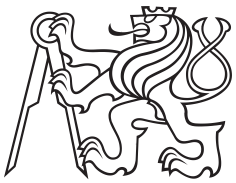
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Radioelectronics

Experimental tranciever platform for verification of coding and processing in radio communications

Ladislav Kršek

Supervisor: Sýkora Jan Prof. Ing. CSc.
Field of study: Open Electronic Systems
Prague 2017

Acknowledgements

I would like to thank my supervisor Sýkora Jan Prof. Ing. CSc. for allowing me to work on this project.

Declaration

I declare that I completed the presented thesis independently and that all used sources are quoted in accordance with methodical instructions that cover the ethical principles for writing academic thesis.

In Prague, 26.5. 2017

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských prací.

V Praze, 26.5. 2017

Abstract

The main purpose of this thesis is to establish transceiver platform with USRP™ (Universal Software Radio Peripheral) N210 from Ettus Research company. Thanks to this platform users should be able to test their Matlab® or Simulink® signal processing codes in real channel. The theory part will be focused on basic signal processing theory. Such as complex envelope, orthogonal expansion and linear memoryless modulations.
In the practical part will be described algorithms behind the platform, its restrictions and requests for users to properly utilize its functions. The programming languages used are *PHP* for creating websites, *Bash Shell* for files control, Matlab® opening and establishing first communication with USRP™. And finally there will be described Matlab® and Simulink® codes for sending and receiving data from USRP™ modules.

Keywords: USRP, Ettus research, Matlab, Simulink, bash, php, signal processing

Supervisor: Sýkora Jan Prof. Ing. CSc.

Abstrakt

Cílem této práce je sestavit platformu pro ověření základních algoritmu kódování a zpracovaní signálu pomocí softwarových rádií USRP™(Universal Software Radio Peripheral) od společnosti Ettus Research. Algoritmy si uživatelé budou moci ozkoušet po nahrání jejich kůdu v Matlabu® nebo Simulinku® . V teoretické časti se zaměřím na základní poznatky použité v teorie zpracování signálů jako je komplexní obálka, ortogonální rozklad či lineární modulace bez paměti.
V praktické časti se pak zaměřím na algoritmy stojící za touto platformou, její omezení a požadavky na uživatele, aby ji správně použili. Použité programovací jazyky jsou PHP, pro sestavení webových stránek, Bash Shell pro ovládaní souborů, první komunikace s USRP™ moduly a automatické spouštení Matlabu® či Simulinku®. Nakonec budou popsány kódy Matlabu® pro přeposílaní dat mezi hostujícím počítačem a USRP™ moduly.

Klíčová slova: USRP, Ettus research, Matlab, Simulink, bash, php, zpracování signálu

Contents

Figures

Tables

Part I

Theory

Chapter 1

Signal Analysis

Because users will use this platform as the test for their signal processing algorithms, I would like to mention some basic theories about signal processing.

1.1 Introduction

[Pro95] We will assume that the Fourier transform ($\mathscr{F}$) of signal $x(t)$ is $X(f)$. If signal $x(t)$ is real, than it has Hermitian symmetry, i.e.:

$$X(-f) = X^*(f) \tag{1.1}$$
$$|X(-f)| = |X^*(f)| \tag{1.2}$$
$$-\arg\left(X(-f)\right) = \arg\left(X^*(f)\right) \tag{1.3}$$

This means that all the information about the signal is stored positive or negative frequencies. Signal $x(t)$ can be reconstructed from $X(f)$ for $f > 0$. We can define two types of real-valued signals.
We define *positive* $X_+(f)$ and *negative* $X_-(f)$ spectrum of s signal $x(t)$ as:

$$X_+(f) = \begin{cases} X(f), & \text{for } f > 0 \\ \frac{1}{2}X(0), & \text{for } f = 0 \\ 0, & \text{for } f < 0 \end{cases} \qquad X_-(f) = \begin{cases} X(f), & \text{for } f < 0 \\ \frac{1}{2}X(0), & \text{for } f = 0 \\ 0, & \text{for } f > 0 \end{cases} \tag{1.4}$$

$$X(f) = X_+(f) + X_-(f) = X_+(f) + X_+^*(-f) \tag{1.5}$$

According to equation 1.5 $X_+(f)$ is sufficient to reconstruct X(f) *Lowpass signal* is a signal whose spectrum is located around zero frequency. We can define interval [-W, W]. Outside this interval $X(f) = 0$. An example Fig. 1.1.

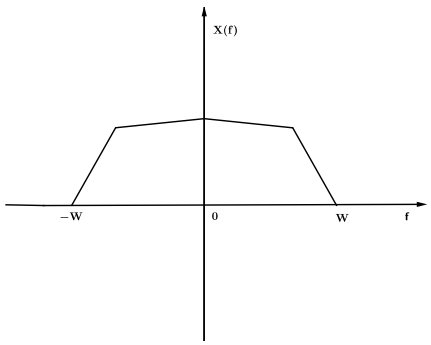


Figure 1.1: Lowpass signal

Bandpass signal is a real signal whose spectrum is located around $\pm f_0$ which is far from zero. An example Fig. 1.2.

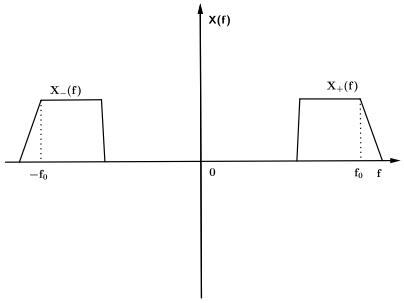


Figure 1.2: Bandpass signal

1.2 Complex envelope

We define unit step $\underline{1}(t)$ as:

$$\underline{1}(t) = \begin{cases} 1, & \text{for } t > 0 \\ \frac{1}{2}, & \text{for } t = 0 \\ 0, & \text{for } t < 0 \end{cases} \tag{1.6}$$

$$\mathscr{F}[\underline{1}(t)] = \frac{1}{2}\delta(t) - j\frac{1}{2\pi f} \tag{1.7}$$

$$\mathscr{F}^{-1}[\underline{1}(f)] = \frac{1}{2}\delta(t) + j\frac{1}{2\pi t} \tag{1.8}$$

[Pro95] We define pre-envelope signal, corresponding to $x(t)$ as the signal $x_+(t)$. This signal has only positive frequency components and his spectrum is not Hermitian, which means that signal $x_+(t)$ is complex in general. We can write:

$$\begin{aligned} x_+(t) &= \mathscr{F}^{-1}[X_+(f)] \\ &= \mathscr{F}^{-1}[X(f)\underline{1}(t)] \\ &= x(t) \star \left(\frac{1}{2}\delta(t) + j\frac{1}{2\pi t}\right) \\ &= \frac{1}{2}x(t) + j\frac{1}{2}\hat{x}(t) \end{aligned} \tag{1.9}$$

Where $\hat{x}(t) = x(t) \star \frac{1}{\pi t}$ is Hilbert transform ($\mathscr{H}$) of $x(t)$. Hilbert transform shifts signal by $-\frac{\pi}{2}$ on positive frequencies and $\frac{\pi}{2}$ on negative frequencies. In frequency domain we get:

$$\mathscr{F}[\hat{x}(t)] = -j\mathrm{sgn}(f)X(f) \tag{1.10}$$

Inverse Hilbert transform $x(t) = \mathscr{H}^{-1}[\hat{x}(t)]$

$$\mathscr{H}^{-1}[\hat{x}(t)] = -\mathscr{H}[\hat{x}(t)] \tag{1.11}$$

Proof

$$\begin{aligned} -j\mathrm{sgn}(f)(-j\mathrm{sgn}(f)X(f)) &= -X(f) \\ \mathscr{H}\left[\mathscr{H}[x(t)]\right] &= -x(t) \end{aligned} \tag{1.12}$$

Now we can define *complex envelope* ($x_e(t)$) of $x(t)$ with signal spectrum defined by:

$$X_e(f) = 2X_+(f + f_0) \tag{1.13}$$

Spectrum of this signal is located around zero frequency and has only positive component. Fig. 1.3.

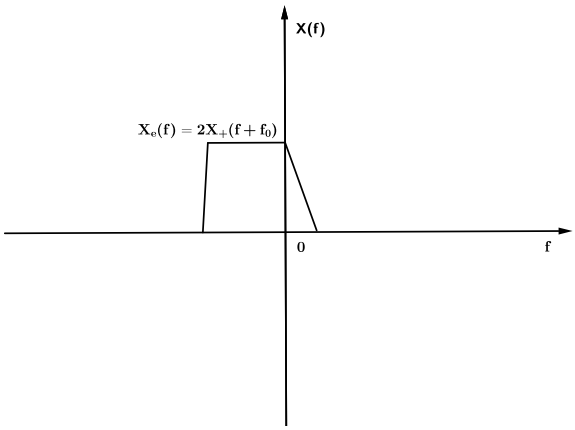


Figure 1.3: Complex envelope of $x(t)$

We can write:

$$\begin{aligned} x_e(t) &= \mathscr{F}^{-1}[X_e(f)] \\ &= 2x_+(t)\mathrm{e}^{-j2\pi f_0 t} \\ &= (x(t) + j\hat{x}(t))\mathrm{e}^{-j2\pi f_0 t} \\ &= \left[x(t)\cos(2\pi f_0 t) + \hat{x}(t)\sin(2\pi f_0 t)\right] \\ &\quad + j\left[\hat{x}(t)\cos(2\pi f_0 t) - x(t)\sin(2\pi f_0 t)\right] \end{aligned} \tag{1.14}$$

We can see, that we can obtain $x(t)$ from $x_e(t)$ by:

$$x(t) = \mathscr{R}\left[x_e(t)\mathrm{e}^{2\pi f_0 t}\right] \tag{1.15}$$

Which is inverse complex envelope transform
We can obtain $X(f)$ from equation:

$$X(f) = \frac{1}{2}\Big(X_e(f - f_0) + X_e^*(-f - f_0)\Big) \tag{1.16}$$

The real part of $x_e(t)$ is called *in-phase component* $(x_i(t))$ and imaginary part $x_e(t)$ is called *quadrature component*$(x_q(t))$
From equation (1.14) we can see that:

$$x_i(t) = x(t)\cos(2\pi f_0 t) + \hat{x}(t)\sin(2\pi f_0 t) \tag{1.17}$$

$$x_q(t) = \hat{x}(t)\cos(2\pi f_0 t) - x(t)\sin(2\pi f_0 t) \tag{1.18}$$

In polar coordinates the previous equations are rewritable as:

$$r_x(t) = \sqrt{x_i^2(t) + x_q^2(t)} \tag{1.19}$$

$$\theta_x(t) = \arctan\frac{x_q(t)}{x_i(t)} \tag{1.20}$$

$$x_e(t) = r_x(t)\mathrm{e}^{j\theta_x(t)} \tag{1.21}$$

Substituting 1.21 into 1.15 we get:

$$x(t) = \mathscr{R}\left[e^{j\left(2\pi f_0 t + \theta_x(t)\right)}\right] \tag{1.22}$$

$$= r_x(t)\cos(2\pi f_0 t + \theta_x(t)) \tag{1.23}$$

1.3 Random processes

Definition 1.1. *Mean, Mean square, Variance, autocorrelation, cross-correlation functions*

$$\mu_x(t) = E[x(t)] \tag{1.24}$$

$$\mu s_x = E[|x(t)|^2] \tag{1.25}$$

$$\sigma_x^2 = E[|x(t) - \mu_x(t)|^2] \tag{1.26}$$

$$R_x(t_1,\ t_2) = E[x(t_1)x^*(t_2)] \tag{1.27}$$

$$R_{xy}(t_1,\ t_2) = E[x(t_1)y(t_2)] \tag{1.28}$$

$$Av[x(t)] = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)dt \tag{1.29}$$

$$\tag{1.30}$$

Definition 1.2. *Stationarity* Signal is stationary if his stochastic properties do not depend on absolute time.
Strict sense stationarity SSS applies to probability density function PDF
Wide sense stationarity WSS $\mu_x = const$

$$R_x(t+\tau,\ t) = R_x(\tau) = E[x(t+\tau)x^*(t)] \tag{1.31}$$

Consequence of equation (2.29):

$$\begin{aligned} R_x(-\tau) &= E[x(t-\tau)x^*(t)] \\ &= E[x(t-\tau+\tau)x^*(t+\tau)] \\ &= \left(E[x^*(t)x^*(t+\tau)]\right)^* \\ &= R_x^*(\tau) \end{aligned} \tag{1.32}$$

For *power spectrum density (PSD)* is obtained by (*Khinchin theorem*):

$$S_x(f) = \mathscr{F}[R_x(\tau)] \tag{1.33}$$

Definition 1.3. Gaussian White Process that has normal distribution and his PSD is constant for all frequencies. Usually:

$$S_x(f) = \frac{N_0}{2} \tag{1.34}$$

According to equation 1.34 correlation function is:

$$R_w(\tau) = \frac{N_0}{2}\delta(\tau) \tag{1.35}$$

"Name" white stands for zero mean:

$$\mu_w = 0 \tag{1.36}$$

1.3.1 Orthogonal expansion of random processes

Definition 1.4. [Sý16b] An $\mathbb{L}_2$ space is space of functions which satisfy condition:

$$\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty \tag{1.37}$$

Definition 1.5. Linear expansion of $x(t)$ in $\mathbb{L}_2$

$$\hat{x}(t) = \sum_n x_n \zeta_n(t) \tag{1.38}$$

Where $\hat{x}(t)$ is approximation of random process $x(t)$ ans $\zeta_n(t)$ is set of linearly independent function, which form basis B.

Let us find $\mathbf{x}$ that minimizes error norm:

$$\begin{aligned} &||\mathbf{x} - \sum_n x_n \zeta_n(t)||^2 = \\ &= ||\mathbf{x}||^2 + \sum_n \sum_{n'} x_{n'}^* \langle \zeta_n,\ \zeta_{n'} \rangle x_n - \sum_n x_n^* \langle \mathbf{x},\ \zeta_n \rangle - \sum_n x_n \langle \mathbf{x},\ \zeta_n \rangle^* \\ &= ||\mathbf{x}||^2 + \mathbf{x}^H \mathbf{G}^H \mathbf{x} - \mathbf{x}^H \mathbf{u} - \mathbf{u}^H \mathbf{x} \end{aligned} \tag{1.39}$$

Where $\mathbf{G}$ is Gram matrix of inner products $[\mathbf{G}]_{n,n'} = \langle \zeta_n,\ \zeta_{n'} \rangle^*$ and $\mathbf{u} = [..., \langle \mathbf{x},\ \zeta_n \rangle, ...]^T$
Find minimum with generalized derivative over vector:

$$\frac{\tilde{\partial}}{\tilde{\partial}\mathbf{x}} \left(||\mathbf{x}||^2 + \mathbf{x}^H \mathbf{G}^H \mathbf{x} - \mathbf{x}^H \mathbf{u} - \mathbf{u}^H \mathbf{x} \right) = \mathbf{G}^* \mathbf{x}^* - \mathbf{u}^* \stackrel{!}{=} 0 \tag{1.40}$$

Vector $\mathbf{x}$ minimizing error norm:

$$\mathbf{x} = \mathbf{G}^{-1}\mathbf{u} \tag{1.41}$$

If the basis is orthogonal than we get coefficients as:

$$\begin{aligned} x_n &= \frac{\langle \mathbf{x},\ \zeta_n \rangle}{\langle \zeta_n,\ \zeta_n \rangle} \\ &= \frac{1}{\mathscr{E}_{\zeta_n}} \int_{-\infty}^{\infty} x(t)\zeta_n^*(t)dt \end{aligned} \tag{1.42}$$

Where $\mathscr{E}_{\zeta_n} = \langle \zeta_n,\ \zeta_n \rangle$

Definition 1.6. [Sý16b]Complete Basis with relation to (w.r.t) correlation function
t_2=const, $R_x(t_1,\ t_2)$:

$$R_x(t_1,\ t_2) = \sum_n r_n(t_2)\zeta_n(t_1) \tag{1.43}$$

For two dimensions:

$$R_x(t_1,\ t_2) = \sum_n \sum_{n'} r_{n,n'}(t_2)\zeta_n^*(t_2) \tag{1.44}$$

Joint 2nd order moment of coefficients:

$$\begin{aligned} E[x_n x_{n'}^*] &= E\left[\frac{1}{\mathscr{E}_{\zeta_n}\mathscr{E}_{\zeta_{n'}}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1)x^*(t_2)\zeta_n(t_1)\zeta_{n'}^*(t_2)dt_1dt_2 \right] \\ &= \frac{1}{\mathscr{E}_{\zeta_n}\mathscr{E}_{\zeta_{n'}}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R_x(t_1,\ t_2)\zeta_n(t_1)\zeta_{n'}^*(t_2)dt_1dt_2 \\ &= r_{n,n'} \end{aligned} \tag{1.45}$$

Theorem 1.7. *[Sý16b] If the basis is complete (w.r.t) correlation function then:*

$$E\Big[|x(t)|^2\Big] = E\Big[|\hat{x}(t)|^2\Big] \tag{1.46}$$

Proof.

$$\begin{aligned} E\Big[|\hat{x}(t)|^2\Big] &= E\left[\sum_n \sum_{n'} x_n x_{n'}^* \zeta_n(t)\zeta_{n'}^*(t) \right] \\ &= \sum_n \sum_{n'} r_{n,n'}\zeta_n(t)\zeta_{n'}^*(t) = \\ &= R_x(t,\ t) \end{aligned} \tag{1.47}$$

□

Theorem 1.8. *If the basis is complete (w.r.t) correlation function then:*

$$E\Big[x(t)\hat{x}^*(t)\Big] = E\Big[|x(t)|^2\Big] \tag{1.48}$$

Proof. [Sý16c]

$$\begin{aligned} E\Big[x(t)\hat{x}^*(t)\Big] &= E\Big[x(t)\sum_{n'} x^*_{n'}\zeta^*_{n'}(t)\Big] \\ &= E\left[x(t)\sum_{n'}\frac{1}{\mathscr{E}_{\zeta_{n'}}}\left(\int_{-\infty}^{\infty} x^*(t')\zeta_{n'}(t')dt'\right)\zeta^*_{n'}(t)\right] \\ &= \sum_{n'}\left(\frac{1}{\mathscr{E}_{\zeta_{n'}}}\int_{-\infty}^{\infty} R_x(t,\ t')\zeta_{n'}(t')dt'\right)\zeta^*_{n'}(t)\Big] \\ &= R_x(t,\ t) \end{aligned} \tag{1.49}$$

□

Theorem 1.9. *[Sý16c] If the basis is complete (w.r.t) correlation function then:*

$$E\Big[|x(t)-\hat{x}(t)|^2\Big] = 0 \tag{1.50}$$

Proof.

$$\begin{aligned} E\Big[|x(t)-\hat{x}(t)|^2\Big] &= E\Big[|x(t)|^2 + |\hat{x}(t)|^2 - 2\mathscr{R}[x(t)x^*(t)]\Big] \\ &= R_x(t,\ t) + R_x(t,\ t) - 2R_x(t,\ t) \\ &= 0 \end{aligned} \tag{1.51}$$

□

1.3.2 Sampling of deterministic signal

For real-valued signals the sampled signal can be uniquely represented when sampled with rate:

$$f_s \geq 2f_m \tag{1.52}$$

Where f_m is maximum frequency contained in signal.

1.3.3 Sampling of random signal

[Sý16c] Sampling of random signal is special case of orthogonal expansion where basis functions are:

$$\zeta_n(t) = \frac{\sin\left(\frac{t-nT_s}{T_s}\right)}{\frac{t-nT_s}{T_s}} = \text{sinc}\left(\frac{t-nT_s}{T_s}\right) \tag{1.53}$$

Where $T_p = \frac{1}{f_s}$ is sampling period. By sampling band-limited signal ($W \leq f_p$, W is width of signal frequencies) and assuming complete basis w.r.t. correlation function we get:

$$R_x[k,j] = R_x(kT_P,\ jT_p) \tag{1.54}$$

The expansion:

$$x(t) = \sum_n x_n \text{sinc}\left(\frac{t-nT_s}{T_s}\right) \tag{1.55}$$

The expansion coefficients can be obtained from:

$$x_n = \frac{1}{T_p}\int_{-\infty}^{\infty} x(t)\text{sinc}\left(\frac{t-nT_s}{T_s}\right)dt \tag{1.56}$$

Because of theorem 1.9 expansion coefficients can be obtained directly as samples:

$$x_n = x(nT_p) \tag{1.57}$$

1.3.4 The Karhuenen-Loeve Expansion

Theorem 1.10. *[Pro95] Theorem states that for a random process $x(t)$ with zero mean and indexed over finite interval $[a,\ b]$ with correlation function $R_x(t_1,\ t_2)$ we can write:*

$$\int_a^b R_x(t_1,\ t_2)\zeta_n(t_1)dt_1 = \lambda_n\zeta_n(t_2) \tag{1.58}$$

Where λ_n are eigenvalues and ζ_n are eigenfunctions. Expansion coefficients :

$$E[x_n] = 0 \tag{1.59}$$

$$E[x_n x_{n'}^*] = \lambda_n\delta_{n,n'} \tag{1.60}$$

Proof. *Marcer's theorem*: $R_x(t,\ t) = \sum_n \lambda_n |\zeta_n(t)|^2$

$$\begin{aligned} E[x_n] &= E\left[\int_a^b x(t)\zeta_n(t)dt\right] \\ &= \int_a^b E[x(t)]\zeta_n(t)dt \\ &= 0 \end{aligned} \tag{1.61}$$

$$\begin{aligned} E[x_n x_{n'}^*] &= E\left[\int_a^b \int_a^b x(t)x^*(t')\zeta_n^*(t)\zeta_{n'}(t')dtdt'\right] \\ &= \int_a^b \int_a^b R_x(t,\ t')\zeta_n^*(t)\zeta_{n'}(t')dtdt' \\ &= \int_a^b \lambda_{n'}\zeta_n^*(t)\zeta_n(t)dt \\ &= \lambda_{n'}\delta_{n,n'} \end{aligned} \tag{1.62}$$

□

Chapter 2

Digital Modulation

2.1 Introduction

[Pro95] The data are digital stream (usually binary). Our goal is to transmit these data to receiver through communication channel. The channel is obviously not perfect, so data are burdened with its parameters. For example noise, attenuation, fading and interference. To be able to transmit data over communication channel, we have to create signal, that represents our digital data stream. In receiver side we have to be able to do the opposite. This process is call "digital modulation" and "digital demodulation". Modulations can be either:

- Memoryless
- With memory

and

- Linear
- Nonlinear

We will discuss only the most basic ones (linear without memory).

2.1.1 Basic properties

Definition 2.1. [Sý16a] Modulated signal

$$s(t) = \sum_n g(d_n, \sigma_n, t - nTs) = \sum_n g(q_n, t - nTs) \tag{2.1}$$

Linear memoryless modulation

$$s(t) = \sum_n q_n g(t - nTs) \tag{2.2}$$

Where $g(t)$ is modulation pulse, d_n are incoming data, σ_n are states of modulator and $q_n = q(d_n, \sigma_n)$ are channel symbols

Definition 2.2. *Finite state machine*
State equation

$$\sigma_{n+1} = \sigma(d_n,\ \sigma_n) \tag{2.3}$$

Output

$$q_n = q(d_n,\ \sigma_n) \tag{2.4}$$

Definition 2.3. *Nyquist condition*
Orthogonality with relation to sequence position

$$R[n' - n] = \int_{-\infty}^{\infty} g_1(t - nTs) g_2^*(t - n'Ts) dt = 0 \tag{2.5}$$

Definition 2.4. *Constellation Space*
Expansion of signal into orthogonal base

$$\left\{\zeta_n(t - nTs)\right\}_{n'=1}^{N_s} \tag{2.6}$$

where n is sequence position and n' is index dimension

Definition 2.5. [Sý16a] *Symbol energy*

$$\mathscr{E}_s = \frac{1}{2} \int_{-\infty}^{\infty} |g(\mathbf{d}_n, \boldsymbol{\sigma}_n, t)|^2 dt \tag{2.7}$$

For linear memoryless modulations we can derive:

$$\mathscr{E}_s = \frac{1}{2} |q_n|^2 \tag{2.8}$$

Mean symbol energy

$$\bar{\mathscr{E}}_s = \frac{1}{2} \lim_{M \to \infty} \frac{1}{M} E\left[\int_{-\infty}^{\infty} \sum_{n=0}^{M-1} |g(\mathbf{d}_n, \boldsymbol{\sigma}_n, t - nTs)|^2 dt \right] \tag{2.9}$$

For linear memoryless modulations we can derive:

$$\bar{\mathscr{E}}_s = \frac{1}{2} E[|q_n|^2] \tag{2.10}$$

Definition 2.6. [Sý16a] *Mean symbol Power*

$$\bar{\mathscr{P}}_s = \frac{1}{2} AvE\left[|s(t)|^2\right] \tag{2.11}$$

$$\begin{aligned}
\bar{\mathscr{E}}_s &= \frac{1}{2} \lim_{M\to\infty} \frac{1}{M} E\left[\int_{-\infty}^{\infty} \sum_{n=0}^{M-1} |g(\mathbf{d}_n, \boldsymbol{\sigma}_n, t - nTs)|^2 dt \right] \\
&= \frac{1}{2} \lim_{M\to\infty} \frac{1}{M} E\left[\int_{-\frac{MTs}{2}}^{\frac{MTs}{2}} \sum_{n=-\infty}^{\infty} |g(\mathbf{d}_n, \boldsymbol{\sigma}_n, t - nTs)|^2 dt \right] \\
&= \frac{1}{2} \lim_{M\to\infty} \frac{Ts}{M} E\left[\int_{-\frac{M}{2}}^{\frac{M}{2}} |s(t)|^2 dt \right] \\
&= Ts \bar{\mathscr{P}}_s
\end{aligned} \tag{2.12}$$

2.2 Linear memoryless modulations

2.2.1 PAM (ASK)

Pulse Amplitude Modulation (Amplitude Shift Keying)
Usually discrete values

$$q_n \in \pm\mathbb{Z}, n \in [1,\ M], M \text{ is number of constellation points} \tag{2.13}$$

Symbol energy and mean symbol energy are represented by 2.8 and 2.10
Examples of some M-ASK modulations on following figures.

(a) : 2ASK **(b) :** 4ASK

Figure 2.1: ASK modulations

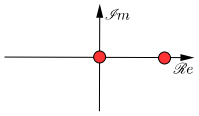


Figure 2.2: On Off Keying

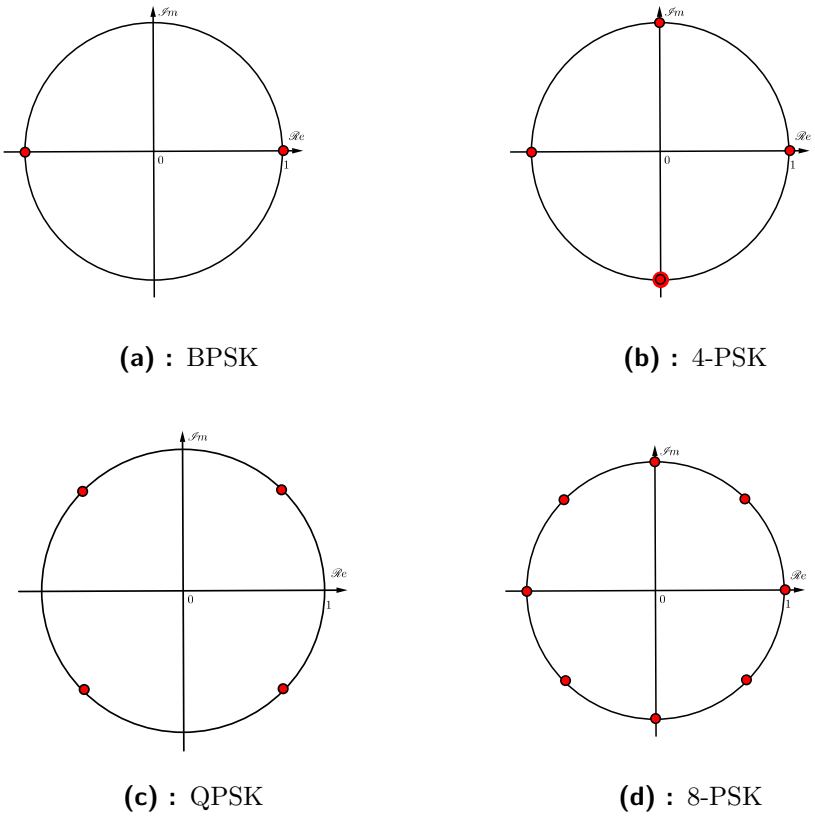
2.2.2 PSK

Phase Shift Keying
Constellation space points are $q_n = e^{j\frac{2\pi}{M}n}$, M is number of constellation points
Symbol energy and mean symbol energy are constants

$$|q_n|^2 = 1 \tag{2.14}$$

$$\mathscr{E}_s = \frac{1}{2} \tag{2.15}$$

$$\bar{\mathscr{E}_s} = \frac{1}{2} \tag{2.16}$$



(a) : BPSK

(b) : 4-PSK

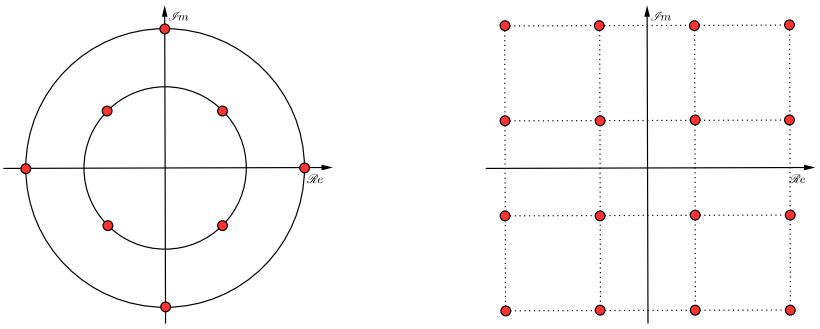
(c) : QPSK

(d) : 8-PSK

Figure 2.3: M-PSK constellation space

2.2.3 QAM

Quadrature Amplitude Modulation [Pro95]
Super position of ASK modulations or PSK and PAM. The constellation space has rectangular shape. Some points may be removed. Examples on fig. 2.4



(a) : 8PAM-ASK **(b) :** 16 QAM

Figure 2.4: QAM modulations

Chapter 3

USRP Ettus Research Hardware Owned by Department of Radio Engineering

3.1 List of hardware

- USRP N210
- OctoClock-G CDA-2990
- Rackmount for USRP N2xx

3.1.1 USRP N210

Universal Software Radio Peripheral enables to design and implement software radio systems. The gigabit Ethernet interface serves as the connection between USRP™ N210 and the host computer. Radios also support full duplex mode. It's possible to realize a complete 2×2 MIMO (multiple input, multiple output) configuration using MIMO cable. Hardware also offers external PPS (Pulse Per Second) and reference input. N210 contains Spartan 3A-DSP 3400 FPGA (Field-programmable gate array) and Daughter Board SBX which is a wide bandwidth transceiver.
Full specification and data sheets are available on [cit17e]. Radio engineering department owns five USRP™ N210.
All of them are working properly (tested on Simulink Example)
Serial numbers and MACS are

- F5F6DA, 00802F160714,
- F5F4B3, 00802F1606D2,
- F4A63A, 00802F0AE761
- E5R24N4UP, A036FA383464
- E2R22NAUP, A036FA38322A

All of them are loaded with firmware uhd-images_003.009.004-rc1. Available on [cit17a]
The main page and manual can be found here [cit17e]
To each USRP™ N210 department owns TP-LINK® TL-ANT2405C Antenna and power supply packaged in the box.

3.1.2 OctoClock

The OctoClock 8-channel clock distribution module can provide both frequency and time synchronization for up to eight USRP™ devices by amplifying and splitting an external 10 MHz reference and PPS (pulse per second) signal 8-ways through matched-length traces. The OctoClock-G adds an internal time and frequency reference using an integrated GPSDO (GPS-disciplined oscillator)
Radio engineering department owns one OctoClock-G CDA-2990. Power supply is also provided. Full specification is available here [cit17b]
Serial number and MAC

- F61C34, 00802F1609BC

Part II

Assembling the Experimental Transceiver Platform (DiRaC-ExpTxR)

Chapter 4

USRP N210 Setup

4.1 DiRaC Research Group

Digital Radio Communications Research Group. Available on
`http://radio.feld.cvut.cz/dirac/`

4.2 Main Concept

Users should be able to sent generated message to transmitter, obtain data from receiver and with signal processing algorithms retrieve data given to transmitter. The main idea how to implement this request is following:
Set up communication between modules and host computer. Allow users to upload their signal processing codes. For this purpose create websites. On the host computer create script that will check the given directory with certain period for uploaded files. If the user's code with certain name is found then use his code for message generation and data reconstruction . In addition specify maximum granted time for each user. If he exceeds this amount of time stop his simulation. After finishing receiving, save his results and allow him to download them. Later delete his data to allow another user use the DiRaC-ExpTxR.
Uploading and downloading files will provide PHP scripts. Bash shell scripts will check for files, start Matlab® and control how much time user spent

transmitting and receiving data. And finally Matlab® or Simulink® will execute user's scripts and transfer data between USRP™ modules.

4.3 Software installation for USRP™ and host computer communication

4.3.1 Installation of UHD binaries to Linux system

We start with installing UHD (USRP hardware driver™) software onto Linux host computer. UHD software communicates between USRP™ modules and host computer.

Listing 4.1: Installing UHD

```
sudo apt-get install libuhd-dev libuhd003 uhd-host
```

Then download your chosen binaries from [cit17c] and install them with software manager. I decided to use binary 003.009.004.

4.3.2 Setting up IP configuration for USRP™ hardware

The communication between host computer and USRP™ devices is established with Ethernet interface. The configuration is following:

Listing 4.2: Ipconfing

```
auto lo
iface lo inet loopback
# is for automatic configuration after restart.
auto eno1
iface eno1 inet static
address 192.168.10.200
network 192.168.10.0
netmask 255.255.255.0
```

4.3.3 Setting IP for USRP™ hardware

[cit17d] To set up IP addresses is used code:

Listing 4.3: Ipconfing

```
cd <install-path>/lib/uhd/utils
./usrp_burn_mb_eeprom --args=<optional device args>
 --values="ip-addr=192.168.10.104"
```

The addresses of used USRP™ modules in DiRaC-ExpTxR were set to:

- 192.168.10.104
- 192.168.10.105

Addresses of three remaining USRP™ modules are set to:

- 192.168.10.101
- 192.168.10.102
- 192.168.10.103

4.3.4 Loading images on USRP™ N210

[cit17d] First we have to download uhd-images from [cit17a] and then load them onto the on-board flash using commands

Listing 4.4: Installing Images

```
uhd_image_loader --args="type=usrp2,addr=<IP address>"
 --fw-path="<firmware path>" --fpga-path="<FPGA path>"
```

Where FPGA path is path to downloaded *uhd_images*_003.009.004
To find all the connected devices we can use following command

Listing 4.5: Find all devices

```
uhd_find_devices
```

Now we have the USRP™ modules ready and we can develop Shell scripts for controlling the platform.

Chapter 5

Server Setup

5.1 Introduction

Server web pages: `http://147.32.198.81/`
NOTE:
In some of the following scripts had to be added extra lines to fit pdf page. Original ones are saved on enclosed CD.

5.2 Creating websites

5.2.1 Apache2

Apache 2 is great software to create website on server under Linux.

Listing 5.1: Installing

```
sudo apt install apache2
sudo apt-get install apache2 apache2-utils
```

[dig17b] After installation we have to restrict access to unauthorized users. This is done with "htaccess". We create file ".htpasswd" in folder /etc/apache2/.

Than we add users with commands:

Listing 5.2: Creating file with users and passwords

```
sudo htpasswd /etc/apache2/.htpasswd another_user
```

Than we have to add code:

Listing 5.3: Using .htpasswd

```
AuthType Basic
AuthName "Restricted Content"
AuthUserFile /etc/apache2/.htpasswd
Require valid-user
```

Log-in user name is set to "usrpweb".
To the file /etc/apache2/sites-enabled/000-default.conf We also create aliases for pages we will need in future

- Alias "/subfolder" "/var/www/subfolder/html"
  For upload/download page.
- Alias "/uploads" "/var/www/subfolder/uploads"
  For folder to store uploaded Matlab® codes.
- Alias "/downloads" "/var/www/subfolder/downloads"
  For downloading results from simulations.
- Alias "/simstatus" "/var/www/subfolder/simstatus"

All of these pages and main page will be hidden after "htaccess".

5.2.2 Installing PHP

[dig17a] Code

Listing 5.4: Installing PHP

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

Than move "index.php" in the file "/etc/apache2/mods-enabled/dir.conf" to the beginning of the index to be the first priority.

5.2.3 Sites

Upload/Download

Main function of this site is to display files in directories "downloads" and "uploads" and restrict access to only one user. [PHP17] Restrict access is done by saving session ID into upload folder. If anyone else tries to load page, the script searches for file "mutex.txt." If it exists and the ID saved in "mutex.txt" is not the same as your session ID, this site is not shown for you. You have to wait until other user finishes his work.
Opening directories is done by using "opendir" until there are no files to open. There are added some restrictions to shown files. Only files with Matlab® extensions are shown. Files "mutex.txt" and "USRPreceive.m" (restricted Matlab® for communication with USRP™s) files are not open.
Next part of the code is for allowing user to pre-upload files thanks to the buttons that the code provides.
In addition there is a link to page with simulation status text file, which will be explained later.
The last part of the code is for opening "downloads" directory. It has similar restrictions and realization as "uploads" directory.

```
<!DOCTYPE html>
<html>
<body>
<h1>CTU FEL in PRAGUE, Department of Radio Engineering</h1>
<h2>USRP Etuss Upload/Download page</h2>


<?php

session_start();
$sid = session_id();
$fileName = '/var/www/subfolder/uploads/mutex.txt';

if (!file_exists($fileName)) {
   file_put_contents($fileName, serialize($sid));
  }
$id = unserialize(file_get_contents($fileName));

if ($id !== $sid) {
echo "Platform is used by another user. Please try later";
$mutex = 0;
} else
```

```
{
echo "Session OK";


echo "<h4>Files uploaded</h4>";

$dir = "/var/www/subfolder/uploads/";

// Open a known directory, and proceed to read its contents
if (is_dir($dir)) {
    if ($dh = opendir($dir)) {
        while (($file = readdir($dh)) !== false) {
        $imageFileType = pathinfo($file,PATHINFO_EXTENSION);
if(($imageFileType == "m" or $imageFileType or "fig"
&& $imageFileType == "mlx"
or $imageFileType == "mat" or $imageFileType == "mdl" or
 $imageFileType == "slx") && $file != "USRPreceive.m"
&& $file != "mutex.txt") {
    $link = "<a href='/uploads/$file'> $file </a><br />";
        echo $link;
    echo "<br />";
}
        }
        closedir($dh);
    }
}

echo "<h4>Upload files</h4>";

//Preparation for upload and buttons
echo "<form action=\"upload.php\" method=\"post\" enctype=
        \"multipart/form-data\">
    Select image to upload:
    <input type=\"file\" name=\"fileToUpload[]\"
         multiple id=\"fileToUpload\">
    <input type=\"submit\" value=\"Upload Files\"
         name=\"submit\">";
//
echo "<h4>Simulation Status</h4>";
echo "<a href=\"/simstatus\">View Simulation Status</a>";


echo "<h4>Files to download</h4>";

$dir = "/var/www/subfolder/downloads/";
```

```
// Open a known directory, and proceed to read its contents
if (is_dir($dir)) {
    if ($dh = opendir($dir)) {
        while (($file = readdir($dh)) !== false) {
    if($file != "." && $file != ".." &&
        $file != "sdruqpsktx_top_vm.bc" &&
        $file != "sdruqpskrx_top_vm.bc"){
    //Doesnt show . , .. and
    //files created for communiacation with USRP
    $link = "<a href='/downloads/$file'> $file </a><br />";
        echo $link;
    echo "<br />";
}
        }
        closedir($dh);
    }
}
}

?>

</body>
</html>
```

Uploading page

The next page provides the upload itself. The variable "uploadOk" is set to FALSE if any of the following file checks doesn't satisfy one of the conditions. Code inspects each file one by one.
[PHP17] First condition checks whether an error occurred during previous update. If not returns "File "name" is not OK".
The second part checks for restricted names. Announces if the user uploaded restricted name. For this check is created variable "uploadOk2" because for other files code allows overwriting. This allows the next part of the code.
Follows size check and file extension. If the file passes all the inspections it is saved into the directory "uploads".
The last part are links to Simulation status and previous page.

```
<html>
<body>

<h1>Upload Page</h1>
```

```
<?php
$target_dir = "/var/www/subfolder/uploads/";
$uploadOk = 1; //Check variable if files should be uploaded

// Check files
if(isset($_POST["submit"])) {
$i=0;
foreach ($_FILES["fileToUpload"]["error"] as &$check){
    if($check == 0) {
    echo "File ";
        echo basename($_FILES["fileToUpload"]["name"][$i]);
        echo " is OK";
        echo "<br />";
        $uploadOk = 1;
    } else {
    echo "File ";
        echo basename($_FILES["fileToUpload"]["name"][$i]);
        echo " is not OK.";
        $uploadOk = 0;
        echo "<br />";
    }
    //Check for every file
    $target_file = $target_dir . basename($_FILES["fileToUpload"]
        ["name"][$i]);

    if (basename($_FILES["fileToUpload"]["name"][$i])
         == "USRPreceive.m"){
        echo "File ".basename($_FILES["fileToUpload"]
                ["name"][$i])." is restricted name!";
        echo "<br />";
        $uploadOk = 0;
        $uploadOk2= 0; //checks only for rrstricted names
        }
        else{
        $uploadOk2= 1;}
        //checks for alrady existing files
    if (file_exists($target_file) && $uploadOk2 == 1) {
    unlink($target_file);
    echo "File ".basename($_FILES["fileToUpload"]
        ["name"][$i])." already existed.
        File was replaced by new one!";
    $uploadOk = 1;
    echo "<br />";
    }
// Check file size
```

```
if ($_FILES["fileToUpload"]["size"][$i] > 5000000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
        echo "<br />";
}
//Allow certain file formats
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
if($imageFileType != "m" && $imageFileType != "fig"
&& $imageFileType != "mlx" && $imageFileType != "mat"
&& $imageFileType != "mdl" && $imageFileType != "bmp"
&& $imageFileType != "slx") {
    echo "Sorry, only Matlab files are allowed.";
    $uploadOk = 0;
    echo "<br />";
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
    echo "<br />";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]
        ["tmp_name"][$i], $target_file)) {
         echo "The file ".basename($_FILES["fileToUpload"]
        ["name"][$i])." has been uploaded.";
         echo "<br />";
    } else {
        echo "Sorry, there was an error uploading file.";
        echo "<br />";
        }
    }
    $i=$i+1;
    echo "<br />";
  }
}
?>
<p><a href="/simstatus">View Simulation Status</a></p>
<p><a href="/subfolder">Back to prev page</a></p>
</body>
</html>
```

Simulation Status

This page displays "status.txt" to which is being saved output of the simulation including Matlab® and Shell scripts. This page is being refresh every 10 seconds. Which obviously means that the output of the simulation is shown to a user at maximum with 10 seconds delay.

```
<!DOCTYPE html>
<?php
$page = $_SERVER['PHP_SELF'];
$sec = "10";
?>
<html>
<body>
<h1>CTU FEL in PRAGUE, Department of Radio Engineering</h1>
<h2>USRP Ettus/Simulation Status Page</h2>
<meta http-equiv="refresh" content="<?php
   echo $sec?>;URL='<?php echo $page?>'">
<?php
$file_handle =
fopen("/var/www/subfolder/simstatus/status.txt", "r");
while (!feof($file_handle)) {
   $line = fgets($file_handle);
   echo $line;

echo "<br />";
}
fclose($file_handle);
?>
<p><a href="/subfolder">
Back to USRP Etuss Upload/Download page</a></p>
</body>
</html>
```

5.3 Shell scripts for Matlab® and files control

I studied basic Linux commands and and Shell script syntax from [Bra96] and [Tut17]
Following scripts are used to run Matlab® scripts containing signal processing codes and manage unnecessary files remaining after simulation, setting maximum time each user can use and simulation status. For better lucidity

the problem is distributed into 4 scripts. The hierarchy is following:

- RunMain.sh
  - Main.sh
    - Timer.sh
    - RunMatlab.sh

RunMain.sh

"Runmatlab.sh" is an infinity loop, which searches for name "SourceCodeSimulink.m" or "SourceCodeScript.m" depending whether user's code uses Simulink® or Matlab® script. If the script doesn't find any of them it sleeps for 20 seconds.
In addition it handles the "mutex.txt" file. If the file is older than 6 minutes it definitely means that someone just loaded the "Upload/Download" page, did not started simulation and blocks DiRaC-ExpTxR for other users. To "unblock it" the script just deletes the "mutex.txt" file which allows another user to load the page and simulate.

```
#!/bin/bash
HowOften=20 #How often should server check for files

while true; #Infinite loop
do
        if [ -f "/var/www/subfolder/uploads/SourceCodeSimulink.m" ]
#If the Simulink Code name file exists on
        then
                export CodeType=1
                /home/hermes/Desktop/spustit/Main
                 > /var/www/subfolder/simstatus/status.txt 2>&1
        elif [ -f "/var/www/subfolder/uploads/SourceCodeScript1.m" ]
#If the Matlab Script Code name file

        then
                export CodeType=2
                /home/hermes/Desktop/spustit/Main
 > /var/www/subfolder/simstatus/status.txt 2>&1
        else
                if [[ $(find "/var/www/subfolder/uploads/mutex.txt"
                          -mmin +1 -print) ]]  > /dev/null 2>&1; then
                        rm -rf /var/www/subfolder/uploads/mutex.txt
                fi #if mutex.txt is older than 6 minutes,
```

```
# it means that platform us iddle,
# so it removes restriction
        fi
wait
sleep $HowOften
done
```

Main.sh

Main purpose of this script is to determinate starting time and ending time, handle file removal, running "Timer.sh" and "RunMatlab.sh" scripts.
Variables are exported to subscripts. Announce user that DiRaC-ExpTxR has been initiated and waits for user to finish copying his files.
"wait > /dev/null 2>&1" is for eliminating Shell messages about terminating "Timer" or "RunMatlab" scripts.
After the end of the simulation the files created during the simulation in "uploads" folder are moved to "downloads" folder for user comfort. Waits specified time and removes users files to all access for other users. If user that just finished wants to retry his simulation, he has to load his files again.
Last part of code write to "simstatus.txt" information, that to simulation has not stared yet.

```
#!/bin/bash
HowLong=20
#How long should server wait for user
#to upload all the files his code needs
WaitTime=60
export max=300
#maximum seconds of simulation allowed
export SleepTime=1
#How long should loop sleep before another iteration
export EchoTimer=10
#How often do we want announcement about simulation time elapsed
echo "DiRaC-ExpTxR platform initiated. Waiting $HowLong seconds
         to load all your remaining files needed for your code"
#sleep $HowLong
touch -t $(date '+%Y%m%d%H%M.%S') start
#Saves start time to determinate what are results

/home/hermes/Desktop/spustit/Timer &
#Runs timer
export pidTimer=$!
#Export PID of timer to RunMatlab script
```

```
/home/hermes/Desktop/spustit/RunMatlab &
#run Matlab script
wait > /dev/null 2>&1
#Waits until both scripts are finished
#(One is terminated by the other)

touch -t $(date '+%Y%m%d%H%M.%S') stop #Saves stop time
find /var/www/subfolder/uploads -newer start
\! -newer stop -type f -exec mv {} /var/www/subfolder/downloads \;
 #Find all files created between start and stop time
 #and copies them into results folder
echo "You have $WaitTime seconds to copy your results!"
sleep $WaitTime #Time for user to copy his results
        rm -rf /var/www/subfolder/downloads/*
         #Remove last users results
        rm -rf /var/www/subfolder/uploads/*
         #Remove last users Matlab Code
        cp /home/hermes/Desktop/spustit/USRPreceive.m
          /var/www/subfolder/uploads/
                #copy last nessesary files
        cp /dev/null /var/www/subfolder/simstatus/status.txt
         #clears simulation status
        echo "SIMULATION HAS NOT STARTED!"
        >> /var/www/subfolder/simstatus/status.txt
exit
```

Timer.sh

This script allows user to use DiRaC-ExpTxR only for limited time. If the user exceeds the maximum allowed time, than this scripts terminates Matlab® and script "RunMatlab". In addition this script prints simulation time to ensure the user, that there is something happening.

```
#!/bin/bash
for i in `seq 1 $max` #simulation time announcement
        #loop (can be commented out if not necessary.
do
        #echo "$i"
        if [ $(($i % $EchoTimer)) -eq 0 ]
        then
           echo
"Script has been running for $(($SleepTime*$i)) seconds."
        fi
```

```
        sleep $SleepTime
done

echo "Matlab terminated. Reason: $(($SleepTime*$max))
 seconds of simulation are maximum allowed!"
#Echo about reaching maximum simulation time
pkill RunMatlab #Kills script to prevent false announcements
pkill MATLAB #Stop simulation in Matlab
exit
```

5.3.1 RunMatlab.sh

This scripts opens Matlab® depending on variable "CodeType" exported from "RunMain" script. Depending on users selection between using Simulink® or Matlab® scripts. If the user is using Matlab® scripts than it is necessary to run receiver and transmitter in different Matlabs. The reason is absence of "Parallel Symbolic Toolbox" which will be added in future. Simulink® option will be removed in future because Simulink codes cannot run without Matlab® graphics which adds some problems. We will discuss these problems later.

```
#!/bin/bash
if [ $CodeType -eq 1 ]
then
      /home/hermes/matlab/bin/matlab -r
"run /home/hermes/Desktop/spustit/main.m;" #Run simulink main
elif [ $CodeType -eq 2 ]
then
/home/hermes/matlab/bin/matlab -nodisplay -nosplash -r
/var/www/subfolder/uploads/SourceCodeScript1.m
&
/home/hermes/matlab/bin/matlab -nodisplay -nosplash -r
/var/www/subfolder/uploads/SourceCodeScript2.m
#in the case of matlab script there must be two
# Matlab instances opened
#(absence of Parallel Computing Toolbox in Matlab)
else
        echo "An error has occured"
fi
#kill $pidTimer #If the simulation
#time ends before maximum time allowed, kills the timer
pkill Timer
echo "The simulation has finished!"
```

```
exit
```

5.3.2 Execute script "RunMain" on Start-up

Pilot Solution

Because of Simulink® request for graphics interface, we will use this solution. To run script on start-up usually you have to add your desired "script.sh" to directory /etc/init.d and then create symbolic link to this script in directory /etc/rcX.d .
Because of Simulink® we can't use this option. So we always need the terminal running. This can be solved by using:

Listing 5.5: Options

```
mate-session-properties
```

and adding command

Listing 5.6: Inserting Code

```
mate-terminal -e /home/hermes/Desktop/RunMain
```

This guarantees to run script in start-up.

5.4 Matlab® scripts for USRP™ control

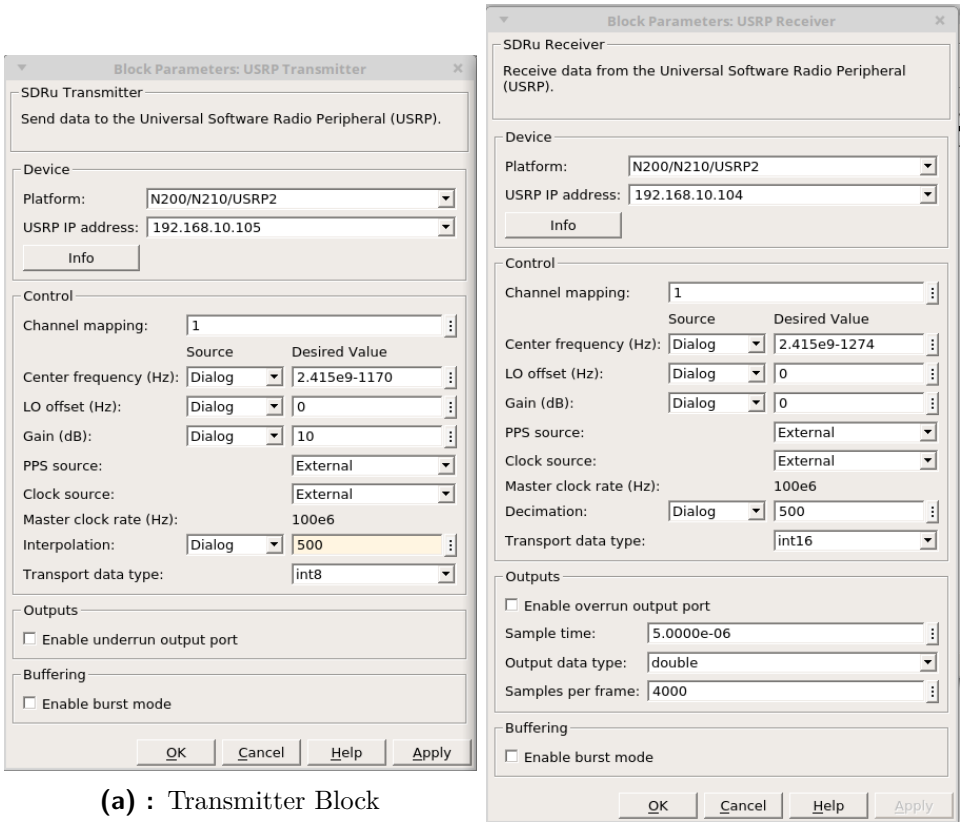
As mentioned before we can control USRP™ either from Simulink® or Matlab® script. In Simulink® it's pretty simple. Just add USRP™ transmitter block and receiver block into two separate models, include signal processing blocks and code generation scripts. Simulink® also allows you to add Matlab® scripts. If you are using Maltab code to communicate with USRP™ you have to add extra rows of code into your own one.
Every user (either using Simulink® or Matlab®) has to add to his own script (or model) **option to save his results**. Otherwise he will not know whether his algorithms were able to successfully transmit and receive the same data.

5.4.1 Simulink®

If you are using Simulink®, you can just download transmitter and receiver blocks from DiRaC-ExpTxR websites 5.1 and add them to the beginning or end of the models. It is recommended to download and install Support Package for USRP™ module from [pac17], because otherwise you will not be able to see settings of the blocks in your model.
Example figure 5.1. For full documentation see [Tra17] and [Rec17]
For Rx and Tx synchronization leave PPS and clock sources to external.
Interpolation in transmitter block should be the same value as decimation in receiver block.
It is necessary to set the IP adresses to 192.168.10.105 and 192.168.10.104.
Channel Mapping has to be set to 1.
You can increase Gain, change your sample time and samples per frame.
You can change transport data type. Int8 - use 8 bit transport achieves higher transport speed, int16 achieves higher precision.
Some parameters can be set from input port instead of dialog.
Overrun output port indicates if one or more packets were dropped when the module transfers data to the host. Return zero when no data packets were lost, one indicates that some packets were dropped during simulation.
You can download examples from MathWorks site. [exa17]. But first you have to install Support Package for USRP™ Radio.
If you want to use platform via Simulink® you have to create Matlab® script that opens your models and starts the simulation named "SourceCodeSimulink.m". In Addition you have to keep this script running until the simulation end. Example how to accomplish this:

```
models={'sdruqpsktx.mdl','sdruqpskrx.mdl'};
load_system(models)
set_param('sdruqpsktx','SimulationCommand','start')
set_param('sdruqpskrx','SimulationCommand','start')
while true
  if (strcmp((get_param('sdruqpsktx','SimulationStatus')),...
        'running')==0 &
       strcmp((get_param('sdruqpskrx','SimulationStatus')),...
       'running')==0)
       break
    end
    pause(5);
end
```

You can download this script on DiRaC-ExpTxR web pages 5.1. Then you just have to upload all the files necessary to "Upload/Downloads" page (link on 5.1) and wait for platform to initiate. When initiated platform waits 20 seconds to finish copying your files. You can check the simulation status by



(a) : Transmitter Block

(b) : Receive Block

Figure 5.1: Blocks set-up

clock on "View Simulation Status". This page refreshes every 10 seconds. Maximum simulation time is set to 5 minutes.
Always add an STOP condition to your model. Otherwise you will have to wait until your simulation will be terminated by script for exceeding maximum allowed time. After the simulation ends you can download your results from "Uploads/Downloads/ page. After 60 seconds your files will be deleted. If you want to retry your code you have to upload your files again.

5.4.2 Matlab® script

If you want to use Matlab® script you have to use names "SourceCodeScript1" for transmitter and "SourceCodeScript2" for receiver.

Transmitting

You have to add the following code to your own and name it "SourceCode-Script1".

```
%///////////////TRANSMIT FUNCTION FROM USRP////////////////////
function SourceCodeScript1()
persistent radioTx
if isempty(radioTx) %SETS UP TRANSMITTER
      radioTx = comm.SDRuTransmitter(...
        'Platform','N200/N210/USRP2', ...
        'IPAddress','192.168.10.105', ...
        'CenterFrequency',2.415e9-1170, ...
        'Gain',10, ...
        'InterpolationFactor',  500, ...
        'ClockSource', 'External', ...
        'TransportDataType','int8', ...
        'ClockSource','External', ...
        'PPSSource','External');
    ContinueTransmit=true;
    %only when you use while loop
end
  %stop transmitting, can be done by
  %by while or IF loops, thats up to
  %to user
  while ContinueTransmit==true
    %////////////////////////////////////

     %YOUR CODE GENERATION
    data=
    %///////////STOP IT HERE/////////////////

    % Data transmission
    step(radioTx, data);
  end

  %STOP TRANSMITTING
  release(radioTx);
  exit
end
```

Variable "RadioTx" sets up the USRP™ module using function "comm.SDRuTransmitter". Parameters are the same as using Simulink® block.
You have to add condition when to stop transmitting. That can be done by using "for" or "while" loop.

Function "step" is used for sending data to transmitter. Command "release" is used for USRP™ radio to stop transmitting. Without this command the radio keep transmitting until Matlab® is closed.
You have to insert your code generation as instructed in "SourceCodeScript1.m" found on DiRaC-ExpTxR web pages 5.1
For full documentation of "comm.SDRuTransmitter" and "step" functions see [com17b]

Receiving

To receive you will also have to add additional code to your own and name it "SourceCodeScript2.m". This script will call function "USRPreceive.m" repetitively to receive data. This function input arguments are "ContinueReceive" and "SamplesPerFrame". If the variable "CountinueReceive" is set to one, USRP™ keeps receiving, if set to zero USRP™ stops receiving. Its up to user how to handle that. In the "SourceCodeScript2" provided on websites variable "Counter" handles variable "CountinueReceive". If the "Counter" reaches maximum value, variable "ContinueReceive" is set to zero and the script "USRPreceive" is terminated.
Variable "SamplePerFrame" is used in USRP™ initialization. It sets number of samples per frame. Return variables are the "DataOut" - the data itself, "datalen" and "overrun". Variable "datalen" are dimension of data vector. If the script returns variable "overrun" equal to zero then no data were lost. The value equal or grater than one indicates data loss.
The user is supposed to add his code into given place.

Code explanation

"USRPreceive.m" code

```
%///////////////RECEIVE FUNCTION FROM USRP////////////////////
function [DataOut,datalen,overrun]=USRPreceive...
(ContinueReceive,ReceiverData)

persistent radioRx len
if isempty(radioRx) %USRP setup
radioRx = comm.SDRuReceiver(...
        'Platform',             'N200/N210/USRP2', ...
        'IPAddress',            '192.168.10.104', ...
        'CenterFrequency',      2.415e9-1170, ...
```

```
        'Gain',                 0, ...
        'DecimationFactor',     500, ...
        'SamplesPerFrame',      ReceiverData, ...
        'OutputDataType',       'double', ...
        'TransportDataType',    'int16', ...
        'ClockSource',          'External', ...
        'PPSSource',            'External');
end
len = uint32(0);
%IF LEN <=0 DATA ARE NOT VALID
        while len <= 0 %KEEPS RECEIVING FRAM UNTIL VALID
             [DataOut, len,overrun] = step(radioRx);
        end
        datalen=len;% RETURNS DATA LENGTH


    if ContinueReceive==false%STOPS RECEIVING
    release(radioRx)
    end
end
```

Variable "radioRx" sets up USRP™ module to start receiving using function "comm.SDRuReceiver". Function "step" reads from object "radioRx" incoming data and returns variable "len". If this variable is equal or less than zero it means that data are no valid."Release" function is used to stop receiving data from USRP™.
For full documentation of "comm.SDRuReceiver" and "step" functions, see [com17a].

"SourceCodeScript2" code

```
%ADD THIS TO YOUR RECEIVE SCRIPT
%SET counterMAX TO RECEIVE THAT NUMBER OF FRAMES
%SET SamplesPerFrame TO SUIT YOUR PROCESSING
%////////////////////////////////////////////////
SamplesPerFrame=4000; %Sets framesize
ContinueReceive=true; %Sets Value for Recieve status
counter=1;
counterMAX=100; %sets maximum received frames
while ContinueReceive==true %While true, keeps receiving frames
      if counter==counterMAX
           %If counter reaches maximum change value in
           %ContinueReceive
          ContinueReceive=false;
      end
          [DataOut,datalen,overrun]=...
```

```
            USRPreceive(ContinueReceive,...
            SamplesPerFrame);
            %call function that receives data
            %////////////INSERT YOUR CODE HERE//////////////

              %RECEIVED DATA ARE IN DataOut
              %datalen displays frame length
              %if overrun 0 no data lost

            %/////////////END OF YOUR CODE/////////////////
        counter=counter+1;
end
exit %ADD EXIT TO QUIT MATLAB
```

Chapter 6

Conclusion

We have successfully developed transceiver platform (DiRaC-ExpTxR) suitable for verification of coding and signal processing. Nodes are based on Matlab® and Simulink® processing. Nodes are SISO configurable by Matlab® scripts or Simulink® models. Processing allows both online and offline processing. Synchronization of nodes is accomplished by using external clock and pulse per second source. It is provided by OctoClock-G CDA-2990. DiRaC-ExpTxR platform allows users to upload their processing algorithms using websites created on host computer. Websites are protected with restricted access. Initialization of DiRaC-ExpTxR platform is done automatically by using bash shell scripts. After data are transmitted, sent through channel and processed on receiver side, the results are downloadable from DiRaC-ExpTxR websites. Users have to follow basic instructions, that are explained on websites and this thesis to successfully use DiRaC-ExpTxR. All the algorithm are shown and explained in chapters above. The functionality was verified using Mathworks® examples.
I would like to continue working on this documentation to implement MIMO extensions. DiRaC-ExpTxR platform should be in the future connected with server Helios for better security.

Bibliography

[Bra96] Michal Brandejs, *Unix-linux*, 1 ed., GRADA publishing, Prague 1, 1996.

[cit17a] *Ettus research binaries/images*, `http://files.ettus.com/binaries/images/`, 2017.

[cit17b] *Octoclock-g cda-2990*, `https://www.ettus.com/product/details/OctoClock-G`, 2017.

[cit17c] *Usrp hardware driver and usrp manual*, `https://files.ettus.com/manual/page_install.html`, 2017.

[cit17d] *Usrp hardware driver and usrp manual*, `http://files.ettus.com/manual/page_usrp2.html`, 2017.

[cit17e] *Usrp n210*, `https://www.ettus.com/product/details/UN210-KIT`, 2017.

[com17a] *comm.sdrureceiver system object*, `https://www.mathworks.com/help/supportpkg/usrpradio/ug/comm.sdrureceiver-class.html`, 2017.

[com17b] *comm.sdrutransmitter system object*, `https://www.mathworks.com/help/supportpkg/usrpradio/ug/comm.sdrutransmitter-class.html`, 2017.

[dig17a] *How to install linux, apache, mysql, php (lamp) stack on ubuntu*, `https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu`, 2017.

[dig17b] *How to set up password authentication with apache on ubuntu 14.04*, https://www.digitalocean.com/community/tutorials/how-to-set-up-password-authentication-with-apache-on-ubuntu-14-04, 2017.

[exa17] *Communications system toolbox support package for usrp® radio examples*, https://www.mathworks.com/help/supportpkg/usrpradio/examples.html, 2017.

[pac17] *Install support package for usrp® radio*, https://www.mathworks.com/help/supportpkg/usrpradio/ug/install-support-package-for-usrp-radio.html, 2017.

[PHP17] *Php*, http://php.net/, 2017.

[Pro95] John G. Proakis, *Digital communications*, 3. ed., Osborne-McGraw-Hill, New York, 1995 (English).

[Rec17] *Sdru receiver*, hhttps://www.mathworks.com/help/supportpkg/usrpradio/ug/sdrureceiver.html, 2017.

[Sý16a] Jan Sýkora, *Digital communications*, 2016.

[Sý16b] ———, *Stochastic signal processing fundamentals*, 2016.

[Sý16c] ———, *Stochastic signal processing fundamentals*, 2016.

[Tra17] *Sdru transmitter*, https://www.mathworks.com/help/supportpkg/usrpradio/ug/sdrutransmitter.html, 2017.

[Tut17] *tutorialspoint*, https://www.tutorialspoint.com/unix/, 2017.

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Radioelectronics

BACHELOR PROJECT ASSIGNMENT

Student: **Ladislav Kršek**

Study programme: Open Electronic Systems

Title of Bachelor Project: **Experimental tranceiver platform for verification of coding and processing in radio communications**

Guidelines:

Student will develop an experimental transceiver platform suitable for the verification of coding and signal processing algorithms in radio networks. The nodes will be based on USRP modules and suitable (e.g. Matlab) processing.

Detailed instructions: The work has three parts. (1) Development of USRP based platform of TxR nodes. Nodes should be configurable as scalar Tx/Rx (and optionally as MIMO Tx/Rx, relay) nodes. The concept should allow highly modular configuration settings with high level user comfort. (2) Node processing should allow off-line and on-line processing modes. The off-line modes should include a) fully synchronised IQ complex envelope Tx/Rx upload/download, b) frame-only coarse synchronised Tx/Rx upload/download, c) (optional) multi-stage and multi-mode Tx/Rx upload/download. On-line modes should allow upload preferably matlab based algorithm fragments for real-time processing on nodes. The user interface should provide comfortable (preferably web based) front-end. (3) The student will demonstrate its basic functionality by implementing selected simple coding and processing algorithms.

Bibliography/Sources:

[1] J. G. Proakis: Digital Communications. McGraw-Hill.

Bachelor Project Supervisor: prof. Jan Sýkora Ing., CSc.

Valid until the end of the summer semester of academic year 2017/2018

L.S.

doc. Mgr. Petr Páta, Ph.D.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 27, 2017