesoru tvaruje (definuje) vysokofrekvenční signál na výstupu vysílače $x_{RF}(t)$.

⁶Skupina modulací manipulujících s fází (v angl. *PSK*, čili *phase shift keying*)

3 Teoretický popis funkce vysílače

Předpokládejme, že systém bude vysílat předem známou – definovanou posloupnost bitů: $b(n)$

$$b(n) = b_0, b_1, \dots, b_N \quad (1)$$

Známe jednak celkovou délku posloupnosti – N bitů a také hodnotu každého z bitů posloupnosti $b(n)$, přičemž n je z oboru přirozených čísel.

3.1 Moduační schéma

Použita bude jednoduchá digitální modulace bez paměti (*angl.* memoryless), kdy je posloupnost bitů $b(n)$ mapována na jednotlivé symboly z M -prvkové množiny (v *angl.* literatuře M -ary mapping schema). Zvolená modulace je potom pojmenována podle počtu dostupných symbolů "M-stavová". Podrobněji viz např. [1], [2] nebo jiná literatura věnující se základům digitální modulace signálů. Posloupnost $b(n)$ je tudíž rozdělena na skupinky bitů o délce k , kde

$$k = \log_2 M. \quad (2)$$

Potom se z posloupnosti bitů $b(n)$, viz rovnice (1), stává posloupnost symbolů $a(l)$

$$a(l) = a_0, a_1, \dots, a_L \quad (3)$$

Délka posloupnosti symbolů je L , kdy $L = \frac{N}{k}$.

Každý symbol $a(l)$, který je tvořen skupinkou k bitů je mapován na jeden z M možných signálů $s_m(t)$ (*angl.* waveforms). Bandpass signál

$$s_m(t), 1 \leq m \leq 2^k. \quad (4)$$

Jedná se o harmonické signály tzv. vysokofrekvenční (VF), tedy fakticky o signály v podobě, v jaké budou vysílány anténou vysílače (*angl.* bandpass). Ty jsou charakteristické (a jedinečné) svou amplitudou, fází nebo svým kmitočtem, podle typu modulace.

V této práci se budeme zabývat převážně signály, u nichž je modulovaná jejich fáze (*angl.* phase-type of modulation). Podle [2] je takový signál analyticky vyjádřen:

$$s_m(t) = h(t) \cdot e^{j\frac{2\pi(m-1)}{M}} \cdot e^{j2\pi f_c t}. \quad (5)$$

V tuto chvíli je možné $h(t)$ považovat za konstantu, která se v průběhu trvání jednoho symbolu nemění. Prostřední člen, který charakterizuje daný symbol, vyjádříme v

exponenciálním tvaru:

$$e^{j\frac{2\pi(m-1)}{M}} = \cos(2\pi\frac{m-1}{M}) + j \sin(2\pi\frac{m-1}{M}). \quad (6)$$

Baseband signál

Jak bylo naznačeno v předchozí kapitole, signály zastupující jednotlivé symboly, tedy *bandpass* signály mohou být reprezentovány jinými - nízkofrekvenčními signály tzv. základního pásma (*angl.* baseband). Ten je složen ze symfázní složky (*angl.* in-phase) $s_m^I(t)$ a kvadraturní složky (*angl.* quadrature) $s_m^Q(t)$ podle:

$$s_m^I(t) = h(t) \cdot \cos(2\pi\frac{m-1}{M}), \quad (7)$$

$$s_m^Q(t) = h(t) \cdot \sin(2\pi\frac{m-1}{M}). \quad (8)$$

Transformace signálu základního pásma na signál VF je realizována pomocí kvadraturního modulátoru (*angl.* quadrature modulation), viz např. [2].

3.2 Tvarování impulzu

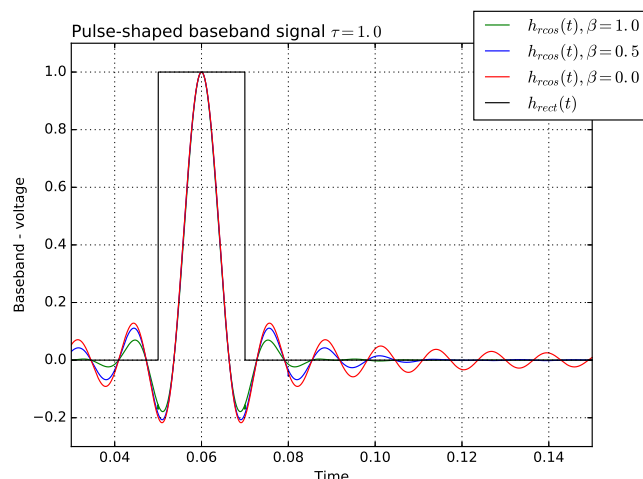
Každý ze symbolů posloupnosti $a(l)$ je vyslán po omezenou dobu. Po dobu trvání toho symbolu je vyslán příslušný signál s_m , který je v našem případě harmonický signál s příslušnou fází. V dalším textu budou tyto časové úseky nazývány v souladu s zaběhnutou terminologií impulzy.

Pokud je, jak jsme doposud uvažovali, hodnota $h(t)$ po dobu celého jednoho impulzu konstantní, je okamžitá úroveň signálu základního pásma (jeho I a Q složky) dána hodnotou m , viz rovnice (7) a (8). Pak mluvíme o obdélníkových impulzech (*angl.* rectangular).

Z toho je zřejmé, že funkce $h(t)$ definuje tvar použitého impulzu. Jak bylo již zmíněno v úvodní části, není použití obdélníkových impulzů vždy výhodné vzhledem k jejich nekonečně širokému kmitočtovému spektru. Je běžnou praxí použít jiné – výhodnější – funkce pro $h(t)$ než je konstanta, a to například funkci *raised-cosine*:

$$h(t) = \frac{\sin(\frac{\pi t}{T})}{\frac{\pi t}{T}} \cdot \frac{\cos(\pi\beta\frac{t}{T})}{1 - 4\beta^2(\frac{t}{T})^2}, \quad (9)$$

kde β , z intervalu $0 \leq \beta \leq 1$, je tzv. *roll-off* faktor. Detailněji viz [1] nebo [2]. Raised-cosine pulzy pro různé hodnoty β jsou znázorněny na obr. 4. Vězte, že pokud $\beta = 0$ potom $h(t)$ přechází na funkci $\text{sinc}(\pi t/T)$.



Obrázek 4: Raised-cosine impulzy s rozdílnou hodnotou roll-off faktoru β .

3.3 Popis časových souvislostí signálu s tvarovaným impulzem

Každý ze signálů $s_m(t)$ je vysílán po dobu trvání jednoho symbolu T_s (*angl.* symbol interval). Odtud také symbolová rychlost R_s (*angl.* symbol rate):

$$R_s = \frac{1}{T_s} \quad (10)$$

Potom T_s definuje délku trvání impulzu a každý ze signálů $s_m(t)$ je vysílán uvnitř příslušného symbolového intervalu T_s s patřičnou hodnotou parametru m . Tvarovací funkce $h(t)$ z rovnice. (9) se tudíž mění na:

$$h(t - lT_s) = \frac{\sin\left(\frac{\pi(t-lT_s)}{T_s}\right)}{\frac{\pi(t-lT_s)}{T_s}} \cdot \frac{\cos\left(\pi\beta\frac{(t-lT_s)}{T_s}\right)}{1 - 4\beta^2\left(\frac{(t-lT_s)}{T_s}\right)^2}, \quad (11)$$

kde l značí pořadové číslo aktuálního symbolu v rámci posloupnosti $a(l)$.

Impuls tohoto tvaru potom splňuje Nyquistovo kritérium, viz [2], a nedochází k mezi-symbolové interferenci (*angl.* inter-symbol interference) a k nejednoznačnosti při detekci a identifikaci symbolu v přijímači.

Signál základního pásma pro příslušný l -tý symbol posloupnosti $a(l)$, potom z rovnice (3) je dán:

$$s_{m,l}^I(t) = h(t - lT_s) \cdot \cos\left(2\pi\frac{m_l - 1}{M}\right), \quad (12)$$

$$s_{m,l}^Q(t) = h(t - lT_s) \cdot \sin\left(2\pi\frac{m_l - 1}{M}\right). \quad (13)$$

4 Generátor rádiového signálu

Z úvodní části tohoto textu a z obrázku 1 je patrné několik stupňů – fází, ze kterých se celý proces generování rádiového signálu skládá. Uvažujme, že vstupem je posloupnost, viz rovnice (1). V první řadě jde o formování požadovaných n -tic bitů přenášené bitové posloupnosti $b(n)$, podle zvoleného počtu stavů modulace M . Každá z n -tic bitů definuje hodnotu patřičného symbolu. Hodnotu symbolu je potřeba "namapovat" na patřičnou podobu složek signálu základního pásma a poté složky I a Q, tedy $s_m^I(t)$ a $s_m^Q(t)$, podle rovnic (7) a (8) vygenerovat.

up-convertor V digitálním rádiovém systému slouží blok "Up-Convertor" k transformaci připraveného signálu v základním pásmu (baseband) do kmitočtového pásma vhodného k rádiovému přenosu. řekněme, že daný krok lze nazvat i "modulací nosné".

V tomto projektu bude vytvořen Up-convertor pro účely simulace. Ve vlastní hardwarové implementaci slouží k přeměně samotné rádio, čili fyzicky RF-frontend.

4.1 Realizace Up-convertoru - Python

Rozbor Let's start with a time-domain first to derive the differential equation from a *free-body diagram*. We will use the second Newton's law, setting the mass $M = 0$.

4.2 Způsob generování

4.2.1 Přímý generátor

4.2.2 Root raised – cosine filtr

4.3 Použití bloku USRP v GNU Radio

We are required to derive both natural and step response.

Listing 1: Mapování bitové posloupnosti

```

1 |
2 | """
3 |
4 |     :return: Baseband signal
5 | """
6 |     tup = 2 # number of bits in one tuple
7 |     n_d = tup - (np.size(data) % tup)
8 |     data = np.append(data, np.zeros(n_d)) # appends zero to get odd number
9 |     data_2s = data.reshape(-1, tup) # reshaping the data vector into matrix
10 |     data_2s = data_2s > 0

```

5 Simulace - popis vytvořených funkcí

Veškeré simulace byly vytvořeny v jazyce Python, variantě 3.5.2. s využitím nástrojů knihoven Matplotlib 1.5.3, NumPy 1.11.1 a SciPy 0.18.1, vše instalováno v rámci balíku Anaconda 4.2.0. Simulace byly napsány, vyzkoušeny a provozovány na počítači s operačním systémem Linux Ubuntu 16.04.

Vytvořené funkce simulací jsou sdruženy do několika samostatných souborů s koncovkou `.py` a umístěné v podadresářích uvnitř samostatného adresáře `numpy`. Členění do podadresářů je podle logické a funkční příslušnosti. Dále hlavní adresář `numpy` obsahuje řadu příkladů použití simulačních funkcí.

Funkce ze souborů ve vnořených adresářích lze využít až poté, co jsou do skriptu hlavní úrovně importovány⁷.

5.1 Generátory signálu

Funkce zde popsané jsou umístěny v podadresáři `siggens`. Patří k základním prostředkům modulací – generují signál požadovaného tvaru a parametrů a to jak v podobě základní (jediný, neopakující se impuls), tak i ve variantě opakujících se impulsů obsahujících kód.

5.1.1 Jednoduché impulsy

Funkce, které byly vytvořeny proto, aby generovaly jediný a neopakující se impuls jsou sdruženy v souboru `one_pulse.py`. Mohou sloužit jednak k prezentaci impulsu daného tvaru a jeho vlastností, zároveň jsou ale základem pro funkce, které spojují jednotlivé impulsy do jejich posloupností.

Funkce `rect_p` - obdélníkový impuls Funkce, viz. listing 2, generuje jednoduchý obdélníkový impuls. Vstupními parametry jsou t_{START} a t_{END} , tedy okamžik začátku a konce impulsu v rámci časové osy t . Výstupem funkce je vektor p vzorků impulsu odpovídajících vzorkovacím okamžikům definovaných vstupním vektorem t .

Listing 2: Generátor jednorázového obdélníkového impulsu

```
1 def rect_p( t, t_start, t_end ):
2     p = ((t > t_start) & (t < t_end)) * 1
3     return (p)
```

⁷Toto je pochopitelně dokumentovanou vlastností jazyka Python. Zde je tento fakt zmíněn pouze jako připomínka.

Funkce sinc_p - sinc(x) impuls Funkce, viz. listing 3, generuje jednoduchý sinc impuls. Vstupními parametry jsou t_0 a T_B , tedy střed impulsu v rámci časové osy t a šířka hlavního laloku T_B . Výstupem funkce je vektor p vzorků impulsu odpovídajících vzorkovacím okmžikům definovaných vstupním vektorem t .

Listing 3: Generátor jednorázového impulsu tvaru $\frac{\sin(x)}{x}$

```
1 def sinc_p( t,t0,Tb ):
2     p = np.sinc(np.pi*(t - t0)/Tb)
3     return (p)
```

Funkce rcos_p - Raised-cosine impuls Funkce, viz. listing 4, generuje jednoduchý raised-cosine impuls. Vstupními parametry jsou t_0 a T_B , tedy střed impulsu v rámci časové osy t a šířka hlavního laloku T_B . Výstupem funkce je vektor p vzorků impulsu odpovídajících vzorkovacím okmžikům definovaných vstupním vektorem t .

Listing 4: Generátor jednorázového impulsu tvaru raised-cosine

```
1 def rcos_p( t,t0,Tb,alpha ):
2     beta = alpha / Tb
3     damp = np.cos(np.pi*beta*(t - t0)) / (1 - 4*(beta*(t - t0))**2)
4     p = np.sinc(np.pi*(t - t0)/Tb) * damp
5     return (p)
```

5.1.2 Posloupnosti impulsů

Funkce, které byly vytvořeny proto, aby generovaly posloupnost impulsů různého tvaru jsou sdruženy v souboru `train_pulse.py`. Jsou napsány tak, aby jednak generovaly impuls požadovaného tvaru několikrát opakovaně v průběhu časové osy dané vektorem t . Dále mohou zahrnout i kódování - impuls je nebo není na daném časovém úseku přítomen, podle toho, je-li příslušný bit kódové posloupnosti roven logické jedničce nebo nule.

Funkce rect_tr - obdélníkový impuls Funkce, viz. listing 5, generuje posloupnost obdélníkových impulsů. Vstupními parametry jsou délka impulsu t_p , délka mezery mezi impulsy t_s , a zpoždění celé posloupnosti – mezera mezi začátkem časové osy t a začátkem prvního impulsu. Dále je vstupem vektor logických hodnot `code`. Ten určuje jednak kódovou posloupnost – tzn. přítomnost nebo nepřítomnost daného impulsu v rámci celé řady impulsů. Dále počet bitů posloupnosti `code` určuje počet impulsů výsledného signálu.

Výstupem funkce je vektor x vzorků impulsů odpovídajících vzorkovacím okmžikům definovaných vstupním vektorem t .

Listing 5: Generátor posloupnosti obdélníkových impulsů

```
1 def rect_tr( t,tp,ts,td,code ):

```

```

2 |     n = np.size(code)
3 |     x=np.zeros(np.shape(t)) > 1
4 |     for i1 in range (0,n):
5 |         p = pulse.rect_p( t,td + i1*(tp+ts),td + tp + i1*(tp+ts) )
6 |         x = (x | p & code[i1]) * 1
7 |     return (x)

```

Funkce sinc_tr - sinc(x) impulz Funkce, viz. listing 6, generuje posloupnost impulzů $\text{sinc}(x)$. Vstupními parametry jsou délka impulzu t_p , délka mezery mezi impulzy t_s , a zpoždění celé posloupnosti – mezera mezi začátkem časové osy t a středem prvního impulzu. Dále je vstupem vektor logických hodnot $code$. Ten určuje jednak kódovou posloupnost – tzn. přítomnost nebo nepřítomnost daného impulzu v rámci celé řady impulzů. Dále počet bitů posloupnosti $code$ určuje počet impulzů výsledného signálu.

Parametr pw je specifický pro tvar impulzu sinc . Určuje délku hlavního laloku funkce $\text{sinc}(x)$ uvnitř celého pulzu t_p . Jestliže parametr t_p můžeme nastavit podle bitové rychlosti $bitrate$, potom samotný $\text{sinc}(x)$ impulz může být v rámci jednoho bitu kratší - užší.

Výstupem funkce je vektor x vzorků impulzů odpovídajících vzorkovacím okmžikům definovaných vstupním vektorem t .

Listing 6: Generátor posloupnosti impulzů $\frac{\sin(x)}{x}$

```

1 | def sinc_tr( t,ts,td,code,pw ):
2 |
3 |     #pw = 2e-3 # pulse width in ms
4 |     n = np.size(code)
5 |     x=np.zeros(np.shape(t))
6 |     for i1 in range (0,n):
7 |         p = pulse.sinc_p( t,td + i1*(ts),pw )
8 |         x = (x + p * code[i1])
9 |     return (x)

```

Funkce rcos_tr - Raised-cosine impulz Funkce, viz. listing 7, generuje posloupnost impulzů raised-cosine. Vstupními parametry jsou délka hlavního laloku impulzu t_p , délka mezery mezi impulzy t_s , a zpoždění celé posloupnosti – mezera mezi začátkem časové osy t a středem prvního impulzu. Dále je vstupem vektor logických hodnot $code$. Ten určuje jednak kódovou posloupnost – tzn. přítomnost nebo nepřítomnost daného impulzu v rámci celé řady impulzů. Dále počet bitů posloupnosti $code$ určuje počet impulzů výsledného signálu.

Parametr pw je jako v případě sinc impulzu specifický pro tvar impulzu raised-cosine. Určuje délku hlavního laloku funkce $\text{sinc}(x)$ uvnitř celého pulzu t_p . Jestliže parametr t_p můžeme nastavit podle bitové rychlosti $bitrate$, potom samotný $\text{sinc}(x)$ impulz může být v rámci jednoho bitu kratší - užší.

Parametr α udává tzv. roll-off faktor funkce raised-cosine.

Výstupem funkce je vektor x vzorků impulzů odpovídajících vzorkovacím okmžíkům definovaných vstupním vektorem t .

Listing 7: Generátor posloupnosti raised-cosine impulzů

```

1 def rcos_tr( t,ts,td,code,pw,alpha ):
2
3     #pw = 2e-3 # pulse width in ms
4     #alpha = .5 # roll-off factor
5     n = np.size(code)
6     x=np.zeros(np.shape(t))
7     for i1 in range (0,n):
8         p = pulse.rcos_p( t,td + i1*(ts),pw,alpha )
9         x = (x + p * code[i1])
10    return (x)

```

Důležité: Parametr t_d znamená u posloupnosti obdélníkových impulzů zpoždění posloupnosti vzhledem k náběžné hraně prvního pulzu, u posloupností impulzů tvaru $\text{sinc}(x)$ a raised-cosine vzhledem je ke středu hlavního laloku. To je potřeba respektovat, pokud se posloupnosti umísťují na časové ose společně.

5.1.3 Generátory pseudonáhodných posloupností

Funkce, které byly vytvořeny proto, aby generovaly posloupnosti bitů — generátory kódových a pseudonáhodných posloupností — jsou sdruženy v souboru `PRN_bitstreams.py`. Tyto funkce jsou vhodné jednak k testování digitálních rádiových systémů a potom k rozptřeni kmitočtového spektra v systémech typu DSSS (z angl. *Direct Sequence Spread Spectrum*).

tri_seq: Tří-bitový SSRG. Funkce, viz. listing 8, generuje bitovou posloupnost pomocí posuvného registru se zavedenými několika zpětnými vazbami SSRG⁸. V tomto konkrétním případě je použit generátor ve Fibonacciho tvaru s tří-bitovým registrem. Prvním vstupem funkce je parametr `init_reg` logický vektor (3 bity), který určuje počáteční stav registru před startem procesu generování posloupnosti. Druhým vstupním parametrem je opět tříbitový logický vektor `fb_reg`, který definuje zpětné vazby posuvného registru. Logická "0" v tomto vektoru znamená, že zpětná vazba není z daného bitu posuvného registru odvadana. Naopak logická "1" indikuje přítomnost zpětné vazby z příslušného bitu registru.

Výstupem funkce je 10-ti bitový vektor logických hodnot - vygenerovaná bitová posloupnost.

⁸z angl. *Simple Shift Register Generator*

Listing 8: Generátor jednoduché posloupnosti. 3-bitový SSRG, generována je posloupnost 10-ti bitů.

```

1  # 10-bits sequence outputting 3-bits SSRG in a Fibonacci form.
2  def tri_seq(init_reg,fb_reg):
3      # Output register
4      x=np.zeros([10])
5      # Shift register
6      shft_reg = init_reg
7      # Feedback registers - bit '1' means -> FB is connected
8      # defined as an input argument fb_reg
9      for i1 in range (0,9):
10         in1 = int(np.dot(shft_reg,fb_reg)%2)
11         x[i1] = shft_reg[2]
12         shft_reg = np.roll(shft_reg,1)
13         shft_reg[0] = in1
14
15     return (x.astype(bool))

```

gold_seq: Generátor Goldova kódu. Funkce, viz. listing 9, generuje Goldovu pseudonáhodnou bitovou posloupnost, tzv. Goldův kód. Jsou použity dva 10-bitové posuvné registry se zavedenými vlastními několika zpětnými vazbami a se dvěma vazbami *vzájemnými*. Vzájemné vazby zabezpečují navázání registrů a spojení dvou nezávislých posloupností do jedné výsledné. Rozmístění vlastních zpětných vazeb jednotlivých registrů, stejně jako jejich počáteční stavy, jsou v souladu se standardem družicového navigačního systému GPS tak, aby generované posloupnosti odpovídaly posloupnostem použitým v tomto systému.

Prvními dvěma vstupními parametry jsou umístění *vzájemných* vazeb **x1** a **x2**. Jedná se o celá čísla v rozsahu 0–9, která definují číslo bitu posuvného registru, z něž je vazba odvedena⁹. Třetím vstupním parametrem je opět celé číslo **N_codes**, které určuje počet period kódu vygenerovaných na výstupu¹⁰.

Výstupem funkce je vektor logických hodnot o délce ($N_{codes} \cdot 1023$) – vygenerovaných **N_codes** period Goldovy posloupnosti.

Listing 9: Generátor Goldovy posloupnosti.

```

1  # Gold sequence generator - two 10-bits SSRG
2  def gold_seq(x1,x2,N_codes):
3      # Output register
4      x=np.zeros([N_codes*1023])
5      # Maximum-length sequence generators:
6      # Shift registers
7      shft_reg_1 = np.array ([1,1,1,1,1,1,1,1,1,1])
8      shft_reg_2 = np.array ([1,1,1,1,1,1,1,1,1,1])

```

⁹Jednotlivé kombinace **x1** a **x2** produkují unikátní posloupnost. Standard popisující systém GPS precizně definuje konkrétní kombinace pro kódy petřící jednotlivým družicím a jiným typům vysílačů v systému (např. družice SBAS, pseudolity a jiné)

¹⁰Jedna perioda Goldovy posloupnosti obsahuje 1023 bitů.

```

9      # Feedback registers - bit '1' means -> FB is connected
10     fbck_reg_1 = np.array ([0,0,1,0,0,0,0,0,1])
11     fbck_reg_2 = np.array ([0,1,1,0,0,1,0,1,1])
12
13     for i1 in range (0,N_codes*1023):
14         g1 = shft_reg_1 [9]
15         g2 = ( shft_reg_1[x1] + shft_reg_2[x2] ) % 2
16         in1 = np.dot(shft_reg_1,fbck_reg_1) % 2
17         in2 = np.dot(shft_reg_2,fbck_reg_2) % 2
18         x[i1] = ( g1 + g2 ) % 2
19         shft_reg_1 = np.roll(shft_reg_1,1)
20         shft_reg_1[0] = in1
21         shft_reg_2 = np.roll(shft_reg_2,1)
22         shft_reg_2[0] = in2
23
24     return (x.astype(bool))

```

5.2 Modulátory

V této části jsou popisovány funkce uskutečňující modulace signálu — tvarování komplexního signálu v základním pásmu a jeho transformaci do vyšších kmitočtů - konverzi do tzv., *passbandu*. Funkce jsou umístěny v podadresáři *modulators*.

5.2.1 Tvarovače signálu v základním pásmu

V základním pásmu — *basebandu* — je potřeba vytvořit komplexní signál, složený ze symfázní (I) a kvadraturní (Q) složky. Aktuální úrovně těchto dvou složek definují okamžitou fázi a amplitudu signálu v *passbandu* — tedy na výstupu *Up-Convertoru*, potažmo celého vysílače.

Zde jsou popisovány funkce, vytvořené k tvarování signálu základního pásma. Jsou sdruženy v souboru *constellation_mappers.py*. Na základě vstupní bitové posloupnosti – *dat* – je tvarován výstupní komplexní signál $I + jQ$ jednak podle požadované výsledné digitální modulace a také podle požadovaného tvaru impulsu.

rect_bpsk_map: BPSK modulace s obdélníkovým impulzem. Funkce tvaruje komplexní signál základního pásma pro modulaci BPSK s obdélníkovým impulzem. Vstupem je časová osa *t*, bitová posloupnost (vektor logických hodnot) *data*, která bude namodulována, požadovaná bitová rychlost *b_rate* a volitelně parametry výstupního signálu:

- *tp* je hodnota t_p , šířka pulzu (bitu). Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_p = \frac{1}{b_rate}$.
- *td* je hodnota t_d , zpoždění posloupnosti od počátku časové osy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_d = 0$.

- **ts** je hodnota t_s , mezera mezi impulzy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_s = 0$.

V případě BPSK je kvadraturní signál konstantní, roven nule; symfázní signál nabývá hodnoty 1 při aktuální hodnotě **data** rovné logické 1 a hodnoty -1, pokud je daný bit vektoru **data** roven logické 0.

rect_qpsk_map: QPSK modulace s obdélníkovým impulzem. Funkce tvaruje komplexní signál základního pásma pro modulaci QPSK s obdélníkovým impulzem. Vstupem je časová osa **t**, bitová posloupnost (vektor logických hodnot) **data**, která bude namodulována, požadovaná symbolová rychlost **s_rate** a volitelně parametry výstupního signálu:

- **tp** je hodnota t_p , šířka pulzu (symbolu). Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_p = \frac{1}{s_rate}$.
- **td** je hodnota t_d , zpoždění posloupnosti od počátku časové osy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_d = 0$.
- **ts** je hodnota t_s , mezera mezi impulzy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_s = 0$.

Modulace QPSK přenáší v jednom symbolu dva datové bity. Proto je datová posloupnost nejprve rozdělena na dvojice (data jsou přeskupeny do matice **data_2s** o dvou sloupcích). Pokud počet bitů v původním vstupním vektoru **data** není sudý, je doplněna nula. *Baseband* signál modulace QPSK je následně tvoren tak, že symfázní signál nabývá hodnoty 1, resp. -1 pro aktuální hodnoty logické 1, resp. 0 prvního bitu aktuální dvojice vstupních bitů (rozuměj aktuálního řádku matice **data_2s**). Obdobně je tvarován i kvadraturní signál, ten ovšem vychází z druhého bitu daného řádku matice **data_2s**.

sinc_bpsk_map: BPSK modulace s impulzem tvaru $\text{sinc}(x)$. Funkce tvaruje komplexní signál základního pásma pro modulaci BPSK s obdélníkovým impulzem. Vstupem je časová osa **t**, bitová posloupnost (vektor logických hodnot) **data**, která bude namodulována, požadovaná bitová rychlost **b_rate** a volitelně parametry výstupního signálu:

- **tp** je hodnota t_p , šířka pulzu (bitu). Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_p = \frac{1}{b_rate}$.
- **td** je hodnota t_d , zpoždění posloupnosti od počátku časové osy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_d = 0$.

- **ts** je hodnota t_s , mezera mezi impulzy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_s = 0$.

V případě BPSK je kvadraturní signál konstantní, roven nule; symfázní signál nabývá hodnoty 1 při aktuální hodnotě **data** rovné logické 1 a hodnoty -1, pokud je daný bit vektoru **data** roven logické 0.

sinc_qpsk_map: QPSK modulace s obdélníkovým impulzem. Funkce tvaruje komplexní signál základního pásma pro modulaci QPSK s obdélníkovým impulzem. Vstupem je časová osa **t**, bitová posloupnost (vektor logických hodnot) **data**, která bude namodulována, požadovaná symbolová rychlost **s_rate** a volitelně parametry výstupního signálu:

- **tp** je hodnota t_p , šířka pulzu (symbolu). Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_p = \frac{1}{s_rate}$.
- **td** je hodnota t_d , zpoždění posloupnosti od počátku časové osy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_d = 0$.
- **ts** je hodnota t_s , mezera mezi impulzy. Pokud není tento parametr explicitně definován při volání funkce, je přednastaveno $t_s = 0$.

Modulace QPSK přenáší v jednom symbolu dva datové bity. Proto je datová posloupnost nejprve rozdělena na dvojice (data jsou přeskupeny do matice **data_2s** o dvou sloupcích). Pokud počet bitů v původním vstupním vektoru **data** není sudý, je doplněna nula. *Baseband* signál modulace QPSK je následně tvrovan tak, že symfázní signál nabývá hodnoty 1, resp. -1 pro aktuální hodnoty logické 1, resp. 0 prvního bitu aktuální dvojice vstupních bitů (rozuměj aktuálního řádku matice **data_2s**). Obdobně je tvarován i kvadraturní signál, ten ovšem vychází z druhého bitu daného řádku matice **data_2s**.

5.2.2 Up – Converter

5.3 Pomocné funkce

5.4 Filtry

5.4.1 Tvarovače impulzu

6 Implementace - popis vytvořených programů

Implementace algoritmů do SDR byly vytvořeny v jazyce Python, variantě 2.7. s využitím nástrojů balíku GNU Radio verze 3.7.10.1, dále knihoven NumPy 1.11.1 a SciPy 0.18.1. Simulace byly napsány, vyzkoušeny a provozovány na počítači s operačním systémem Linux Ubuntu 16.04.

Dále pro přímou kompilaci algoritmů napsaných v jazyce C++ a využívajících API funkcí dodaných s ovladačem UHD výrobce rádia Ettus E310 bylo použito kompilátoru GNU C++ verze 4.9.2. Verze ovladače UHD 003.009.002.

Vytvořené příklady implementací jsou umístěné v podadresářích uvnitř samostatného adresáře `srcpy`

`hwimpl`, resp. `srcc`. Členění do podadresářů je podle logické a funkční příslušnosti.

Funkce ze souborů ve vnořených adresářích lze využít až poté, co jsou do skriptu hlavní úrovně importovány¹¹.

6.1 Implementace vysílače pomocí GNURadio v jazyce Python

Funkce zde popsané jsou umístěny v podadresáři `siggens`. Patří k základním prostředkům modulací – generují signál požadovaného tvaru a parametrů a to jak v podobě základní (jediný, neopakující se impuls), tak i ve variantě opakujících se impulsů obsahujících kód.

6.2 Implementace vysílače pomocí API funkcí UHD v jazyce C++

Programy, které byly vytvořeny proto, aby demonstrovaly možnosti použití API funkcí dodaných výrobcem rádia spolu s ovladači UHD jsou uloženy v adresáři `srcc`.

Jsou napsány jako zdrojové soubory jazyka c++ v souborech s koncovkou `.cpp`. Definice použitých tříd jsou potom přidány v podobě hlavičkových souborů s koncovkou `.hpp`.

Pro to, aby bylo možno programy spustit na cílové platformě, je potřeba je tam i zkompileovat. Spolu s API funkcemi výrobce dodává i vlastní sadu příkladů použití a dále vzorový příklad kompilace. Ten byl po nezbytných úpravách uložen v adresáři `src UHD_compile`. Postup je popsán v příloženém souboru `README`.

¹¹Toto je pochopitelně dokumentovanou vlastností jazyka Python. Zde je tento fakt zmíněn pouze jako připomínka.

Reference

- [1] B. Lathi and Z. Ding, *Modern Digital and Analog Communication Systems*, ser. Oxford series in electrical and computer engineering. Oxford University Press, 2009.
[Online]. Available: <https://books.google.cz/books?id=dltNPwAACAAJ>
- [2] Proakis, *Digital Communications 5th Edition*. McGraw Hill, 2007.