VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2016 Pavel Röszler

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

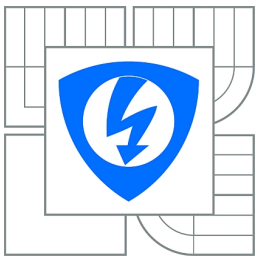
AUTOMATIZACE MĚŘENÍ VLIVU RUŠENÍ NA PŘENOSOVÉ VLASTNOSTI XDSL TECHNOLOGIÍ
AUTOMATION OF MEASUREMENT OF DISTURBANCE INFLUENCE ON XDSL TECHNOLOGIES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Pavel Röszler**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Pavel Šilhavý, Ph.D.**
SUPERVISOR

BRNO 2016



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Pavel Röszler	***ID:***	164384
Ročník:	3	***Akademický rok:***	2015/2016

NÁZEV TÉMATU:

Automatizace měření vlivu rušení na přenosové vlastnosti xDSL technologií

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku rušivých vlivů na přenosové vlastnosti DSL modemů a možnosti jejich testování s využitím generátoru a injektoru rušení Spirent. Vytvořte program, který bude umožňovat zcela automatizované měření závislosti přenosové rychlosti ADSL a VDSL modemů na délce vedení a úrovni daného typu rušení.

DOPORUČENÁ LITERATURA:

[1] CHEN, Walter Y.. Simulation Techniques and Standards Development for DSL Systems. Indianopolis, USA: Macmillan Technical Publishing, 1998. ISBN 1-57870-017-5.
[2] RAUSCHMAYER, D. J.. ADSL/VDSL Principles. Indianopolis, USA: Macmillan Technical Publishing, 1999. ISBN 1-57870-015-9.
[3] POKORNÝ, L.. Automatizované měření přenosových vlastností DSL modemů. Bakalářská práce, VUT v Brně, 2013.

Termín zadání:	1.2.2016	***Termín odevzdání:***	1.6.2016

Vedoucí práce: Ing. Pavel Šilhavý, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se věnuje rušivým vlivům na přenosové vlastnosti DSL modemů a možnosti jejich testování s využitím generátoru a injektoru rušení Spirent. Zabývá se návrhem automatizovaného měření závislosti přenosové rychlosti ADSL a VDSL modemů na délce vedení a úrovni daného typu rušení. Teoretická část se věnuje popisu technologie ADSL a vlivu rušení. Následuje popis měřicí sestavy zahrnující simulátory vedení DLS E414 a 8234, generátor DLS-5800 a injektor DLS-5410DC od firmy Spirent. Podstatná část se věnuje možnostem obsluhy generátoru a injektoru rušení především možnostem pro vzdálené řízení, které je využíváno pro automatizované měření. Jsou uvedeny možnosti využívaného SW xDSL Measure tool pro automatizované měření se simulátory vedení a jsou popsány způsoby doplnění o části, které zajišťují komunikaci a automatizaci změn parametrů generátoru a injektoru.

KLÍČOVÁ SLOVA

Rušení, DSL, ADSL, VDSL, automatizované měření, šum, přeslechy

ABSTRACT

This work focuses on interference of DSL transmission, measurement and testing with Spirent noise generator and injection unit. Deals with proposition of automated measurement of transmission bit rate dependency on wireline lenght and noise type and level. In theoretical part are described ADSL technology and noise types. Following part focuses on measurement setup and description of wireline simulator units E414 and 8234, generator DLS-5800 a injection unit DLS-5410DC by Spirent company. Main part describes generator and injection unit control software and remote control commands, which are used for automated measurement. There are also described xDSL Measure tool software, which is using for automated measurement with wireline simulators and upgrade, which implements remote communication with generator a injection unit, for remote control and automatic changing noise types generated at the output.

KEYWORDS

Interference, DSL, ADSL, VDSL, automated measurement, noise, crosstalks

RÖSZLER, Pavel *Automatizace měření vlivu rušení na přenosové vlastnosti xDSL technologií*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 77 s. Vedoucí práce byl Ing. Pavel Šilhavý, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Automatizace měření vlivu rušení na přenosové vlastnosti xDSL technologií“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Pavlovi Šilhavému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Tato práce se věnuje rušivým vlivům na přenosové vlastnosti DSL modemů a možnosti jejich testování s využitím generátoru a injektoru rušení Spirent. Testování je zcela automatizované a umožňuje měření závislosti přenosové rychlosti ADSL a VDSL modemů na délce vedení a úrovni daného typu rušení.

Teoretická část práce se nejprve věnuje popisu technologie ADSL, především využívaným modulačním technikám a rámcové struktuře. Dále jsou popsány rušivé vlivy, které se u DSL přenosu vyskytují a jejich rozdělení. Pro jednotlivé druhy jsou uvedeny jejich matematické modely.

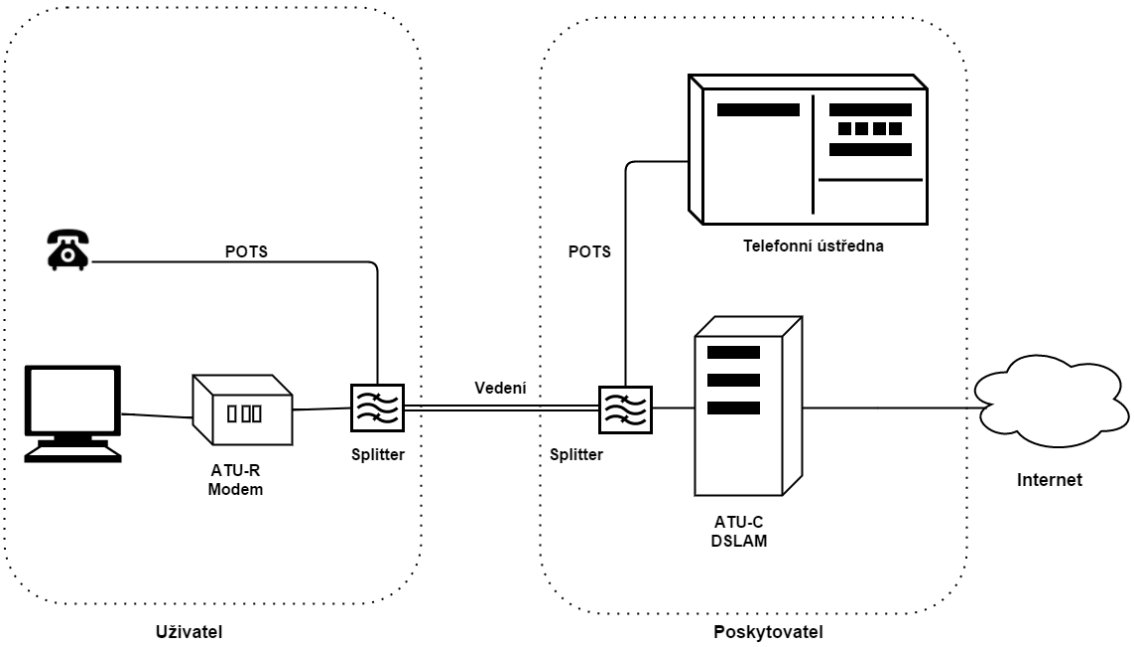
Následuje popis měřicí sestavy, která zahrnuje simulátory vedení, generátor a injektor od výrobce Spirent. V této části jsou uvedeny možnosti obsluhy generátoru a injektoru rušení, především jsou popsány způsoby vzdáleného řízení, které jsou využity při automatizovaném měření.

Programu pro automatizované měření je věnována stěžejní část práce, jsou zde uvedeny jeho možnosti, řešení komunikace, návrh profilů rušení, měřicích sekvencí a popis měřicího procesu. Následně je popsán formát výsledků měření pomocí tohoto softwaru a uvedeny prakticky naměřené příklady pro technologie ADSL a VDLS.

V přílohách této práce je podrobnější návod, kde jsou popsány jednotlivé ovládací prvky a postup při vytváření profilů rušení. Dále je přiložena dokumentace k programu, kde jsou uvedeny a popsány zdrojové kódy nejdůležitějších částí měřicího SW, které byly autorem implementovány.

1 TECHNOLOGIE DSL

DSL (Digital Subscriber Line – digitální účastnická přípojka) je velice rozšířená technologie pro připojení uživatelů (nejčastěji běžné domácnosti) k Internetu. Velkou výhodou je využití stávajícího metalického telefonního vedení, případně koaxiálního připojení kabelové televize. Spektrum je umístěno v kmitočtovém pásmu nad hovorovým pásmem a je odděleno kmitočtovým filtrem tzv. Splitterem, což umožňuje zachování telefonních hlasových služeb. Existuje celá řada variant xDSL technologií, které se historicky vyvíjely a liší se zejména využívanou šířkou pásma a rozdělením pásma pro Upstream – směr od účastníka k ústředně a Downstream – směr od ústředny k účastníkovi. Z toho vyplývají i rozdíly v maximální dosažitelné přenosové rychlosti, maximální vzdálenosti od ústředny, požadovaných parametrech metalického vedení, použitých modulačních technikách apod.

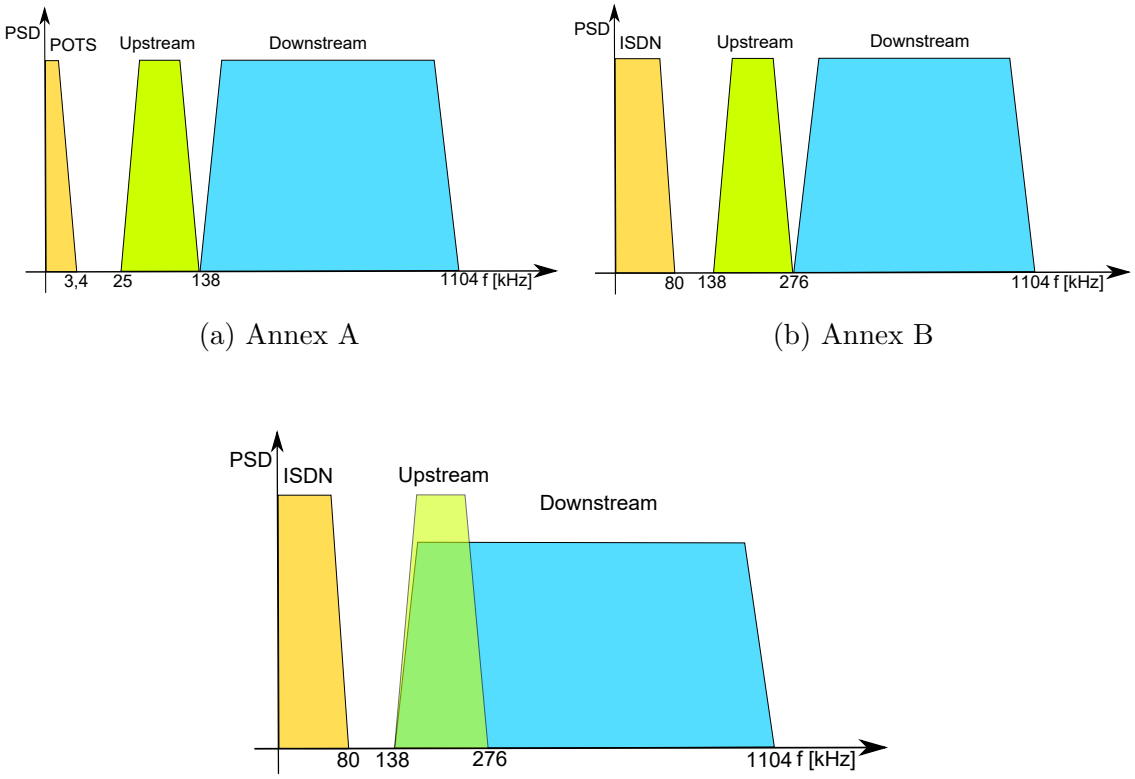


Obr. 1.1: Blokové schéma xDSL technologie

1.1 ADSL

ADSL (Asymmetric Digital Subscriber Line) je technologie využívající oddělené frekvenční pásmo pro upstream a downstream, konkrétně 25–138 kHz (Annex A) nebo 138–276 kHz (Annex B) pro upstream a 138–1104 kHz (Annex A) nebo 276–1104 kHz (Annex B) pro downstream. Annex A je starší modifikace ADSL, kde je zachována tradiční analogová telefonní služba (POTS – Plain ordinary telephone service) v kmitočnovém pásmu 300–3400 Hz, zatímco u novějšího Annexu B je uvažovaný

současný provoz s technologií ISDN. Oddělení přenosu pro upstream a downstream je možné pomocí frekvenčního dělení FDM (Frequency Division Multiplexing), nebo pomocí metody potlačení ozvěny EC (Echo Cancellation), kdy se hranice pásem pro upstream a downstream překrývají a v přijímači je přenos oddělen obvodově pomocí tzv. vidlice a obvodu pro EC. Rozdělení spektra pro různé varianty ADSL technologie je znázorněno na obr. 1.2.



(a) Annex A

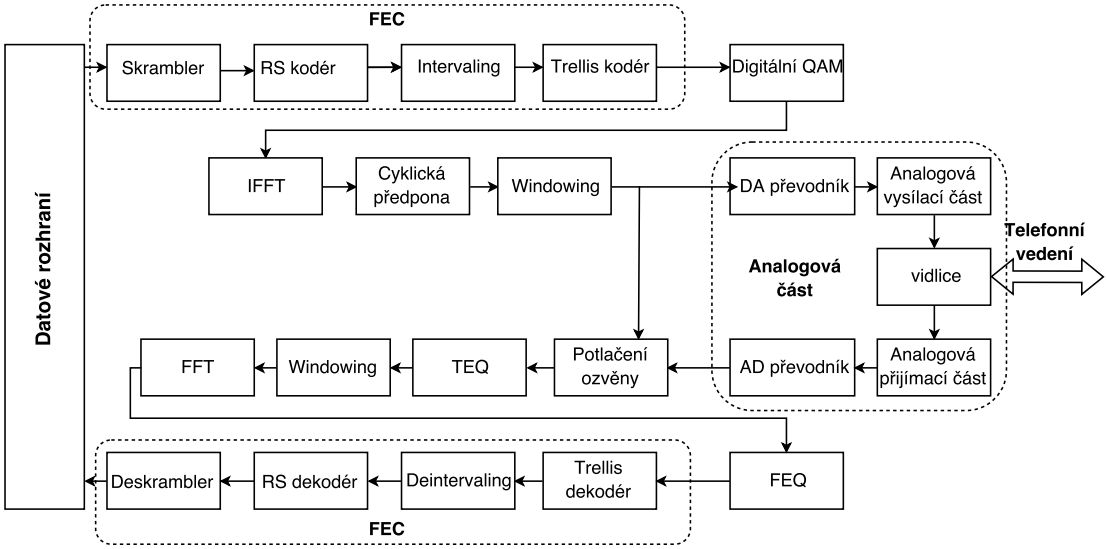
(b) Annex B

(c) Annex B s EC

Obr. 1.2: Varianty ADSL technologie

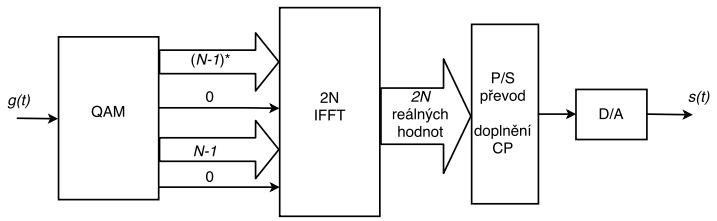
1.1.1 DMT modulátor

ADSL modem, jehož zjednodušené schéma je zobrazeno na obr. 1.3, využívá modulaci DMT (Discrete MultiTone), která rozděluje výše uvedené frekvenční pásmo na 256 subpásem s šířkou 4,3 kHz, v každém subpásmu je využívána mudulace QAM (Quadrature amplitude modulation) s nosným kmitočtem vzdáleným o 4,3125kHz od vedlejší subnosné. Kmitočty jednotlivých subnosných jsou vzájemně ortogonální, maximum jednoho z kmitočtů se překrývá s minimem ostatních frekvencí.



Obr. 1.3: Blokové schéma ADSL modemu

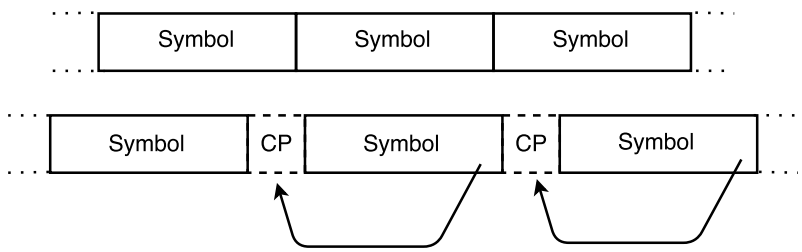
Tato modulace je realizována pomocí algoritmu rychlé Fourierovy transformace (FFT – Fast Fourier Transformation) a inverzní rychlé Fourierovy transformace (IFFT – Inverse Fast Fourier Transformation).



Obr. 1.4: Princip vzniku DMT pomocí IFFT

Vysílaná data $g(t)$ jsou nejprve zabezpečena pomocí FEC (Forward error coding) zahrnující skramblování, zabezpečení Reed-Solomonovým kódem a prokládání. Následuje mapování na jednotlivé subnosné, při inicializaci probíhá adaptivní bitová alokace, kdy je v závislosti na hodnotě SNR v daném pásmu přiřazeno 2–15 bitů pro každou subnosnou a těmto bitům je přiřazený konstelační QAM diagram zahrnující

trellis kódování. Takto dostáváme $N - 1$ komplexních čísel reprezentujících amplitudu a fázi, která jsou doplněna o stejný počet komplexně sdružených čísel a 2 nulové hodnoty odpovídající stejnosměrné složce a zrcadlovému kmitočtu, tento princip je znázorněn na obr. 1.4. Po aplikaci IFFT dostáváme dvě reálné hodnoty v časové oblasti, které jsou převedeny na sériovou posloupnost bitů a doplněny o tzv. cyklickou předponu vložením několika posledních vzorků na začátek, jak je vidět na obr. 1.5. Z výstupu D/A převodníku je výsledný signál s(t) přenesen na vedení pomocí



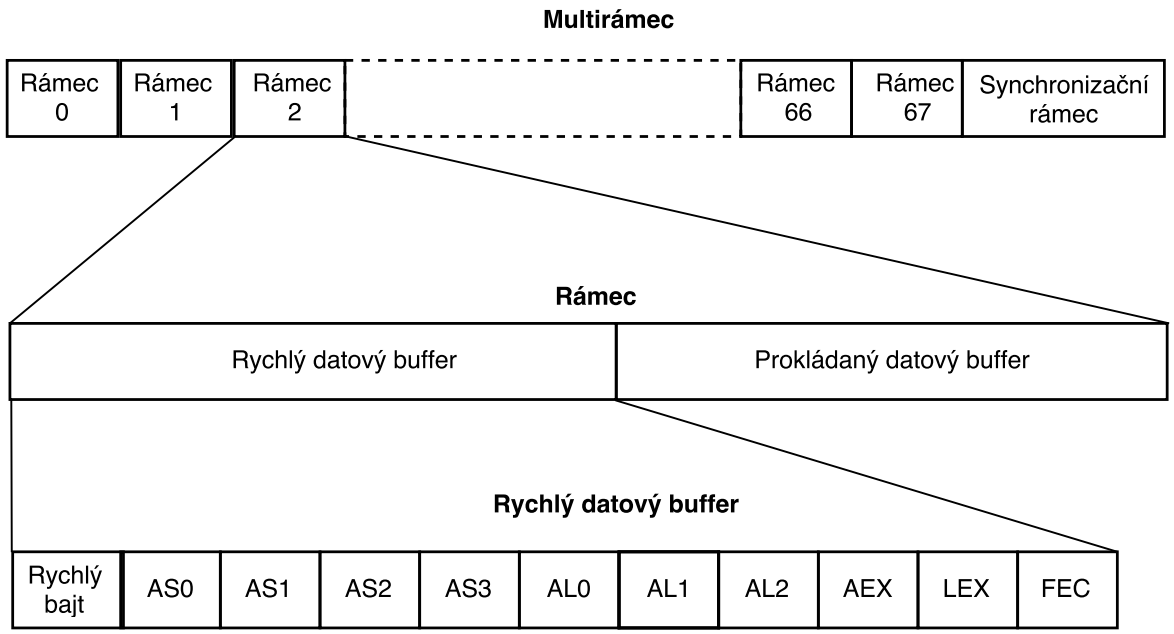
Obr. 1.5: Vkládání CP

Cyklická předpona odděluje symboly a snižuje jejich interferenci a zlepšuje synchronizaci. Vytvořením pseudoperiodického signálu se snižují interference mezi subkanály. Cyklická přípona je následně odstraněna na straně přijímače, nevýhodou je pouze přenášení nepotřebné informace. Na straně přijímače dochází vlivem zkreslení kanálu k rozprostření symbolů a interferencím. Pro vyrovnání kanálu se používají ekvalizery s inverzní impulsovou charakteristikou k přenosovému kanálu. TEQ (Time domain equalizer) zajišťuje zkrácení impulsní odezvy na dobu cyklické předpony pomocí digitálního adaptivního filtru, jehož koeficienty se stanovují při počáteční inicializaci pomocí různých adaptačních algoritmů. Vyrovnávání probíhá i ve frekvenční oblasti, zde FEQ (Frequency domain equalizer) koriguje amplitudu a fázi signálu v jednotlivých subkanálech násobením signálu komplexním číslem, čímž je eliminováno lineární zkreslení přenosového kanálu.

1.1.2 Rámcová struktura

Data mohou být přenášena v simplexním (AS0–AS3) nebo duplexním (LS0–AL2) kanálu, celkově maximálně 7 kanálů. Data z jednotlivých kanálů se vkládají do rámců a rámce jsou přenášeny pomocí multirámce, celý proces je znázorněn na obr. 1.6.

Data jsou dále ukládána do rychlého a prokládaného bufferu a obsah obou bufferů tvoří rámec. Rychlý buffer obsahuje rychlý bajt, data všech kanálů, synchronizační bajty a FEC zabezpečení. Prokládaný buffer zajišťuje bezpečnost proti shlukovým chybám za použití konvolučního prokládání za cenu zvýšeného zpoždění. Jednotlivé rámce jsou vkládány do multirámce, číslované od 0 do 67 a přechod mezi jednotlivými multirámci je oddělený synchronizačním symbolem. CRC kontrola multirámce je provedena pro rychlý a prokládaný bufferu a je vložena do prvního rámce následujícího multirámce.



Obr. 1.6: Rámcová struktura

Tab. 1.1: Přehled xDSL Technologií

Technologie	**ITU-T**	**Přenosová rychlost**	**Frekvenční pásmo**	**max. délka**
ADSL	G.992.1	1,5–8 Mb/s down 16–800 kb/s up	138–1104 kHz down 138–276 kHz up [I]	2000–5500 m
ADSL2+	G.992.5	24 Mb/s down 1,4 Mb/s up	254–2208 kHz down 120–276 kHz up [II]	7000 m
HDSL	G.991.1	1,5–2 Mb/s sym.	0–485 kHz [III]	4000–5500 m
VDSL	G.993.1	55 Mb/s down 3 Mb/s up	0,138–3 a 5,1–7,05 MHz 3–5,1 a 7,05–12 MHz [IV]	300–1350 m
VDSL2	G.993.2	až 100 Mb/s sym.	0,3–30 MHz [V]	300–1350 m

I – G.992.1 Annex B [2];

II – G.992.5 Annex B [3];

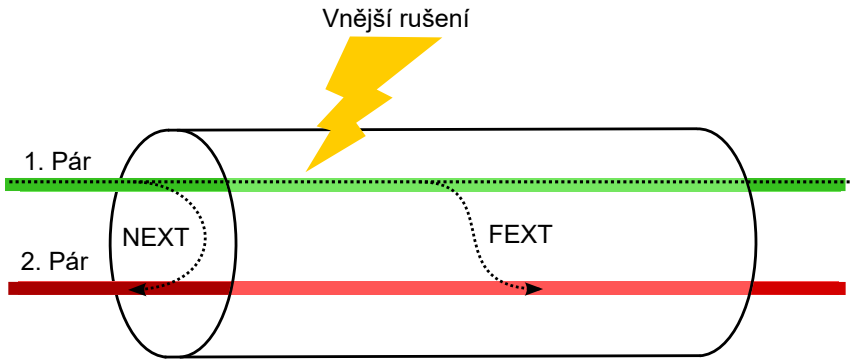
III – G.991.1 1160 kbaud systems [6];

IV – G.993.1 Annex B (plan 997) [4];

V – G.993.2 profile 30a [5];

2 RUŠENÍ V DSL

Přenosové vlastnosti DSL systémů v praxi nedosahují teoreticky dosažitelných parametrů, to je dáno zejména parametry použitého metalického vedení a vnějšími a vnitřními rušivými vlivy. Na signál působí útlum, který je daný délkou vedení, dále může vedení obsahovat nezakončené paralelní odbočky, vložené pupinační cívky (v ČR nebyly u účastnického vedení používané), rozdělení přidružených zkroucených párů ve vedení a úseky s rozdílným průřezem párů, nebo nekrouceným vedením. Mezi vnější vlivy patří impulsní rušení IN (Impulsive Noise) a vysokofrekvenční rušení RFI (Radio Frequency Interference). Vnitřní vlivy jsou přeslechy, které jsou způsobený vazbami mezi sousedními páry ve vícežilových kabelech, a aditivní bílý Gaussův šum AWGN (Additive White Gaussian Noise).



Obr. 2.1: Rušení v xDSL

2.1 Vnitřní rušivé vlivy

2.1.1 Přeslech na blízkem konci – NEXT

Jedním z největších omezení v xDSL přenosových systémech je rušení na blízkém konci NEXT (Near End crossTalk). Rušení je způsobeno indukční a kapacitní vazbou sousedních párů v jednom společném kabelu. Signál, vyslaný modemem do jednoho z párů, vstupuje na stejném konci vedení do druhého páru a způsobuje přeslech v druhém modemu. Tento typ rušení se dále rozděluje na vlastní a cizí, self-NEXT je způsoben stejným typem přenosové technologie, tedy pro přenos je použité stejné přenosové pásmo a úrovně. Cizí přeslech foreign-NEXT je přeslech způsobený jinou technologií, která má rozdílné spektrum a úrovně, např. přeslechy v ADSL způsobené přenosem HDSL na sousedním páru. V ADSL technologii je navíc přenos

pro upstream a downstream spektrálně oddělený, což značně omezuje self-NEXT přeslechy.

Rušení je popsáno jeho výkonovou spektrální hustotou, která je dána vztahem

$$PSD_{\mathrm{NEXT}} = |H_{\mathrm{NEXT}}|^2 \cdot PSD_{\mathrm{PR}}, \qquad [\mathrm{W/Hz}] \quad (2.1)$$

kde $|H_{\mathrm{NEXT}}|^2$ je přenosová funkce přeslechu a PSD_{PR} je výkonová spektrální hustota původce rušení.

Přenosová funkce přeslechu byla empiricky určena na základě simulací a shrnuta v roce 1985 do Ungerova modelu, který používá vedení s 50 páry[1]. Podle tohoto modelu je přenosová funkce určena rovnicí

$$NEXT_{49} = \frac{1}{1{,}134 \cdot 10^{13}} \cdot f^{\frac{3}{2}}. \qquad [-] \quad (2.2)$$

Pro N zdrojů přeslechů platí obecný vztah

$$NEXT_{\mathrm{N}} = \left(\frac{N}{49}\right)^{0.6} \cdot \frac{1}{1{,}134 \cdot 10^{13}} \cdot f^{\frac{3}{2}}. \qquad [-] \quad (2.3)$$

Z výše uvedených vztahů a z principu modelu tohoto přeslechu, je patrné, že úroveň přeslechu na blízkém konci není závislá na délce vedení, ale je dána počtem zdrojů přeslechů, který odpovídá počtu okolních párů v kabelu, dále pak závisí na frekvenci a na výkonové spektrální hustotě původce rušení.

2.1.2 Přeslech na vzdáleném konci – FEXT

Podobně jako NEXT, je zdrojem rušení provoz v okolních párech, v tomto případě je signál, vysílaný modemem z ústředny, přenesen vazbou do sousedních párů a po průchodu vedením je přijat přijímačem na vzdáleném konci. I zde se přeslechy rozdělují na cizí a vlastní (foreign-FETX a self-FEXT). Vzhledem k úrovním přeslechů NEXT je míra přeslechů FEXT často zanedbatelná. Zde už se logicky uplatní v modelu přenosové funkce délka vedení, protože jím přeslech prochází. Obdobně i v níže uvedeném vztahu pro výkonovou spektrální hustotu se objevuje přenosová funkce páru $|H_{\mathrm{P}}|^2$.

$$PSD_{\mathrm{FEXT}} = |H_{\mathrm{FEXT}}|^2 \cdot |H_{\mathrm{P}}|^2 \cdot PSD_{\mathrm{PR}} \qquad [\mathrm{W/Hz}] \quad (2.4)$$

Přenosová funkce pro N zdrojů přeslechů je určena rovnicí

$$FEXT_{\mathrm{N}} = \left(\frac{N}{49}\right)^{0.6} \mathrm{K_{FEXT}} \cdot l \cdot |H_{\mathrm{P}}|^2 \cdot f^2, \qquad [-] \quad (2.5)$$

kde l je délka vedení a $\mathrm{K_{FEXT}}$ je empirická konstanta přeslechu, která je závislá na použitém kabelu. V litetuře[1] je udávána hodnota $\mathrm{K_{FEXT}} = 8 \cdot 10^{-12}\,\mathrm{Hz^{-2} \cdot ft^{-1}}$. Po přepočtu jednotek dostáváme $\mathrm{K_{FEXT}} = 2{,}625 \cdot 10^{-11}\,\mathrm{kHz^{-2} \cdot km^{-1}}$, což odpovídá hodnotě uvedené zde[8].

2.1.3 Aditivní bílý šum – AWGN

Aditivní bílý šum AWGN (Additive white Gaussian noise) je teoretický model náhodného procesu, jehož spektrální výkonová hustota je konstantní, má nulovou střední hodnotu a velikost amplitudy v časové oblasti je dána normálním rozdělením pravděpodobnosti.Je přičítán k užitečnému signálu a prakticky je tvořen několika přírodními jevy, které téměř odpovídají teoretickému modelu bílého šumu. Patří mezi ně zejména:

- **Tepelný šum** – způsobený chaotickým pohybem elektronů ve vodiči.
- **Výstřelkový šum** – způsobený pohybem nosičů náboje v polovodičích.
- **Kvantizační šum** – způsobený rozdílem kvantizační hladiny a analogové úrovně v A/D převodníku.
- **Zbytkový odrazový šum** – způsobený odrazovými interferencemi.

V průběhu standardizace ADSL byla stanovena běžná hodnota úrovně bílého šumu na $-140\,\mathrm{dBm/Hz}$ a tato hodnota je uváděna v ITU-T doporučení, avšak v praxi může dosahovat i vyšších úrovní, zejména při špatném návrhu obvodů, kdy se mohou uplatnit velké úrovně kvantizačního a odrazového šumu. [1, 2, 8]

2.2 Vnější rušivé vlivy

2.2.1 Vysokofrekvenční rušení – RFI

Vysokofrekvenční rušení RFI (Radio-frequency interference) je rušení, které je způsobeno rádiovým provozem v blízkosti vedení. Rádiový provoz je vysílán na krátkých (3–30 MHz) a středních vlnách (520–1 610 kHz). Z těchto hodnot vyplývá, že rušením uvnitř pásma (in-band) budou ovlivněny technologie ADSL vysíláním na středních vlnách a technologie VDSL vysíláním na krátkých vlnách.

2.2.2 Impulzní rušení – IN

Impulzní rušení IN (Impulse noise) je považováno za nejnebezpečnější a je považováno za nejčastější zdroj chyb v datových přenosech. Je charakteristické náhodnými vysokofrekvenčními impulsy o značné amplitudě. [11]

Impulzní rušení lze popsat pomocí úrovně impulsů, spektrálním rozložením, dobou trvání a četností. Zdrojem impulsního rušení jsou přechodové děje v okolních rozvodech elektrické energie, ve kterých dochází ke spínání spotřebičů. Nebo klasické telefonní přístroje a obslužná spojovací zařízení, ve kterých dochází ke spojování účastníků, případně probíhá pulsní volba.[8]

Velikost impulsu se pohybuje běžně v rozmezí 5–20 mV a frekvence impulsů je v rozmezí 1–5 impulsů za minutu. Impulsy dosahují doby trvání 30–150 µs . Tyto

hodnoty závisejí na daných okolních podmínkách a mohou se měnit i s denní dobou. Hustotu pravděpodobnosti pro maximální hodnotu impulzního rušení u můžeme popsat funkcí

$$p(u) = \frac{u_0^2}{u^3}, \tag{2.6}$$

kde $u_0 = 5\,\mathrm{V}$ pro pulzy o velikosti 5–40 mV. Pravděpodobnost, že hodnota impulzu překročí hodnotu u_t lze popsat pomocí

$$P(|u| > u_t) = \left(\frac{u_0}{t}\right)^2. \tag{2.7}$$

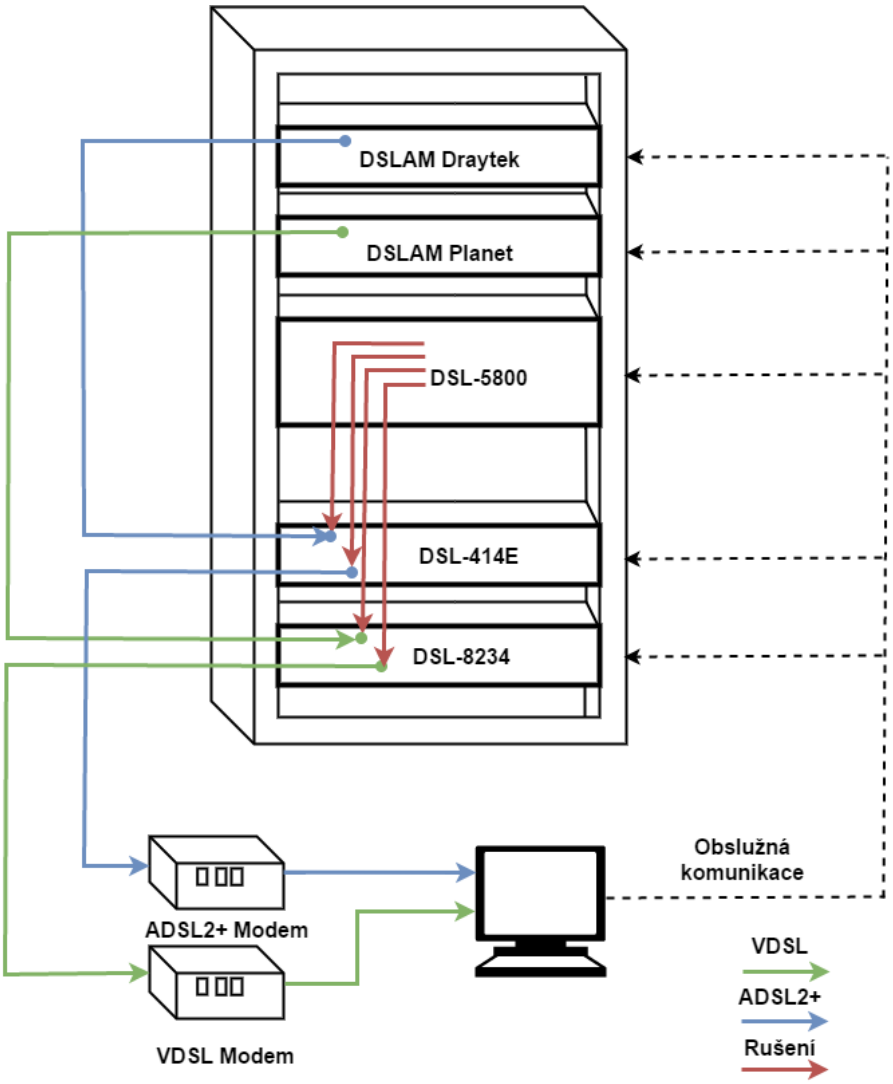
Chybám při přenosu, které jsou způsobeny tímto druhem rušením lze předcházet za použití opravných korekčních kódů FEC (Forward error correction).[1]

3 AUTOMATIZOVANÉ MĚŘENÍ

3.1 Měřicí sestava

V laboratoři je k dispozici k výzkumným účelům s technologiemi DSL měřicí sestava složena z následujících zařízení:

- **Simulátor vedení**
  Simulátor vedení simuluje chování reálného vedení s příslušnou délkou, výpočtem a nastavením parametrů takového vedení mezi vstupy simulátoru, ke kterým jsou připojeny další testovaná zařízení. K dispozici je zařízení DLS 414E podporující technologie ADSL až do verze ADSL2+ a zařízení DLS 8234 podporující technologie VDSL. Komunikace a parametry simulátorů jsou popsány v kapitole 3.1.1.
- **DSLAM**
  Pro generování datového toku a získání přenosových parametrů jsou použity DSLAMy Draytek Vigor 3600 pro ADSL a Planet VC-820M pro VDSL.
- **DSL modem a PC**
  K DSLAMu je připojen DSL modem, který je připojen k PC. PC obsahuje měřicí SW, který obstarává komunikaci se všemi zařízeními.
- **Generátor a injektor rušení**
  K simulaci reálných podmínek na účastnickém vedení slouží Generátor rušení DLS-5800 a injektor DLS-5410DC, komunikace a automatizované měření v závislosti na parametrech rušení je hlavní náplní této práce a jsou ji věnovány následující kapitoly 3.1.3 a 3.2.



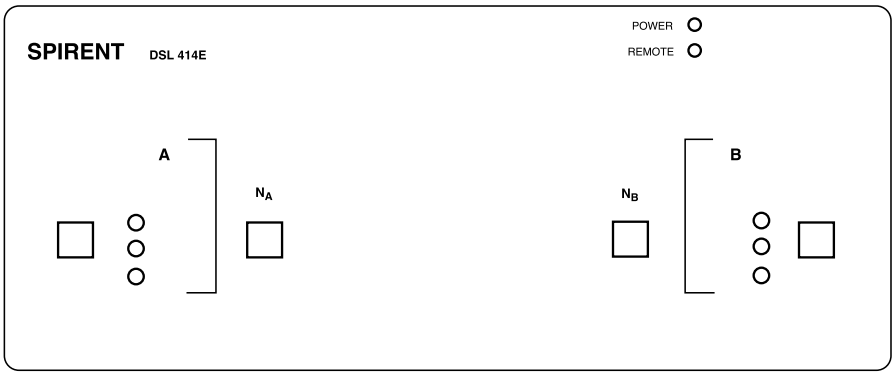
Obr. 3.1: Měřicí sestava

3.1.1 Simulátory vedení

Spirent DLS 414E

Simulátor vedení od společnosti Spirent je zařízení, které pomocí pasivních RLC prvků dokáže simulovat parametry reálného telefonního vedení, konkrétně se jedná o evropský typ vedení PE04, jehož sekundární parametry jsou popsány v doporučení ETSI TS 101 388 [7]. Délka simulovaného vedení může být až 7 km a může být měněna s krokem 25 cm. Jako podporovaná xDSL technologie je uvedeno ADSL2++, avšak tato verze nebyla žádnou normalizační organizací specifikována a jedná se o rozšíření frekvenčního pásma až do frekvence 4,4 MHz oproti 2,2 MHz technologie ADSL2+. Levá strana čelního panelu, označená jako A, typicky slouží k připojení DSLAMu pomocí RJ-45 konektoru nebo 3 pinového CF konektoru, další RJ-45 ko-

nektor, označený NA je k dispozici pro přivedení externího zdroje rušení. Stejné konektory najdeme i na pravé straně, označené jako B, sloužící k připojení ADSL modemu. Na čelním panelu jsou také 2 diody pro indikaci napájení a vzdáleného řízení.



Obr. 3.2: Čelní panel DLS 414E

Na zadní straně zařízení se nacházejí RJ-45 konektory pro připojení dalšího simulátoru vedení, dále konektory IEEE 488 (GPIB) a RS-232 pro připojení počítače ke vzdálenému řízení. Je zde také umístěn konektor a přepínač pro napájení simulátoru.

Základní paramatry simulátoru vedení DLS 414E:

Typ vedení	kroucená dvojlinka PE04
Standard	ITU-T G992.5
Délka vedení	0,50–7000 m s krokem 25 m
Šířka pásma	0–4,5 MHz
Vlastní šum	PSD$\leq$ -150 dBm/Hz
Váha	28 kg
Výška	194 mm
Šířka	452 mm
Hloubka	494 mm

3.1.2 Spirent DLS 8234

Druhý z dvojice simulátorů vedení je DLS 8234 podporující technologii VDSL s frekvenčním pásmem do 30 MHz, taktéž simuluje sekundární parametry vedení PE04 dle doporučení ETSI TS 101 388 [7]. Délka vedení se může měnit v rozmezí 50–3750 m s krokem 5 m. Zvýšení maximální délky simulovaného vedení lze provést pomocí připojení dalšího simulátoru, např. DLS 414E, takto je možné simulovat vedení s délkou až 10770 m. Čelní panel na fotografii 3.3 má obdobné rozložení jako předcházející simulátor, čtveřice RJ-45 konektorů složí k připojení DSLAMu (Side A), DLS modemu (Side B) a rozšiřujícího simulátoru (Ext A a Ext B). Na zadním panelu se opět nacházejí konektory pro vzdálené řízení IEEE 488 (GPIB) a RS-232 a konektor pro připojení napájení.

Základní parametry simulátoru vedení DLS 8234:

Typ vedení	kroucená dvojlinka PE04
Standard	ITU-T G993.2
Délka vedení	0,50–3700 m s krokem 5 m
Šířka pásma	0–30 MHz
Vlastní šum	PSD≤-150 dBm/Hz
Váha	28 kg
Výška	194 mm
Šířka	452 mm
Hloubka	494 mm

Obr. 3.3: Čelní panel DLS 8234

3.1.3 Generátor a injektor rušení

Pro experimentování s vlivy rušení na přenos pomocí technologií xDSL je v laboratoři k dispozici generátor rušení Spirent DLS-5800, který dokáže generovat různé

typy rušení v pásmu do 30 MHz. Toto rušení je přiváděno na vedení pomocí injektoru Spirent DLS-5410DC. Zařízení DLS-5800 je v podstatě PC v šasi, kterou lze umístit do racku. Tento počítač je doplněný o volitelný počet modulů generátoru šumu. Každý z modulů obsahuje 4 BNC konektory pro výstup kanálu, v zařízení je možné mít 2, 4 nebo 6 modulů, čemuž odpovídá až 24 nezávislých kanálů. Na výstupu každého z kanálů je použit 14 bitový AD převodník. Simulátor je dodáván s instalací WinXP Professional, která obsahuje DLS 1100 Software, DLS-5800 Control Software a knihovnu souborů definující parametry jednotlivých druhů rušení. Injektor rušení DLS-5410DC je zařízení o výšce 1 U umístitelné do racku, na zadní straně je umístěno 8 BNC vstupů pro přivedení rušení z generátoru, 4 kanály pro stranu A a 4 pro stranu B. Na přední straně jsou dvojice konektorů RJ-45 pro každou stranu, konektor označený DUT slouží pro připojení DSLAMu na straně A nebo xDSL modemu na straně B, konektory označené Wireline slouží k připojení simulátoru vedení (případně k reálnému vedení). Injektor také může generovat některé druhy impulsního rušení nebo mikropřerušení.

Základní parametry generátoru DLS-5800:

Výstupní úroveň	± 10 V, ± 5 V, do zátěže 50 Ω
Maximální výkon	+13 dBm, do zátěže 50 Ω
Vlastní šum	≤-150 dBm/Hz
Výstupní impedance	50 Ω
Maximální odchylka PSD	0,5 dB
Dinamický rozsah	100 dB
Váha	19 kg
Výška	483 mm
Šířka	545 mm
Hloubka	177 mm

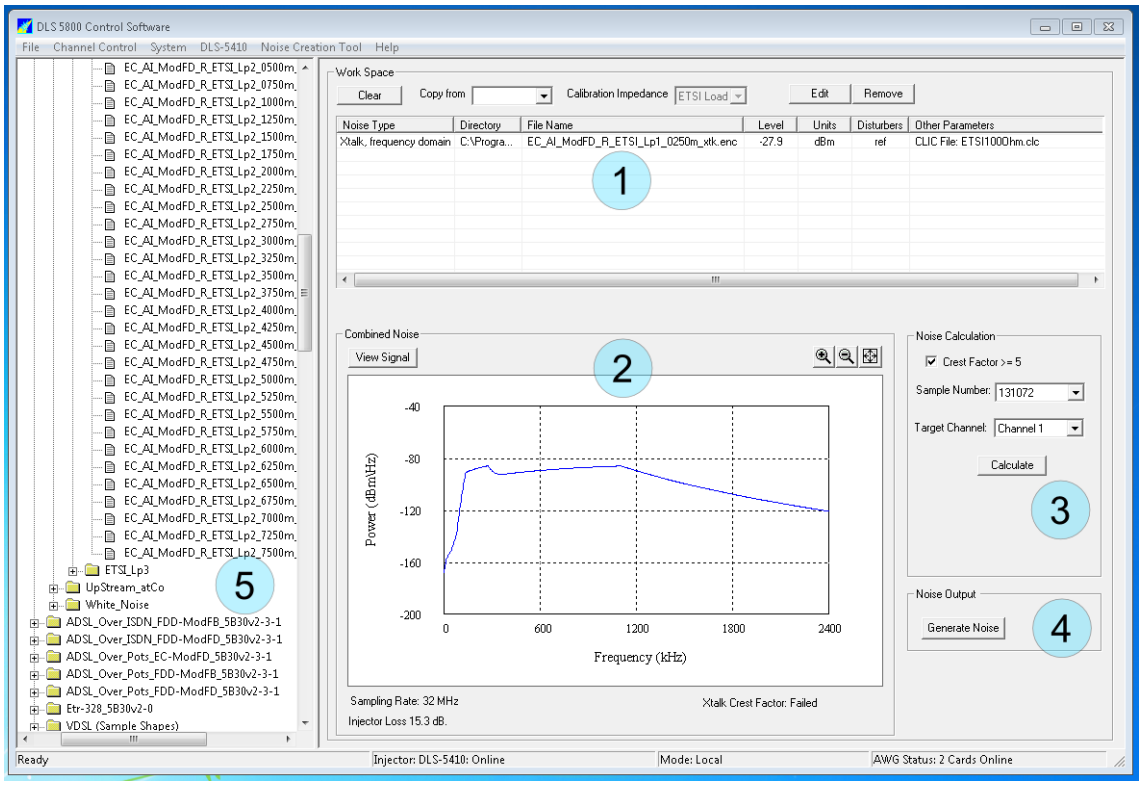
Základní parametry injektoru DLS 5410DC:

Vlastní šum	PSD≤-150 dBm/Hz
Výstupní impedance	150 Ω,diferenční režim 4 kΩ
Váha	4,8 kg
Výška	483 mm
Šířka	483 mm
Hloubka	45 mm

DLS-5800 Control Software

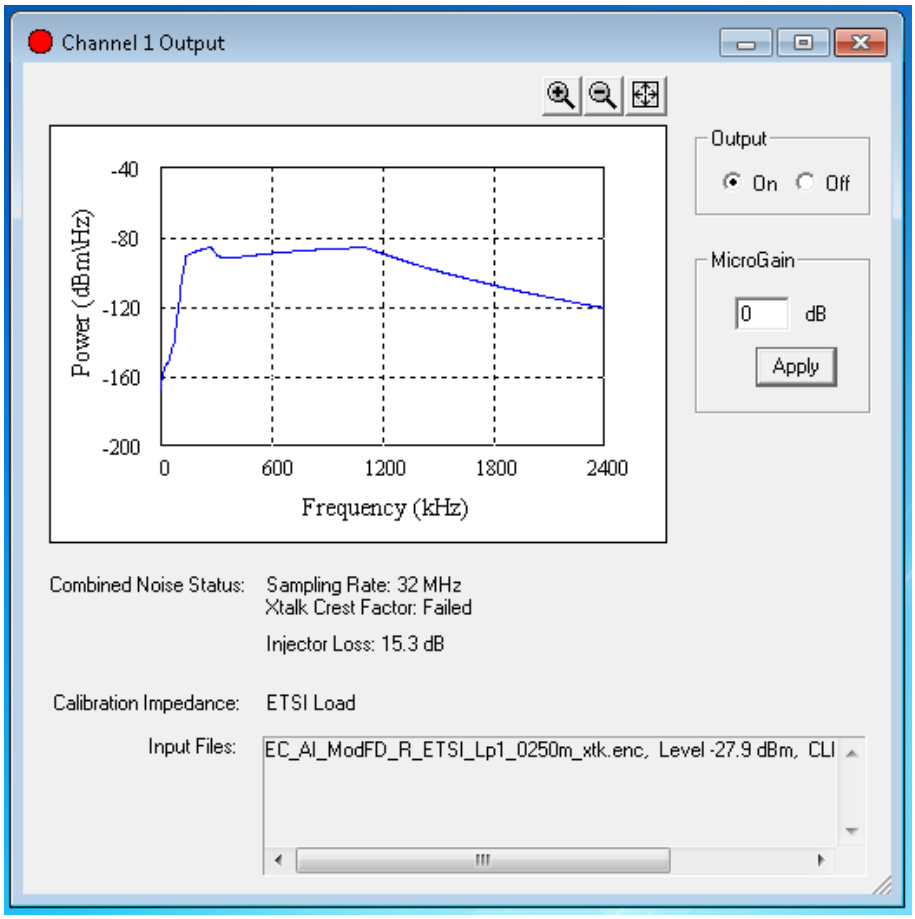
Tento program dodávaný výrobcem je nainstalován v generátoru a slouží k výběru rušení a nastavení jeho parametrů. Lze kombinovat několik druhů rušení a zvolené kombinaci vybrat příslušný kanál generátoru a injektoru. Po spuštění programu je zobrazeno hlavní okno, viz obrázek 3.4 obsahující tyto ovládací komponenty rozdělené do těchto částí:

1. **Work Space**
   Tato část slouží ke kombinování profilů rušení, lze kombinovat až 7 různých profilů typu XTK a RFI. Rušení typu TD a IMP lze kombinovat pouze v případě stejné vzorkovací frekvence, jinak je možné nahrát na kanál jen jeden profil. Každému profilu odpovídá jeden řádek tabulky, který obsahuje informace o typu rušení, cestu k adresáři a název souboru profilu, úroveň a jednotku rušení, v některých případech počet zdrojů rušení a další specifické parametry. Tlačítkem *Clear* se vymažou všechny řádky tabulky, pomocí rozbalovací nabídky *Copy From* lze zvolit číslo kanálu, ze kterého se zkopírují profily rušení do *Work Space*. Další rozbalovací nabídka složí k volbě kalibrační impedance a lze volit mezi 100 Ω,135 Ω, ETSI (komplexní impedance pro určité typy profilů). Tlačítko *Edit* slouží k nastavení parametrů rušení, nejčastěji se jedná o jeho úroveň. Poslední tlačítko *Remove* slouží k odstranění profilu z *Work Space*.
2. **Combined Noise**
   Jedná se o zobrazení charakteristiky zvolených profilů rušení.
3. **Noise Calculation**
   Zde jsou umístěny volby s nastavením výpočtu vzorků rušení podle daných profilů rušení. Zaškrtnutím nabídky *Crest Factor* ≥ 5 bude algoritmus kontrolovat zda činitel výkyvu překročil hodnotu 5, pokud se po 5 iteracích nepodaří dosáhnout vyšší hodnoty, výpočet se ukončí a zobrazí se „Failed“. Tento požadavek je součástí některých ETSI specifikací.Jako další se zde nacházejí rozbalovací nabídky *Sample Number* a *Target Channel* sloužící k výběru počtu vzorků a cílového výstupního kanálu. Tlačítkem *Calculate* se přepočítají vzorky podle zvolených profilů a jejich parametrů.
4. **Noise Output**
   Kliknutím na tlačítko *Generate Noise* se aktivuje vybraný kanál generátoru a na výstup se odesílají vypočtené vzorky rušení.
5. **Stromová adresářová struktura**
   Slouží k výběru profilů, které jsou organizovány do složek podle typu organizace, typu rušení, typu technologie, frekvenčního rozsahu apod.

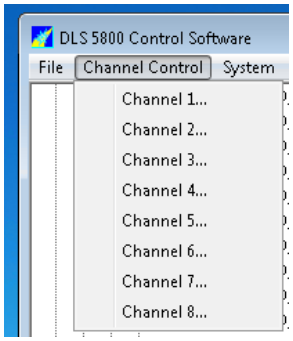


Obr. 3.4: Hlavní okno programu DLS-5800 Control Software

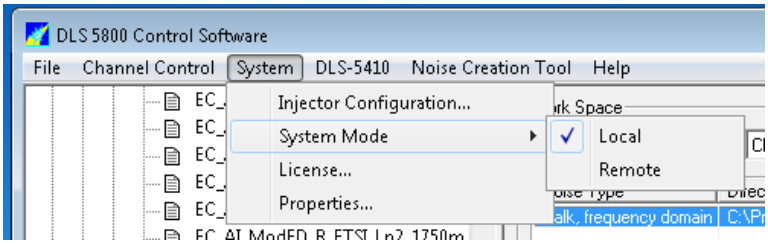
Pokud je vygenerované rušení přivedeno na výstup daného kanálu, je k dispozici okno *Channel Output* viz obr. 3.5 (a), toto okno je možné pro každý kanál vyvolat z nabídky *Channel Control*. V okně je graficky zobrazena výkonová spektrální hustota výstupního rušení, informace o nakombinovaných profilech rušení, kalibrační impedanci a vzorkovací frekvenci. V tomto okně lze pomocí *MicroGain* změnit úroveň výstupního rušení bez nutnosti přepočítat vzorky rušení, úroveň je možné měnit v rozmezí -3–9 dB pro profily rušení ve frekvenční oblasti, případně -3–7 dB pro profily definované v časové oblasti. V tomto okně je možné generovaní rušení na výstupu zastavit a opět spustit pomocí tlačítek *On* a *Off*. Nastavení pro aktivovaní vzdáleného řízení je v nabídce *System* obr. 3.5 (c), po aktivování lze generátor vzdáleně řídit pomocí příkazů popsaných v následující části 3.1.3.



(a) Output Channel



(b) Channel Control

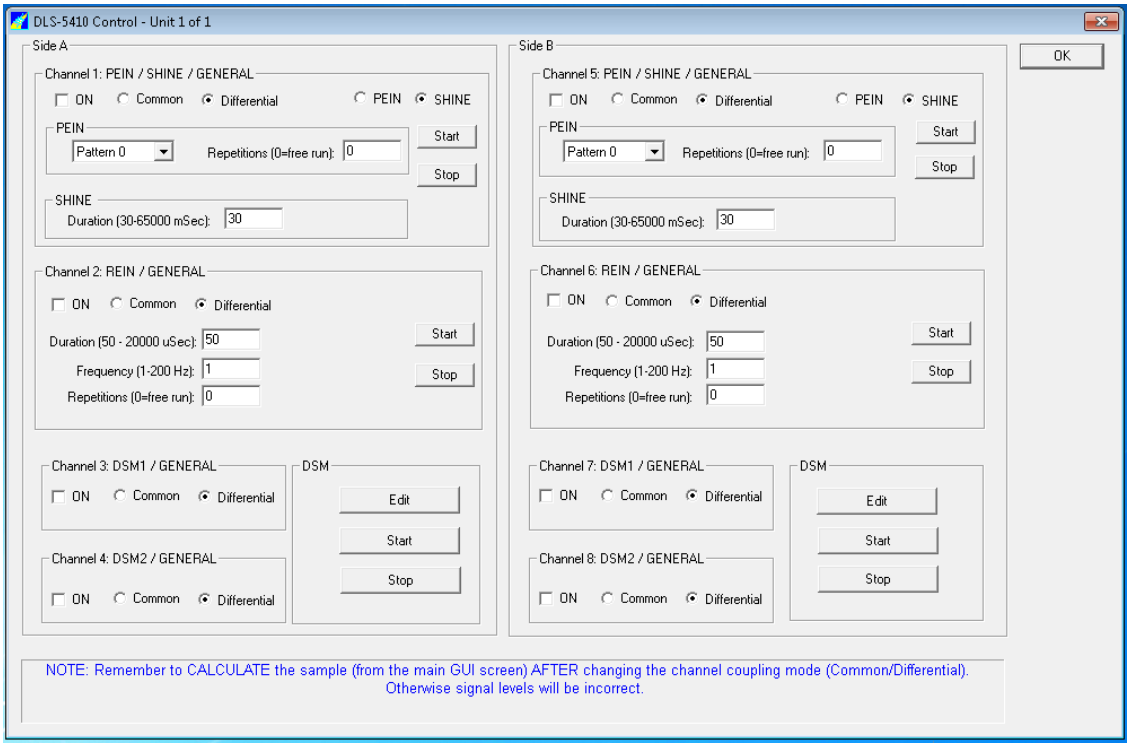


(c) Nabídka System

Obr. 3.5: Možnosti DLS-5800 Control Software

DLS-5410 Control Z hlavní nabídky je možné vyvolat okno pro ovládání injektoru rušení DLS-5410, každá z připojených jednotek se ovládá ve zvláštním okně viz obrázek 3.6. Okno je rozděleno na 2 části *Side A* a *Side B*, v každé části je možné ovládat 4 kanály injektoru. Každý kanál lze aktivovat zaškrtnutím tlačítka *ON* a přepnout mezi režimem *Common* a *Differential*. Další možnosti jsou specifické pro

každý z kanálu, první kanál na straně A i B umožňuje přivádět na kanál rušení typu PEIN nebo SHINE, na druhém kanálu je možné nadefinovat rušení typu REIN a 3. a 4. kanál umožňuje nadefinovat uživatelské DMS sekvence.



Obr. 3.6: Okno pro řízení injektoru

Vzdálené řízení

Generátor rušení je možné přepnout do režimu pro Vzdálené řízení (remote control) a následně jej ovládat pomocí textových zpráv, které jsou do zařízení zasílány prostřednictvím protokolu Telnet. Podoba textových příkazů je popsána v manuálu přístroje, každý příkaz má na počátku řetězec `!STX`, následuje tělo zprávy a ukončující řetězec `ETX!`. Na každou zprávu zařízení odpovídá potvrzením správnosti formátu zprávy, následně je proveden příkaz a odešle se informace o úspěšném či neúspěšném provedení příkazu.

Zprávy se dělí na 4 typy:

- `SET(parameter ID)`
- `SET(parameter ID):VAL(value)`
- `GET(parameter ID)`
- `TRAP(parameter ID):VAL(value)`

příkladem zprávy může být:

```
!STX:GET(M_SELECTED_OUTPUT);ETX!
```

nebo

```
!STX:SET(M_ENABLE_OUTPUT):VAL(OUTPUT_1:ON);ETX!
```

Aplikace pro vzdálené řízení se připojí k zařízení, odesílá požadované příkazy a čeká na odpovědi od zařízení. V manuálu [13] je popsáno typické pořadí zpráv pro správnou funkci generování na daném kanálu:

- Nahrání profilu rušení
- Nastavení všech nezbytných parametrů
- Vygenerování vzorku rušení
- Načtení a aktivování požadovaných výstupu
- Aktivování požadovaných kanálů injektoru

K dispozici jsou zprávy určené k získání a nastavení systémových a síťových informací, další zprávy jsou určené k nastavování parametrů výstupu daného kanálu, nastavení jednotky injektoru, nastavení synchronizace měření atd. Podrobný popis použitých příkazů a způsob jejich generování je uveden v dokumentaci měřicího SW v části A.2.1.

Druhy rušení a jejich profily

Součástí DLS-5800 Contol Software je obsáhla knihovna obsahující soubory profilů, ve kterých jsou uloženy parametry generovaného rušení, které jsou zpravidla specifikovány v některém z doporučení. Soubory jsou umístěny v adresářích podle organizace, která dané doporučení stanovila. Přeslechy a šum, které jsou definované pomocí spektrální výkonové hustoty, jsou ukládány do souborů s koncovkou `xtk.dat`, v tomto souboru je na každém řádku uvedena hodnota frekvence a příslušná úroveň rušení, výsledný průběh spektrální hustoty je dán lineární interpolací těchto hodnot. Tento soubor může obsahovat hodnotu s referenční zátěží, která je umístěna na řádku se zápornou frekvencí.

Profily definující RFI rušení mají koncovku `rfi.dat`, toto rušení je popsáno pomocí amplitudové modulace, soubor definuje parametry amplitudy, kmitočtu a fáze nosné, hloubku modulace a velikost normalizovaného modulovaného šumu. Pomocí těchto parametrů je generováno rušení odpovídající amplitudové modulaci s dvěma postranními pásmy.

Poslední možností je definovat rušení v časové oblasti, tyto soubory mají koncovku `td.dat`. Soubor obsahuje hodnotu vzorkovací frekvence a sekvenci dat s velikostmi daných vzorků, které jsou periodicky generovány na výstup generátoru.

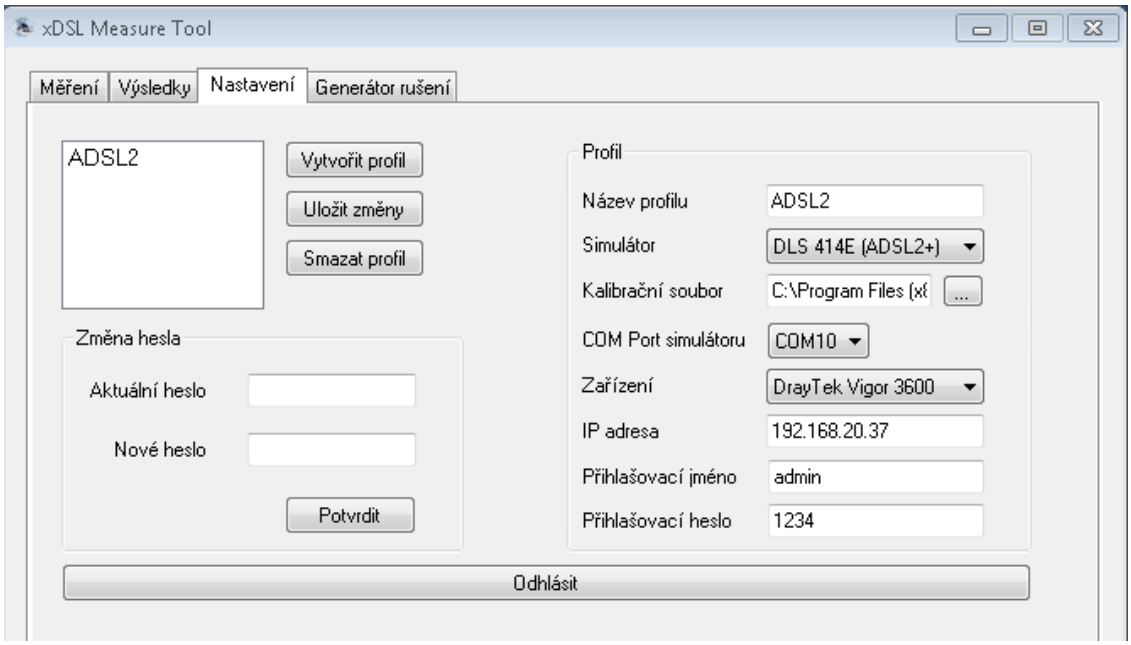
3.2 Měřicí SW

Podle zadání byla vytvořena aplikace umožňující plně automatizované měření přenosové rychlosti ADSL a VDSL modemů na délce vedení a úrovni daného typu rušení. Jako nejvhodnější byla zvolena varianta, kdy bude využito stávající aplikace xDSL measurement tool, kerá byla vytvořena jako součást bakalářské práce [12]. Tento program umožňuje automatizované měření přenosových vlastností pomocí simulátorů vedení Spirent E414 a 8234, přenosové vlastnosti jsou získávány z DSLAMu Draytek Vigor 3600 pro ADSL a Planet VC-820M pro VDSL. Rozšířením tohoto programu o možnosti automatizovaného měření s využitím generátoru DLS-5800 je možné měřit přenosové vlastnosti v závislosti na délce vedení, druhu rušení a úrovni daného rušení. Byl kladen důraz na zachování vlastností původního SW a uživatelské přívětivosti oproti originálnímu SW výrobce jehož vlastnosti byli uvedeny v 3.1.3.

V následující části jsou stručně popsány možnosti programu xDSL Measure tool, v další části je pak popsán samotný návrh a realizace výsledného měřicího programu. Podrobnější informace o ovládání a používání programu jsou dostupné v nápovědě, která je přiložena k této práci A.1. Zdrojové kódy a podrobné informace o návrhu SW jsou uvedeny v přiložené dokumentaci A.2.

3.2.1 xDSL Measure tool

Po spuštění aplikace je nutné v záložce měření zvolit některý z profilů, který lze nadefinovat v záložce nastavení. V nastavení profilu 3.7 se zvolí název profilu, vybere se typ simulátoru vedení, načte se příslušný kalibrační soubor a zvolí se COM port simulátoru, na kterém probíhá komunikace. Následně lze vybrat mezi DSLAMem Draytek Vigor 3600 nebo Planet VC-820M a nastavit jeho IP adresu a vyplnit přihlašovací údaje k DSLAMu.



Obr. 3.7: Nastavení profilu měření xDSL Measure tool

Takto nadefinovaný profil lze zvolit v záložce měření, ve které je možné provést kontrolu aktivních portů DSLAMu, zvolit některý z aktivních portů a nastavit parametry měření. Mezi parametry patří maximální a minimální délka simulovaného vedení a krok, se kterým se bude měnit, nebo zvolit načtení parametrů z profilu rušení a měřit dle jednotlivých sekvencí nadefinovaných v aktuálním profilu. Dále je možné vybrat mezi měřením parametrů pro downstream a downstream+upstream, případně zvolit opakované měření a nastavit počet opakovaní. Pomocí tlačítka *Spustit měření* se spustí měření, jehož průběh je možné sledovat v nejnižší části okna. Zde jsou vypisovány jednotlivé příkazy, kterými je řízen simulátor vedení a DSLAM.

Po dokončení, ale i v průběhu měření, je možné zobrazit výsledky v záložce *Výsledky*, zde je umístěný seznam dokončených měření s názvem profilu a datem měření, ve vedlejší části je zobrazena tabulka výsledků měření. Ve spodní části jsou výsledky graficky znázorněny a je možné přepnout zobrazovaný parametr, výsledky lze exportovat do souboru a otevřít pomocí MS Excel nebo Matlabu. Podrobný popis ovládacích prvků a úkonů je uveden v nápovědě, která je přiložena k této práci, nastavení programu naleznete v částí A.4.

3.2.2 Automatizované měření s využitím generátoru a injektoru

Při návrhu a realizaci programu byly postupně vytvářeny a testovány části, zajišťující stěžejní funkce programu, kterými jsou zejména:

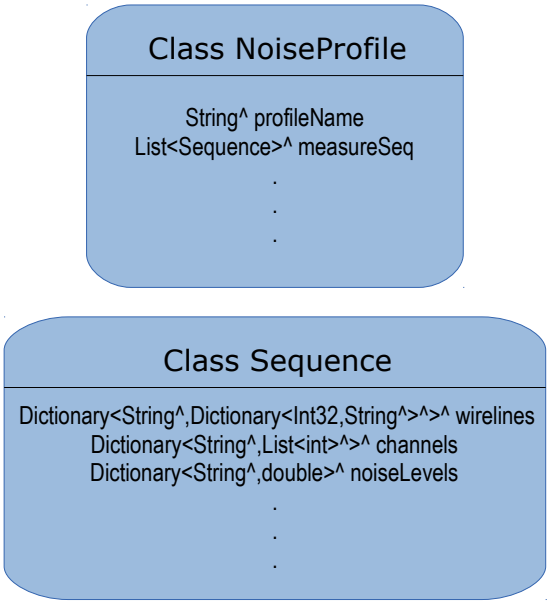
- Komunikace s generátorem pomocí protokolu Telnet.
- Možnosti pro nadefinování parametrů měření,volba typu a úrovně rušení.
- Průběh vlastního měření, korektní změny všech parametrů a zaznamenání naměřených dat.
- Zobrazení a export získaných výsledků.

Komunikace s generátorem

Zahájení komunikace a definice parametrů pro měření je pro přehlednost umístěno ve zvláštní záložce *Generátor rušení*. Jako první je nutné navázat spojení s generátorem rušení, k tomuto účelu byly vytvořeny metody, které využívají knihovnu *Winsock*. V této záložce je možné zadat IP adresu generátoru rušení a port pro komunikaci, pomocí tlačítka *Připojit*, se naváže spojení s generátorem a program může odesílat řídicí zprávy. K tomu je však nutné přepnout generátor na vzdálené řízení v obslužném programu generátoru rušení. Podle návodu ke generátoru[13] byla pro každou ze zpráv definovaná funkce, která má argumenty s požadovanými parametry (např.: zvolený soubor, úroveň rušení, číslo výstupu generátor) a vrací textový řetězec se zprávou pro generátor v přesně definovaném formátu. Tento textový řetězec je předáván funkci, která odesílá zprávu generátoru a přijímá a vrací odpověď generátoru. Zprávy i odpovědi je možné kontrolovat ve výpisu.

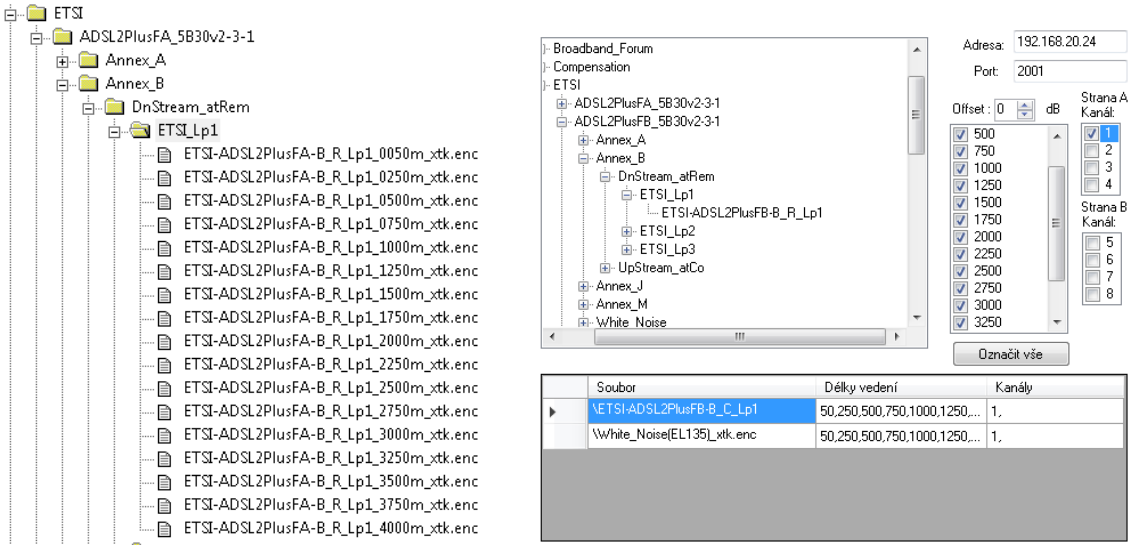
Profily rušení

Vzhledem k tomu, že součástí DLS-5800 je obsáhlá knihovna souborů, které obsahují parametry rušení rozděleny dle typu rušení a příslušného doporučení, byla navržena struktura profilu rušení. Tento profil je vytvořen v programu, uložen v rámci xDSL Measure tool a je instancí třídy *NoiseProfile*. Třída mimo jiné obsahuje proměnou s názvem profilu a seznam sekvencí, kde každá z položek seznamu odpovídá instanci třídy *Sequence*. Instance třídy třídy *Sequence* slouží k uložení informací o typu rušení, které budou působit na vedení současně a uchování seznamu požadovaných délek vedení. Struktura navržených tříd je znázorněna na obr. 3.8.



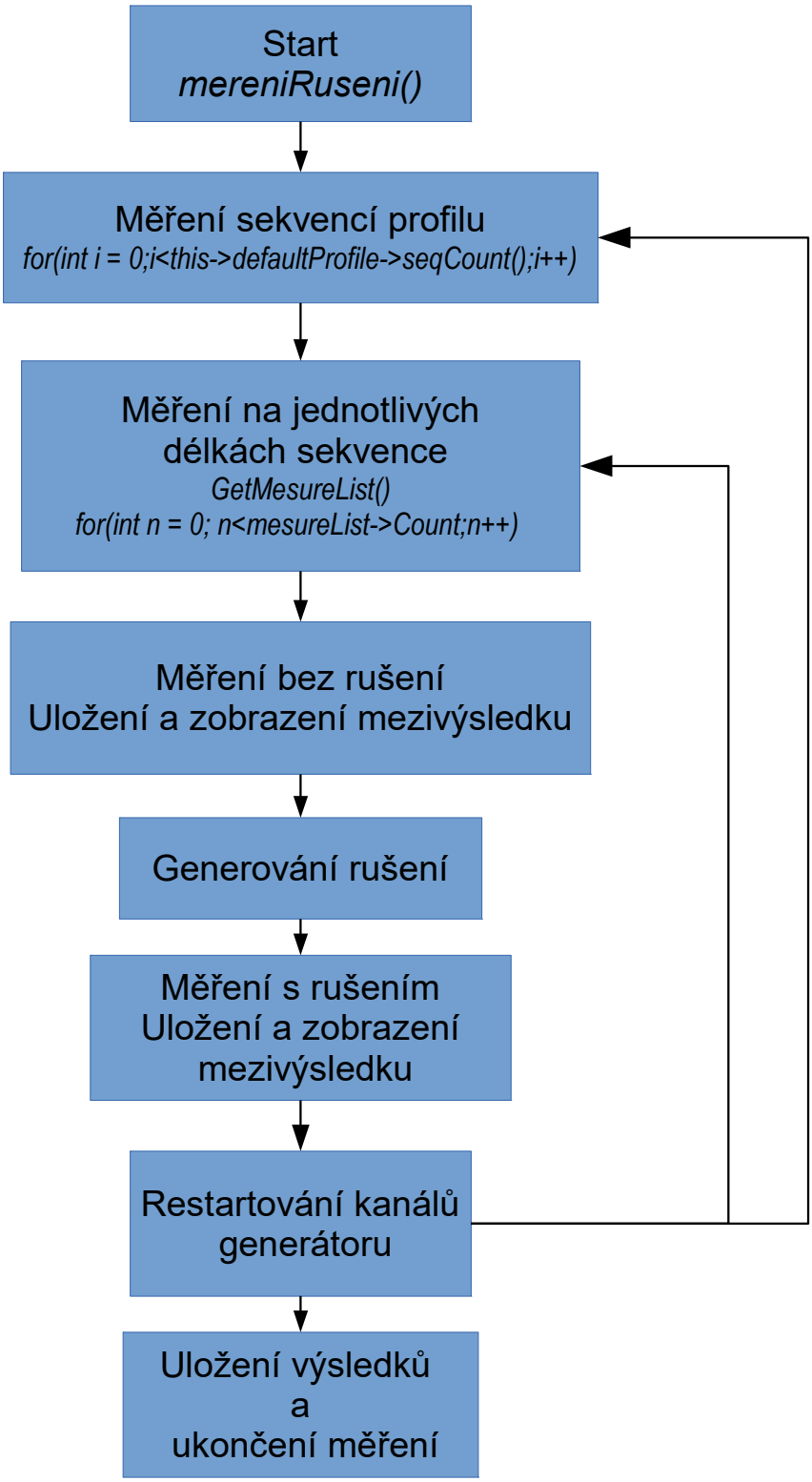
Obr. 3.8: Návrh třídy NoiseProfile a Sequence

Takto složitou strukturu bylo nutné volit zejména proto, že u většiny typů rušení je přenosová funkce závislá na délce daného vedení. Z tohoto důvodu obsahuje knihovna pro každý typ rušení soubory, které jsou určené pro definovanou délku vedení. Příkladem může být zvolení přeslechů pro ADSL2+, jak vidíme na obr. 3.9a. V instanci *Sequence* je uložena informace bez údaje o vzdálenosti a při měření je načítán na výstup generátoru soubor korespondující s nastavenou délkou vedení simulátoru vedení. Zároveň je zajištěno, aby byly při měření nastavené hodnoty délky vedení simulované smyčky, definované pro jednotlivé sekvence.



(a) Originální SW Sprirent (b) Zjednodušení výběru

Obr. 3.9: Výběr souborů s přeslechy pro dané vzdálenosti



Obr. 3.10: Diagram měřicího procesu

Vytvoření profilu a sekvencí

Definice profilu se provádí v záložce *Generátor rušení*, zde se ze stromové adresářové struktury zvolí druhy rušení podle typu technologie. Adresářová struktura odpovídá struktuře v obslužném SW generátoru, výběr je pro uživatele výrazně zjednodušen tím, že nejsou vypisovány všechny soubory pro jednotlivé délky, ale poslední list stromu tvoří pouze přesné označení typu rušení a výběr požadovaných délek pro měření je proveden ve vedlejším panelu. Usnadnění práce uživatele je patrné při porovnání obrázku 3.9a a 3.9b, na kterém je vidět část záložky *Generátor rušení* pro definování parametrů měření.

Kombinacemi různých druhů rušení a jejich úrovní jsou vytvářeny jednotlivé sekvence výsledného profilu, které se ukládají do výše zmíněných tříd. Takto vytvořený profil je možné uložit, případně načíst již uložený profil a přejít k vlastnímu měření.

Bohužel do profilu je možné zahrnou pouze přeslechová rušení a bílý šum, které jsou nejběžnějšími rušivými vlivy a jsou nejvíce zastoupeny v knihovně se soubory rušení od výrobce Sprirent, další typy rušení uvedené v části 3.1.3 nebyly do měřicího programu implementovány.

Měřicí proces

Celý měřicí proces probíhá v několika vnořených cyklech, které jsou znázorněny na diagramu 3.10. Po spuštění měření s nadefinovanými parametry a profilem rušení přejde program do smyčky a vybere se první z měřených sekvencí a pro každou sekvenci se postupně proměřují definované délky vedení. Při měření každé délky vedení jednotlivých sekvencí jsou zaznamenány přenosové parametry s nulovou úrovní rušení, zobrazí a uloží se průběžné výsledky a měření proběhne znovu, tentokrát s načtenými soubory rušení a spuštěným generátorem. Na výstupu generátoru jsou po dobu druhého měření načtena zvolená rušení a pomocí injektoru přivedena na vedení. Po změření a zaznamenání výsledků ovlivněných rušením pro danou délku a sekvenci se výstupy generátoru restartují, měřicí program vrátí do smyčky v místě se změnou měřené délky vedení simulátoru a smyčka se opakuje pro novou vzdálenost. Takto probíhá proměření celé sekvence a může se přejít k následující sekvenci, ve které jsou definované jiné kombinace vstupních parametrů rušení.

Výsledky měření

Po dokončení měření jsou zobrazeny výsledky měření, které nalezneme na příslušné kartě. Ty jsou současně uloženy do složky *Mer* umístěné v kořenovém adresáři programu. Výsledné soubory jsou pojmenovány podle profilu měření (nikoliv rušení),

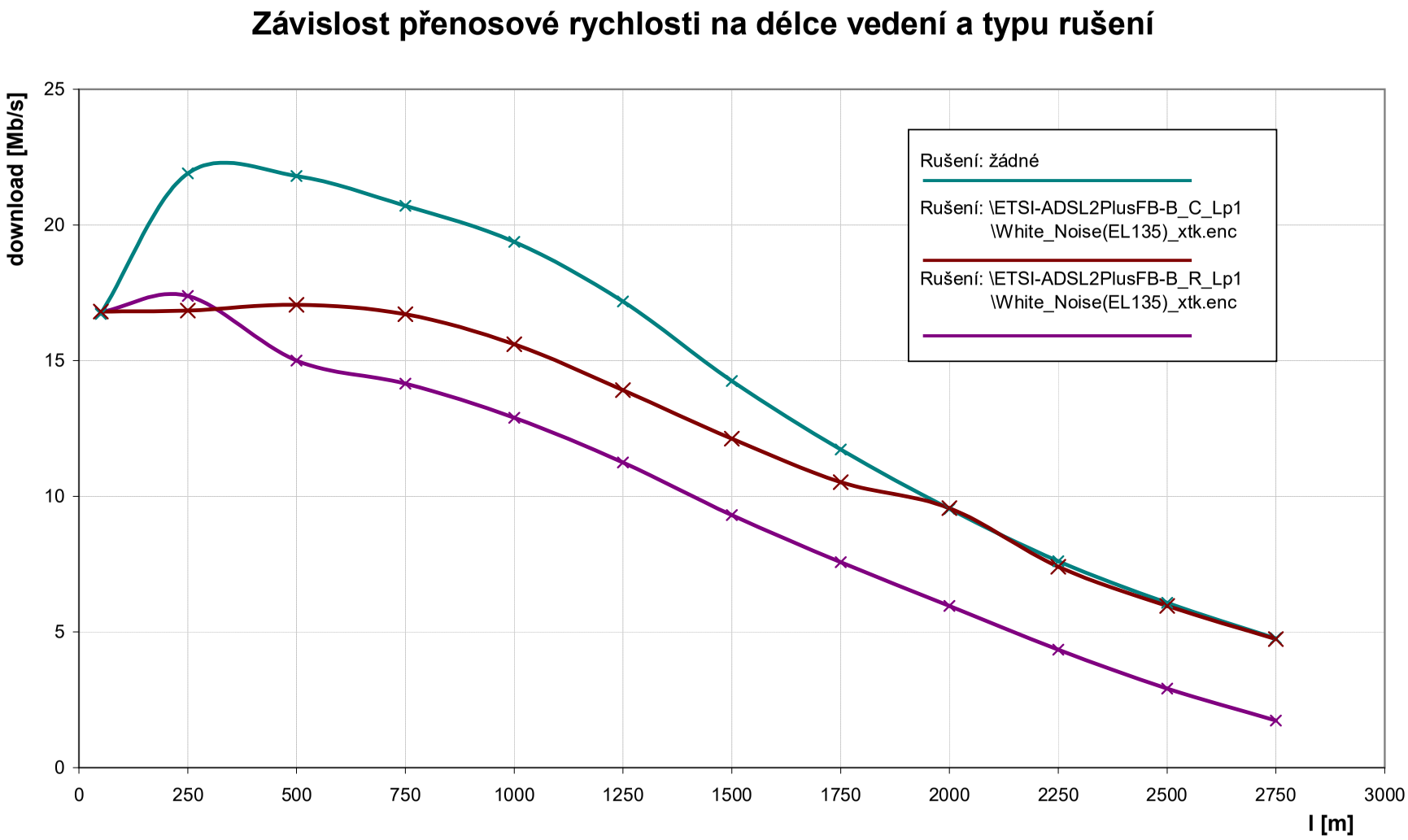
data a času měření, jsou zde zvlášť soubory s výsledky měření pro upstream a downstream, soubory obsahující výsledky bitové alokace DSLAMu, soubor použitého profilu rušení a případně záznam chyb, které mohly vzniknout v důsledku nemožnosti uskutečnění komunikace při vysokém zarušení a dlouhé délky vedení.

V samotném programu jsou výsledky zobrazené v tabulce a graficky. Tabulka obsahuje řádky s výsledky, tak jak byly postupně naměřeny v měřicím cyklu, sekvence po sekvenci, dvě hodnoty pro každou vzdálenost s rušením a bez rušení. Je zde uvedena výsledná přenosová rychlost, útlum, SNR margin a vysílací výkon. V grafu jsou zobrazeny dva průběhy pro každou změřenou sekvenci profilu, jeden průběh odpovídá závislosti naměřené s vlivy rušení a druhý bez rušících vlivů. Je zde zobrazena jednoduchá legenda, ve které je možné zvolit pro větší přehlednost jen některé z průběhů.

3.3 Naměřené výsledky

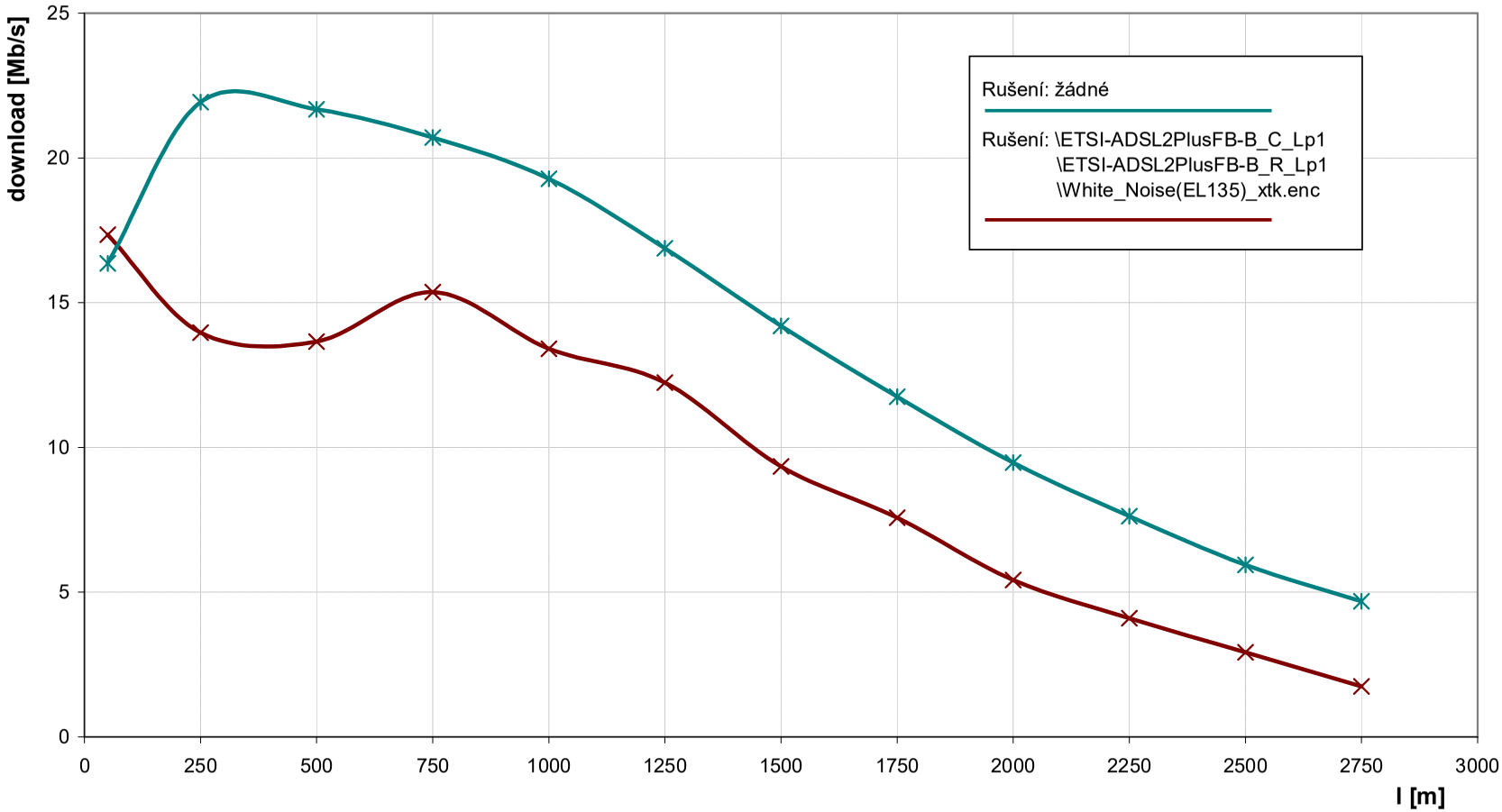
3.3.1 ADSL 2+

Jako příklad naměřených výsledků pomocí měřicího programu byl zvolen test přenosu ADSL 2+. Výsledky jsou pouze pro downstream a jsou rozděleny do dvou částí, první tvoří srovnání přeslechů generovaných na straně DSLAMu a na straně modemu v kombinaci s bílým šumem. V druhé části je přenos ovlivněný přeslechy na obou stranách vedení. V první části si můžeme povšimnout, že přeslechy generované na straně DSLAMu výrazně ovlivňují přenos na kratších vzdálenostech od DSLAMu, pro vzdálenosti od cca. 2 km je rušení i užitečný signál utlumen a celková přenosová rychlost je dána především vzdáleností od DSLAMu. Přeslechy generované na straně modemu ovlivňují přenos výrazněji než na straně DSLAMu a nejsou závislé na délce vedení, protože se po něm nešíří. Pokud přivádíme přeslechy na obě strany vedení, stojí za povšimnutí poklesu přenosové rychlosti při délce vedení 250 a 500 m, takže nemusí platit, že čím blíže DSLAMu tím lepší přenosové v případě zarušeného vedení.


Závislost přenosové rychlosti na délce vedení a typu rušení
download [Mb/s]
25
20
15
10
5
0
0
250
500
750
1000
1250
1500
1750
2000
2250
2500
2750
3000
l [m]
Rušení: žádné
Rušení: \ETSI-ADSL2PlusFB-B_C_Lp1 \White_Noise(EL135)_xtk.enc
Rušení: \ETSI-ADSL2PlusFB-B_R_Lp1 \White_Noise(EL135)_xtk.enc

Tabulka naměřených výsledků pro ADSL2+				
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	vysílací výkon [dB]
50	16,716	0	6,5	16,3
250	21,904	0,5	9	20,3
500	21,804	4	9	20,3
750	20,704	7,5	9	20,3
1000	19,376	12	9,5	20,3
1250	17,184	18	9	20,3
1500	14,248	21	9	20,3
1750	11,724	24,5	7,5	20,3
2000	9,536	27,5	7	20,3
2250	7,608	31	7	20,3
2500	6,072	34	7,5	19,8
2750	4,764	37,5	8	19,3
Rušení:	žádné			
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	vysílací výkon [dB]
50	16,784	0	9	16,3
250	17,384	1	3,5	20,3
500	15	4	6,5	20,3
750	14,148	7,5	5	20,3
1000	12,892	12	6	20,3
1250	11,248	18	5,5	20,3
1500	9,304	21	6,5	20,3
1750	7,568	24,5	6,5	20,3
2000	5,956	27,5	6,5	20,3
2250	4,352	31	7	20,3
2500	2,912	34,5	6,5	19,6
2750	1,74	37,5	7,5	18,7
Rušení:	\ETSI-ADSL2PlusFB-B_R_Lp1 \White_Noise(EL135)_xtk.enc			
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	vysílací výkon [dB]
50	16,804	0	10,5	16,3
250	16,844	0,5	6	20,3
500	17,052	4	20,5	20,3
750	16,708	7,5	6,5	20,3
1000	15,604	12	6	20,3
1250	13,916	18	5,5	20,3
1500	12,128	21	6,5	20,3
1750	10,52	24,5	6,5	20,3
2000	9,564	27,5	6,5	20,3
2250	7,404	31	6,5	20,3
2500	5,96	34	6,5	19,8
2750	4,74	37,5	6	19,3
Rušení:	\ETSI-ADSL2PlusFB-B_C_Lp1 \White_Noise(EL135)_xtk.enc			

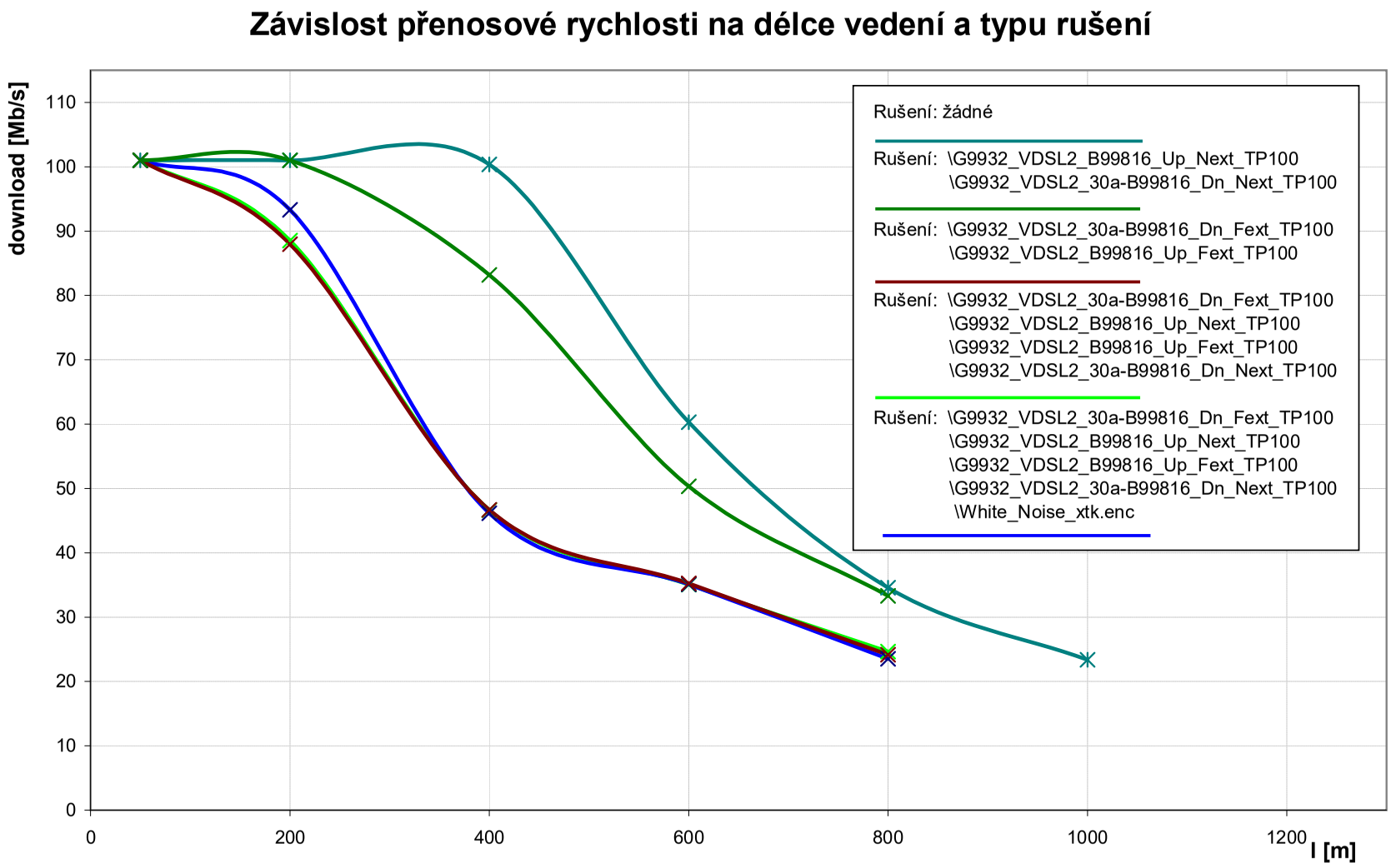
Závislost přenosové rychlosti na délce vedení a typu rušení



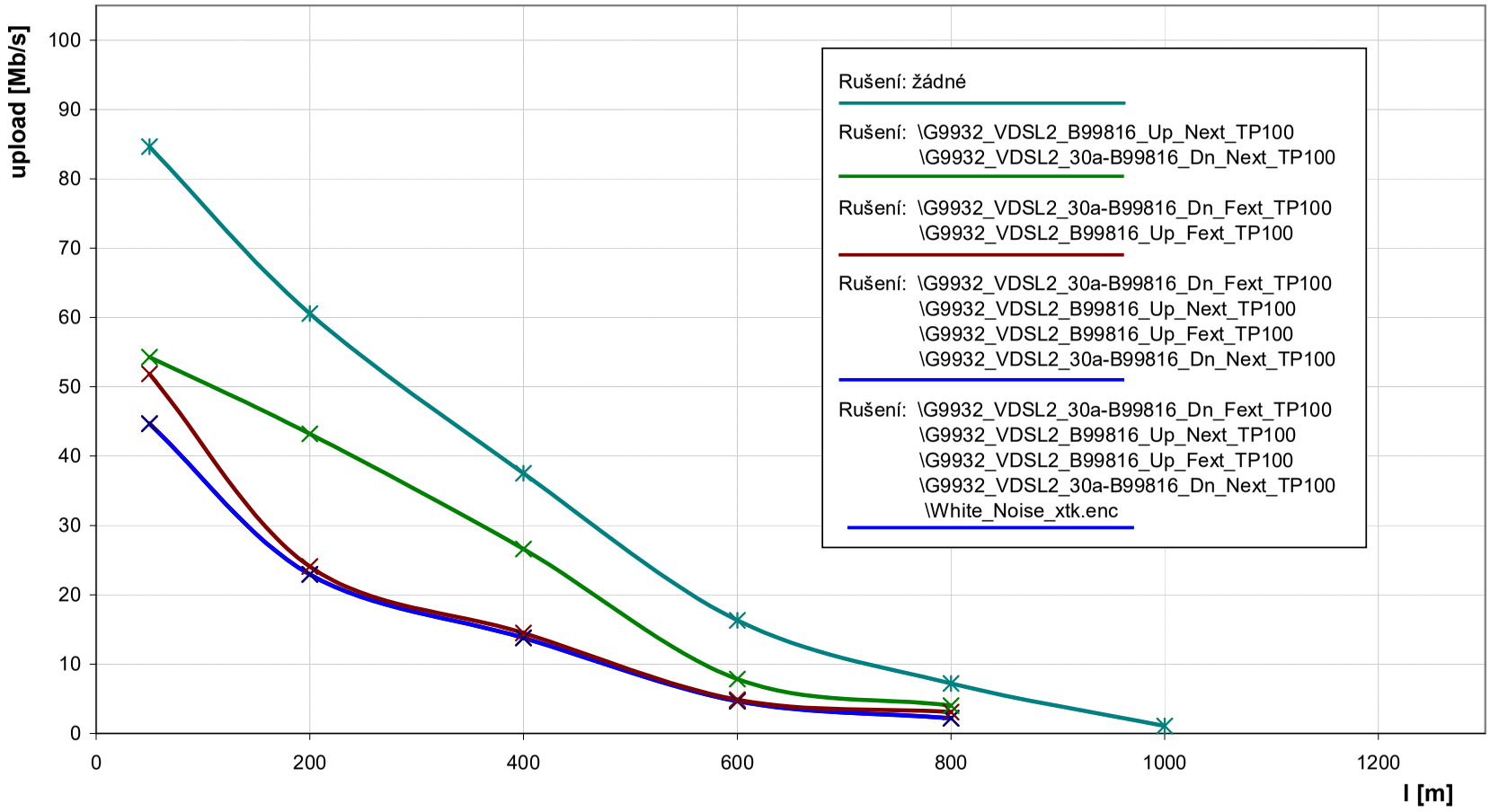
Tabulka naměřených výsledků pro ADSL2+				
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	vysílací výkon [dB]
50	16,356	0	6,5	16,3
250	21,924	0,5	9	20,3
500	21,68	4	9	20,3
750	20,708	7,5	9	20,3
1000	19,28	12	9,5	20,3
1250	16,876	18	9	20,3
1500	14,192	21	8	20,3
1750	11,748	24,5	7,5	20,3
2000	9,48	27,5	7	20,3
2250	7,628	31	7,5	20,3
2500	5,944	34	7,5	19,7
2750	4,684	37,5	7,5	19,3
Rušení:	žádné			
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	vysílací výkon [dB]
50	17,352	0	12,5	16,3
250	13,964	0,5	4	20,3
500	13,66	4	6,5	20,3
750	15,368	7,5	6,5	20,3
1000	13,412	12	6,5	20,3
1250	12,236	18	6,5	20,3
1500	9,34	21	6,5	20,3
1750	7,568	24,5	6,5	20,3
2000	5,42	27,5	6,5	20,3
2250	4,1	31	6,5	20,3
2500	2,924	34	6,5	19,6
2750	1,748	37,5	6	18,7
Rušení:	\ETSI-ADSL2PlusFB-B_R_Lp1 \ETSI-ADSL2PlusFB-B_C_Lp1 \White_Noise(EL135)_xtk.enc			

3.3.2 VDSL 2

Pro technologii VDSL 2 jsou uvedeny naměřené přenosové rychlosti pro dowload i upload. Zde jsou postupně porovnávány jednotlivé typy přeslechového rušení, které jsou vždy přiváděny na obě strany vedení. Pro přenos v downstreamu i upstreamu je zhoršení přenosových vlastností nejméně patrné u přeslechu typu NEXT, protože technologie VDSL dokáže tento druh účinně potlačovat. Při rušení přeslechy typu FEXT jsou přenosové vlastnosti výrazně ovlivněny a rychlost přenosu klesá až o 50 %. V případě kombinací s přeslechem NEXT je výsledná přenosová rychlost daná především rušením FEXT, a proto se výsledné průběhy přenosové rychlosti téměř překrývají.


Závislost přenosové rychlosti na délce vedení a typu rušení
download [Mb/s]
110
100
90
80
70
60
50
40
30
20
10
0
0
200
400
600
800
1000
1200
l [m]
Rušení: žádné
Rušení: \G9932_VDSL2_B99816_Up_Next_TP100
\G9932_VDSL2_30a-B99816_Dn_Next_TP100
Rušení: \G9932_VDSL2_30a-B99816_Dn_Fext_TP100
\G9932_VDSL2_B99816_Up_Fext_TP100
Rušení: \G9932_VDSL2_30a-B99816_Dn_Fext_TP100
\G9932_VDSL2_B99816_Up_Next_TP100
\G9932_VDSL2_B99816_Up_Fext_TP100
\G9932_VDSL2_30a-B99816_Dn_Next_TP100
Rušení: \G9932_VDSL2_30a-B99816_Dn_Fext_TP100
\G9932_VDSL2_B99816_Up_Next_TP100
\G9932_VDSL2_B99816_Up_Fext_TP100
\G9932_VDSL2_30a-B99816_Dn_Next_TP100
\White_Noise_xtk.enc

Závislost přenosové rychlosti na délce vedení a typu rušení



Tabulka naměřených výsledků pro VDSL2						
Download				**Upload**		
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]
50	100,987	0	21	84,653	6	6,3
200	100,987	14	18,2	60,546	18	6,1
400	100,367	24	6,6	37,545	42	6,1
600	60,307	30	7	16,326	68	6,7
800	34,572	38	6,3	7,224	88	6,2
1000	23,359	46	6,2	1,069	118	7,1
Rušení:	žádné					
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]
50	100,987	0	10,3	44,713	8	6,2
200	88,52	14	6,2	22,949	22	6,3
400	46,63	20	6,2	13,78	46	6,3
600	35,051	28	6,2	4,669	68	14,6
800	24,643	36	0	2,174	88	6
Rušení:	\G9932_VDSL2_30a-B99816_Dn_Fext_TP100 \G9932_VDSL2_B99816_Up_Next_TP100 \G9932_VDSL2_B99816_Up_Fext_TP100 \G9932_VDSL2_30a-B99816_Dn_Next_TP100					
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]
50	100,987	0	10,3	44,66	8	6,2
200	88,52	14	6,2	22,934	22	6,3
400	46,63	20	6,2	13,803	46	6,3
600	35,051	28	6,2	4,661	68	6,4
800	24,643	36	0	2,174	88	6,1
Rušení:	\G9932_VDSL2_30a-B99816_Dn_Fext_TP100 \G9932_VDSL2_B99816_Up_Next_TP100 \G9932_VDSL2_B99816_Up_Fext_TP100 \G9932_VDSL2_30a-B99816_Dn_Next_TP100 \White_Noise_xtk.enc					
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]
50	100,987	0	10,3	51,879	8	6,2
200	87,983	14	6,2	24,061	22	6,3
400	46,676	20	6,2	14,474	50	6,3
600	35,238	28	6,2	4,885	64	6,5
800	24,15	36	6,2	3,093	88	6,3
Rušení:	\G9932_VDSL2_30a-B99816_Dn_Fext_TP100 \G9932_VDSL2_B99816_Up_Fext_TP100					
délka [m]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]	rychlost [Mb/s]	útlum [dB]	SNR Margin [dB]
50	100,987	0	12,5	54,296	8	6,2
200	100,987	14	10,5	43,204	18	6,2
400	83,204	24	6,2	26,616	42	6,2
600	50,334	30	6,2	7,858	66	6,5
800	33,311	36	6,3	4,005	88	6,3
Rušení:	\G9932_VDSL2_B99816_Up_Next_TP100 \G9932_VDSL2_30a-B99816_Dn_Next_TP100					

4 ZÁVĚR

V závěru práce je uveden souhrn jednotlivých částí bakalářské práce, je zde krátká diskuze nad dosaženými výsledky této práce a popsány možnosti výsledného měřicího programu. Součástí bakalářské práce, je prostudování teorie xDSL přenosových technologií a především vlivu rušení. Problematika xDSL technologií je stručně popsána v teoretickém úvodu, následuje rozdělení a popis rušivých vlivů a jejich matematické modely. Další části práce jsou věnovány měřicí sestavě, použitým přístrojům a výslednému programu pro automatizované měření. Následuje příklad naměřených výsledků pomocí měřicího SW. V příloze této práce je uvedena nápověda a dokumentace, ve které jsou zdrojové kódy nejpodstatnějších částí programu a jejich popis.

Bakalářská práce navazuje na poznatky získané během semestrální práce, při které bylo rozhodnuto o využití programu xDSL Measure Tool, který byl vypracován jako součást bakalářské práce [12]. Součástí semestrální práce bylo také implementování komunikace s generátorem a návrh profilu a sekvencí pro měření s generátorem rušení. Takto navržené třídy byly zakomponovány do programu pro měření a postupně testovány, při této práci byl kladen důraz především na snadnou obsluhu a přehlednost oproti originálnímu SW výrobce a také na jednoduchou implementaci do měřicího procesu, tak aby mohlo být využito, co nejvíce již vytvořených metod pro vlastní měření. Tento postup sebou přinášel řadu komplikací, program byl tvořen ve vývojovém prostředí Visual Studio 2010 a napsán v programovacím jazyce C++/CLI, který oproti běžnému C++ dovoluje použít a vytvářet objekty, které jsou součástí .NET frameworku. Velkým problémem je převod mezi klasickými prvky C++ a CLI prvky, navíc míchání postupů klasického C++ a objektů .NET vede na velice špatně čitelný kód. Dokumentace tříd .NET navíc často neobsahuje zdrojové kódy pro C++/CLI ale pouze pro novější C#. Visual Studio pro C++/CLI nepodporuje tzv. *InteliSense*, který výrazně napomáhá práci programátora a poskytuje interaktivní nápovědu a automatické dokončení kódu. Bohužel k původnímu programu nebyla k dispozici dokumentace a obsahoval pouze stručný popis pomocí komentářů. I přes všechny uvedené komplikace byla práce na profilu rušení a měřicích sekvencí poměrně brzy dokončena a mohla být implementována do měřicího procesu. Tato práce byla komplikovanější vzhledem absenci dokumentace a složitosti programu měřicího procesu např. funkce `void Form1::vizualizace_dat(int i,int j,int k,int cislo_profilu,int opakovani)`, ve které jsou parametry dále předávány další funkci `zpracuj(...)`. Velmi nepříjemný byl také fakt, že samotné měření trvá velice dlouhou dobu a lze tento proces jen velice obtížně testovat a klasické debugování je prakticky znemožněno. Z tohoto důvodu je výsledný program náchylný k chybám a není zaručen bezproblémový běh programu např. při chybně definované

sekvenci. I přes všechny potíže byl program dokončen a podařilo se změřit řadu výsledků, příklady jsou uvedeny v kapitole 3.3 Naměřené Výsledky.

LITERATURA

[1] CHEN, Walter Y. *DSL: simulation techniques and standards development for digital subscriber line systems.* Indianapolis, IN: Macmillan Technical Pub., c1998, 644 p. ISBN 1578700175.

[2] ITU-T Recommendation G.992.1 *G.992.1: Asymmetric digital subscriber line (ADSL) transceivers* ITU-T, June 1999.

[3] ITU-T Recommendation G.992.5 *G.992.5: Asymmetric digital subscriber line transceivers 2 (ADSL2) – Extended bandwith (ADSL2plus).* ITU-T, January 2009.

[4] ITU-T Recommendation G.993.1 *G.993.1 : Very high speed digital subscriber line transceivers (VDSL)* ITU-T, June 2004.

[5] ITU-T Recommendation G.993.2 *G.993.2: Very high speed digital subscriber line transceivers 2 (VDSL2)* ITU-T, February 2006.

[6] ITU-T Recommendation G.991.1 *G.991.1: High bit rate digital subscriber line (HDSL) transceivers* ITU-T, October 1999.

[7] ETSI TS 101 388 V1.4.1 *Access Terminals Transmission and Multiplexing (ATTM); Access transmission systems on metallic access cables; Asymmetric Digital Subscriber Line (ADSL) - European specific requirements*
ETSI, 2007-08
URL:
`http://www.etsi.org/deliver/etsi_ts/101300_101399/101388/01.04.01_60/`

[8] VODRÁŽKA, J. *Rušivé vlivy působící na vedení xDSL systémů* elektrorevue.cz 9.1.2005
URL: `http://www.elektrorevue.cz/clanky/05003/index.html`

[9] ČERMÁK, J. *Modelování rušení pro xDSL* Diplomová práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008.

[10] SMÉKAL, Z. *Analýza signálů a soustav – BASS* Skripta, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012.

[11] ŠILHAVÝ, P. *Datová komunikace* Skripta, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011.

[12] POKORNÝ, L. *Automatizované měření přenosových vlastností DSL modemů*, Bakalářská práce, VUT v Brně, 2013.

[13] SPIRENT COMMUNICATIONS *User Guide: DLS-5800 xDSL/DSM Custom Noise Generator/Sequencer DLS-5410DC Differential/Common Mode Noise Injection Unit DLS-5409 Mini Passive Noise Injector* ,
URL: `http://ekb.spirent.com/resources/sites/SPIRENT/content/live/SOLUTIONS/10000/SOL10769/en_US/DLS-580020v2.1%20User%20Manual.pdf`

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DSL	digitální účastnická přípojka – Digital Subscriber Line
ADSL	Zkratka ADSL technologie – Asymmetric Digital Subscriber Line
POTS	tradiční analogová telefonní služba – Plain ordinary telephone service
ISDN	Digitální síť integrovaných služeb – Integrated Services Digital Network
FDM	frekvenční oddělení přenosu – Frequency Division Multiplexing
EC	potlačení ozvěny – Echo Cancellation
DMT	Diskrétní vícetónová modulace – Discrete MultiTone
QAM	kvadraturní amplitudová modulace – Quadrature amplitude modulation
FFT	rychlá Fourierova transformace – Fast Fourier Transformation
IFFT	inverzní rychlá Fourierova transformace – Inverse Fast Fourier Transformation
FEC	samoopravné kódování – Forward error coding
SNR	kvadraturní amplitudová modulace – Quadrature amplitude modulation
TEQ	vyvážení v časové oblasti – Time domain equalizer
FEQ	vyvážení ve frekvenční oblasti – Frequency domain equalizer
CRC	cyklický redundantní součet – Cyclic redundancy check
IN	impulsní rušení – Impulsive Noise
RFI	vysokofrekvenční rušení – Radio Frequency Interference
AWGN	aditivní bílý Gaussův šum – Additive White Gaussian Noise
NEXT	přeslech na blízkem konci – Near End crossTalk
PSD	výkonová spektrální hustota – Power Spectral Density
FEXT	přeslech na vzdáleném konci – Far End CrossTalk

A NÁPOVĚDA A DOKUMENTACE

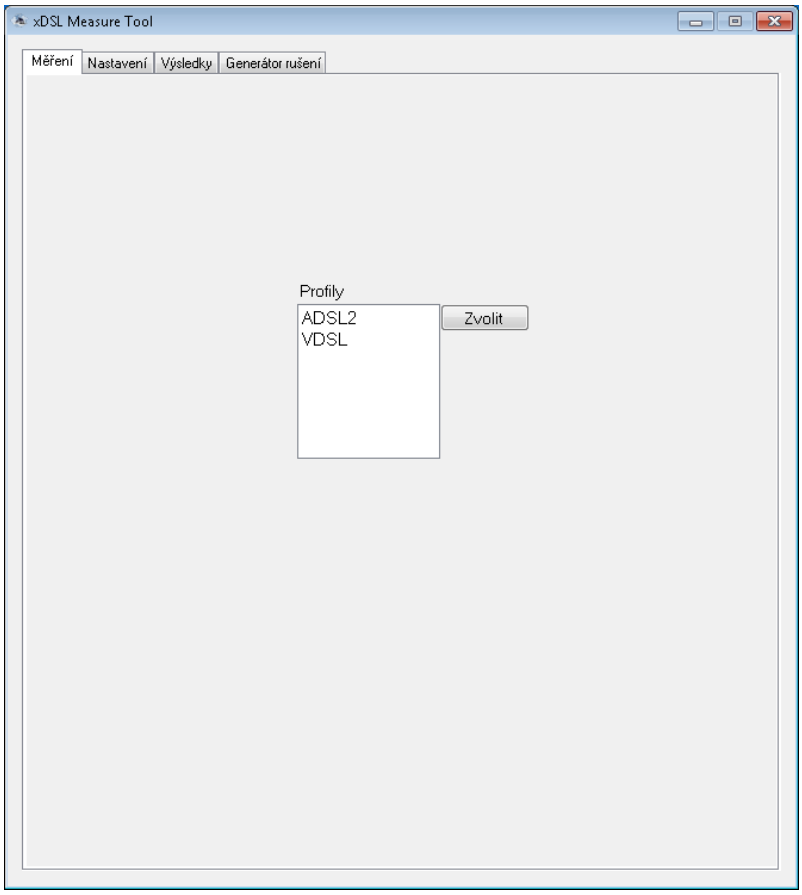
Jelikož obsluha programu není triviální, je zde uvedena stručná nápověda a popsán postup pro měření. Následuje krátká dokumentace, která obsahuje zdrojové kódy a popis zásadních prvků, které byly autorem zakomponovány do měřícího SW *xDSL Measure Tool* [12].

A.1 Nápověda

Měřící SW xDSL Measure Tool obsahuje panel se 4 kartami, 3 původní karty *Měření*, *Nastavení* a *Výsledky* byly upraveny do výsledné podoby pro měření přenosových parametrů v závislosti na rušení, které je generováno pomocí simulátoru rušení Spirent DLS-5800. Parametry rušení je možné nastavit na kartě *Generátor rušení*.

A.1.1 Karta Měření

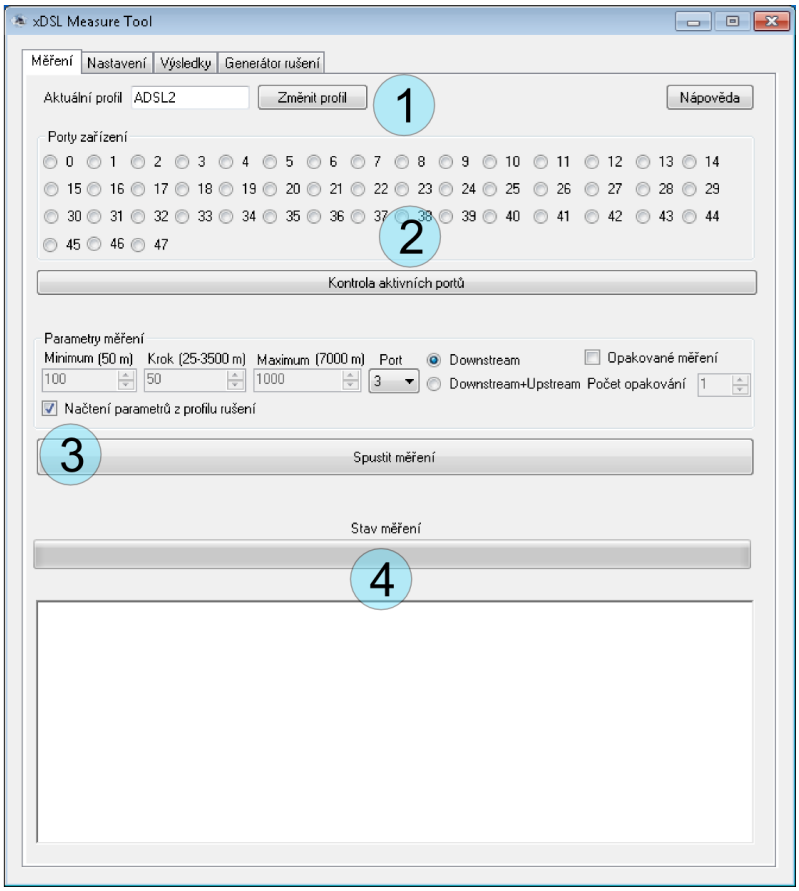
Po spuštění programu je nutné vybrat jeden z profilů měření, které je možné definovat v kartě *Nastavení* viz část A.3.



Obr. A.1: Karta Měření – výběr profilu

Následně je zobrazena karta A.2 s měřením, která obsahuje následující části:

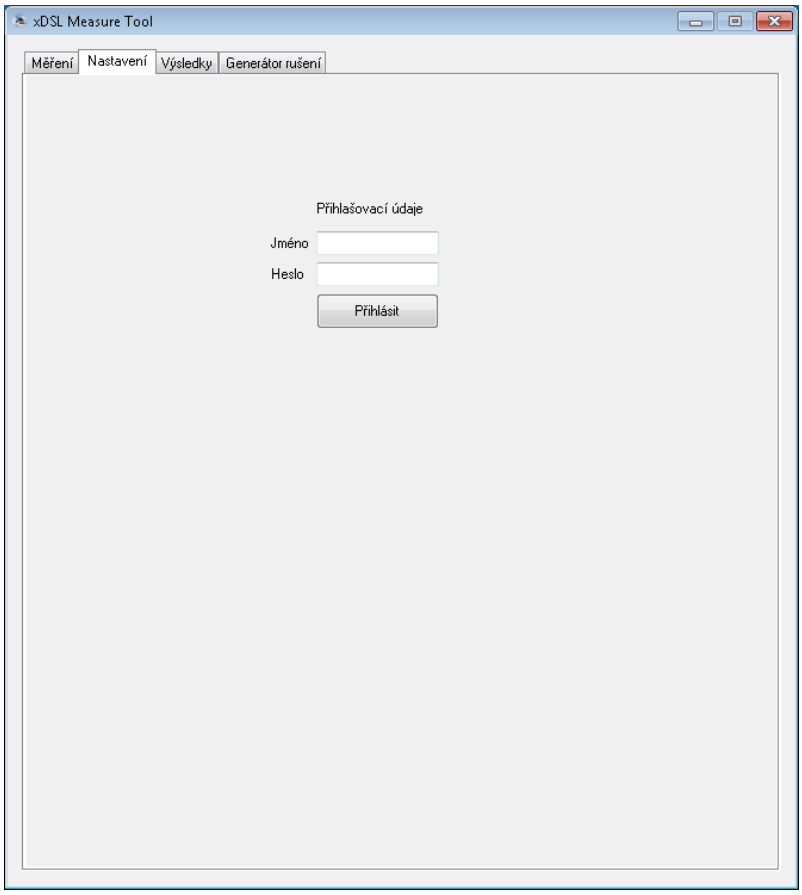
1. V horní části je umístěno tlačítko, pomocí kterého je možné změnit načtený profil pro měření a úplně vpravo je tlačítko pro vyvolání stručné nápovědy.
2. Tlačítko a panel s porty zařízení slouží k zjištění, které porty připojeného DSLAMu jsou aktivní, po stisku tlačítka proběhne kontrola a následně jsou aktivní porty zobrazeny na panelu.
3. Tlačítko pro spuštění měření s parametry nastavenými v panelu nad tlačítkem. Je zde možné nastavit parametry pro měření bez rušení, jako jsou maximální a minimální měřená vzdálenost a měřený krok, případně zaškrtnout volbu *Načtení parametrů z profilu rušení*, která spustí proměřování sekvencí z profilu rušení.
4. Ve spodní části okna je zobrazen panel s průběhem měření a pod ním jsou vypisovány zaslané příkazy simulátorům vedení a DSLAMu.



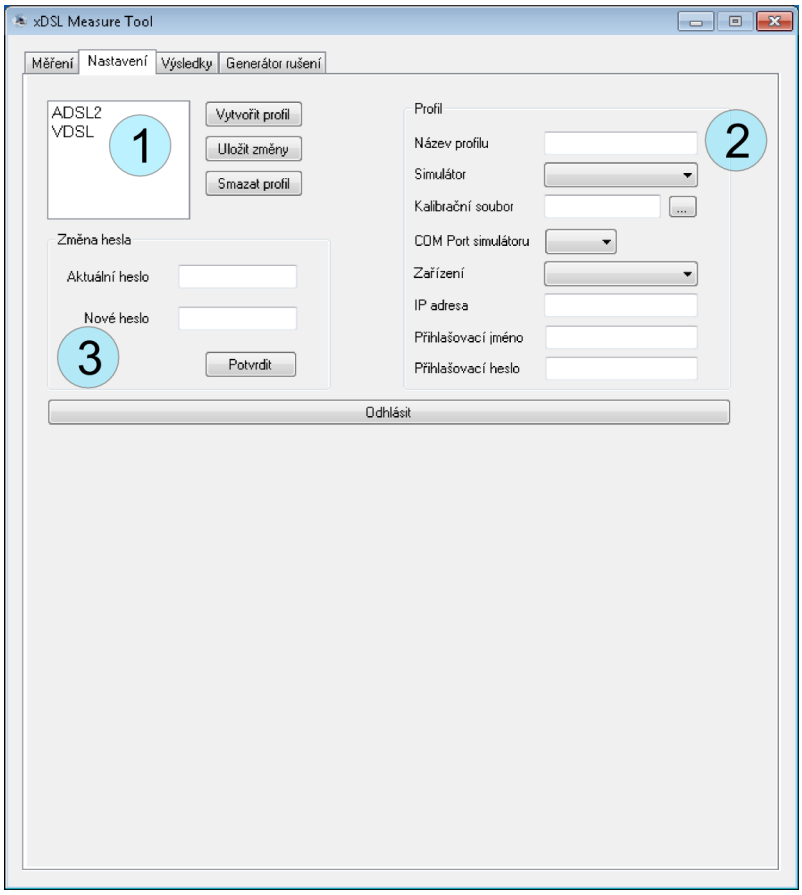
Obr. A.2: Karta Měření

A.1.2 Karta Nastavení

Po příhlášení do nastavení A.3, jehož defaultní přihlašovací údaje jsou `admin` bez hesla, se zobrazí karta A.4, na které je možné přihlašovací údaje změnit v části `3`. V nastavení je možné vytvořil profil pro měření, to se provádí v části 2, kde je nutné vyplnit *název profilu*, zvolit mezi simulátorem vedení pro ADSL případně VDSL technologie a vyhledat příslušný kalibrační soubor. Další je položka *COM Port simulátoru*, kde se volí příslušný port. Následuje volba DSLAMu v položce *Zařízení*, jehož *IP adresa, přihlašovací jméno a heslo* se vypíše do příslušných formulářů.



Obr. A.3: Karta Nastavení – přihlášení

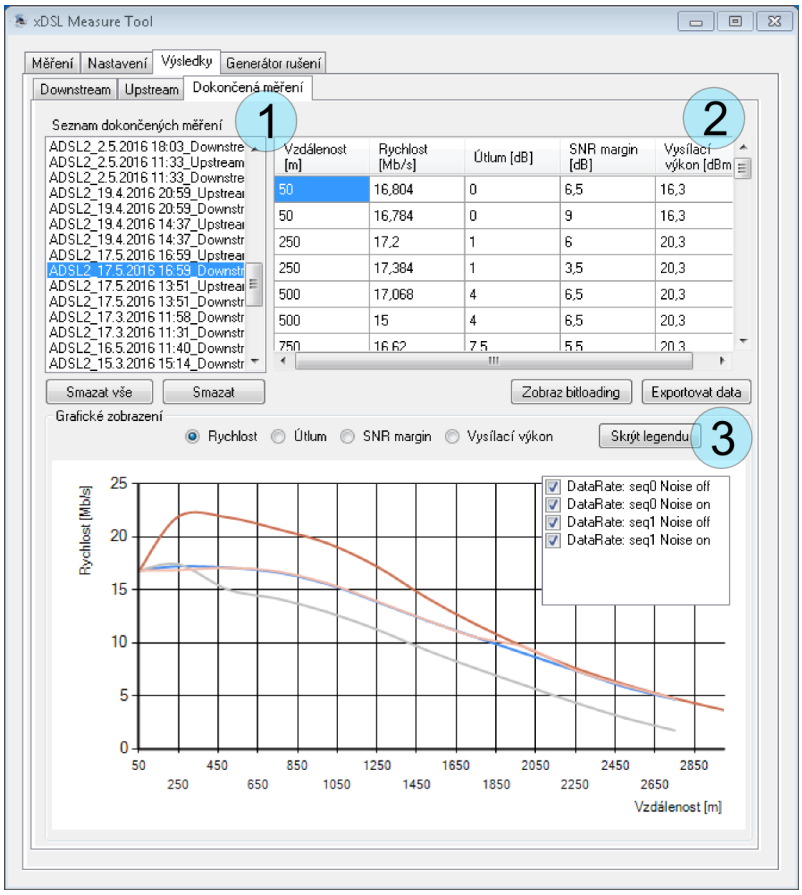


Obr. A.4: Karta Nastavení

A.1.3 Karta Výsledky

Karta s výsledky je rozdělana na 3 záložky. V prvních 2 jsou graficky zobrazena probíhající měření pro *Downstream* a *Upstream*. Ve třetí záložce se pak nacházejí dokončená měření, která jsou uložena ve složce *Mer* umístěné v kořenovém adresáři programu. V části 1 je seznam dokončených měření, pod kterým jsou tlačítka pro *Smazat* a *Smazat vše* pro odstranění naměřených dat. Naměřené údaje jsou zobrazeny v tabulce v části 2, ve které je v prvním sloupečku uvedena délka měřeného vedení, následují údaje o přenosové rychlosti [Mb/s], útlum[dB], SNR margin [dB] a vysílací výkon [dBm]. Pod touto tabulkou je tlačítko *Zobraz bitloading* pro vyvolání okna s 3D grafickým zobrazením bitloadingu a tlačítko *Exportovat data* pro export do formátu textového souboru, Matlabu nebo Excelu.

V části 3 je graf naměřených hodnot, nad ním je možné změnit naměřený parametr. Vpravo je zobrazena legenda, která umožňuje zobrazit jen některé naměřené průběhy a zlepšuje tak přehlednost. V případě, kdy zasahuje graf do legendy je možné ji tlačítkem *Skrýt legendu* vypnout.

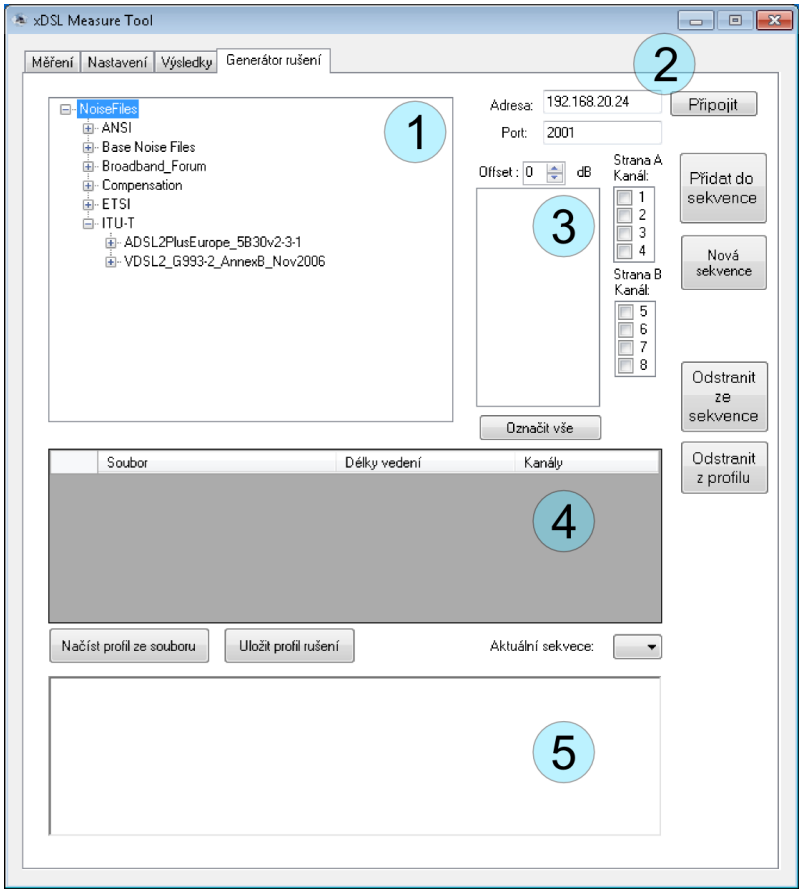


Obr. A.5: Karta Výsledky

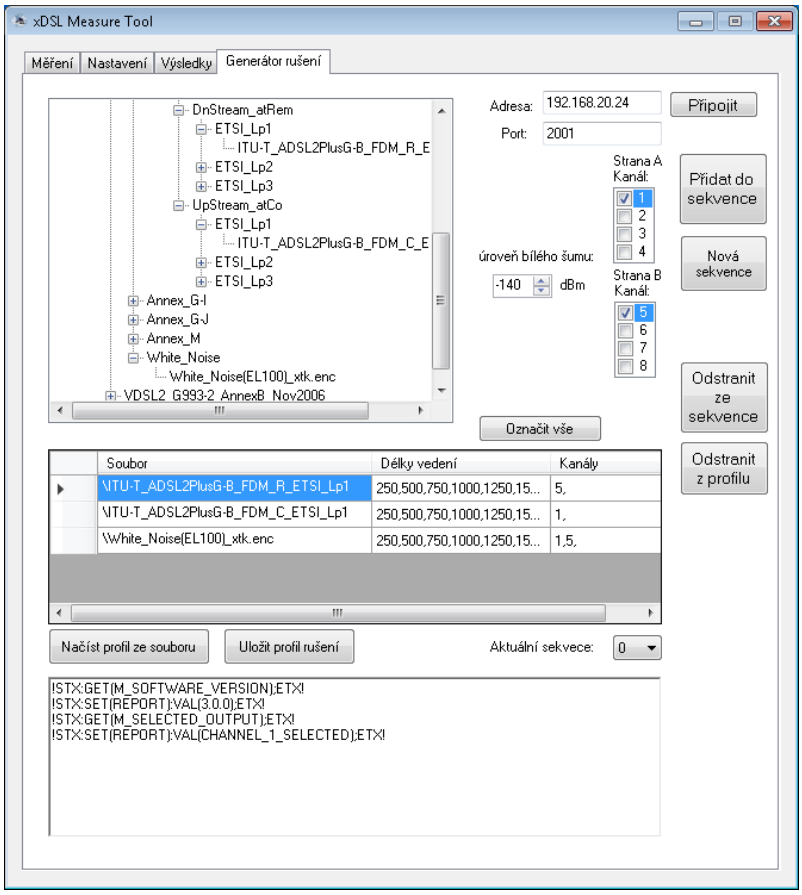
A.1.4 Karta Generátor rušení

Na této kartě je možné vytvořit a uložit profil rušení, který se skládá z jednotlivých sekvencí. Okno s touto kartou je zobrazeno na A.6. V části 1 jsou vybírány jednotlivé soubory rušení ze stromové struktury. Nutnou podmínkou je mít na disku D kopii knihovny souborů od výrobce generátoru. V části 3 jsou pro přeslechová rušení zobrazeny jednotlivé délky vedení, pro které existují soubory v knihovně generátoru rušení. Nad seznamem délek je možné nastavit offset úrovně rušení od jeho referenční úrovně. V případě výběru bílého šumu je zobrazeno pouze nastavení úrovně spektrální hustoty bílého šumu. Následně je možné vybrat výstupní kanály generátoru a injektoru pomocí zaškrtnutí příslušného čísla, přičemž čísla kanálů 1–4 náleží straně A (strana DSLAMu) a čísla 5–8 straně B (strana modemu). Vytváření sekvencí probíhá pomocí tlačítka *Nová sekvence* a vybraný typ rušení lze přidat stisknutím *přidat do sekvence*. Pro správnou funkci programu je nutné vybírat nejprve přeslechová rušení a volit stejné délky vedení, následně je možné volit rušení typu bílý šum, ten je poté generovaný na všech měřených délkách sekvence. Aktuální sekvence profilu je zobrazena v části 4 a pomocí volby *Aktuální sekvence* ji lze

změnit. Pomocí tlačítka *Odstranit ze sekvence* je možné vybrané rušení ze sekvence vymazat, nebo tlačítkem *Odstranit z profilu* vymazat z profilu celou sekvenci. Příklad vytvořené sekvence je na obrázku A.7. Vytvořený profil je možné uložit a načíst ze souboru pomocí příslušných tlačítek *Načíst profil ze souboru* a *Uložit profil rušení*. Před samotným měřením je nutné přepnout generátor do režimu vzdáleného řízení a inicializovat komunikaci pomocí protokolu Telnet. K tomu je nutné v části 2 nastavit adresu a port a stisknut tlačítko *Připojit*. Veškeré komunikace s generátorem je vypisována v částí označené 5.



Obr. A.6: Karta Generátor rušení



Obr. A.7: Karta Generátor rušení – příklad sekvence

A.2 Dokumentace

A.2.1 Komunikace s generátorem

Implementace funkcí pro komunikaci s generátorem DLS-5800. Výstupem jsou řetězce řídících příkazů, které jsou popsané v návodu výrobce [13].

Systémové a sítově příkazy

```
char* Form1::SOFTWARE_VERSION(){
   return ("!STX:GET(M_SOFTWARE_VERSION);ETX!");
}

char* Form1::INSTALLED_PACKAGES(){
   return ("!STX:GET(M_INSTALLED_PACKAGES);ETX!");
}

char* Form1::MAC_ADDRESS(){
   return ("!STX:GET(M_MAC_ADDRESS);ETX!");
}
char* Form1::NETWORK_NAME(){
```

```
   return ("!STX:GET(M_NETWORK_NAME);ETX! ");
}
char* Form1::SYSTEM_ID(){
   return ("!STX:GET(M_SYSTEM_ID);ETX! ");
   }}
```

Příkazy pro ovládání kanálů

```
char* Form1::SELECTED_OUTPUT(){
   return ("!STX:GET(M_SELECTED_OUTPUT);ETX!");
   }
char* Form1::FILE_NAMES(String^ name){
   String^ str = "!STX:GET(M_FILE_NAMES):VAL("+name+");ETX!";

   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);
   }

char* Form1::OUTPUT_LEVEL(String^ output){
   String^ str = "!STX:GET(M_OUTPUT_LEVEL):VAL("+output+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);
   }

char* Form1::SELECT_OUTPUT(String^ output){

   String^ str = "!STX:SET(M_SELECT_OUTPUT):VAL("+output+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }

char* Form1::LOAD_FILE(String^ fileName){
   String^ str = "!STX:SET(M_LOAD_FILE):VAL("+fileName+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }

char* Form1::GENERATE_SAMPLE(){
   return ("!STX:SET(M_GENERATE_SAMPLE);ETX!");
   }
char* Form1::RESET_CHANNEL(){
   return ("!STX:SET(M_RESET_CHANNEL);ETX!");
   }

char* Form1::LOAD_OUTPUT(){
   return ("!STX:SET(M_LOAD_OUTPUT);ETX!");
   }

char* Form1::NOISE_GAIN(double noiseGain){

   String^ str = "!STX:SET(M_NOISE_GAIN):VAL("+noiseGain+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);


   }

char* Form1::NOISE_GAINX(int fileName,double noiseGain){
   String^ str = "!STX:SET(M_NOISE_GAINX):VAL("+fileName+":"+noiseGain+");ETX!";
```

```
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }

char* Form1::MICRO_GAIN(double microGain){
   String^ str = "!STX:SET(M_MICRO_GAIN):VAL("+microGain+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }
char* Form1::CREST_FACTOR(bool enable){

   if(enable) return "!STX:SET(M_CREST_FACTOR):VAL(ON);ETX!";
   else return "!STX:SET(M_CREST_FACTOR):VAL(OFF);ETX!";
   }

char* Form1::NUMBER_SAMPLES (int numberSamples){
   String^ str = "!STX:SET(M_NUMBER_SAMPLES):VAL("+numberSamples+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }

char* Form1::TD_WHITENOISE(bool enable){

   if(enable) return "!STX:SET(M_TD_WHITENOISE):VAL(ON);ETX!";
   else return "!STX:SET(M_TD_WHITENOISE):VAL(OFF);ETX! ";
   }

char* Form1::IMPULSE_RATE (int impulse_rate){
   String^ str = "!STX:SET(M_IMPULSE_RATE):VAL("+impulse_rate+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);


   }

char* Form1::CLEARWORKSPACE(){
   return ("!STX:SET(M_CLEARWORKSPACE);ETX!");
   }
char* Form1::ENABLE_OUTPUT(int output,bool enable){
   String^ ena="";
   if(enable) ena = "ON";
   else ena = "OFF" ;
   String^ str = "!STX:SET(M_ENABLE_OUTPUT):VAL(OUTPUT_"+output+":"+ena+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }

char* Form1::SAVE_CUSTOM_FILE (String^ fileName){
   String^ str = "!STX:SET(M_SAVE_CUSTOM_FILE):VAL("+fileName+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }
```

Příkazy pro ovládání injektoru

```
   char* Form1::INJ_CHAN(int n,bool enable){
   String^ ena="";
```

```
   if(enable) ena = "1";
   else ena = "0" ;
   String^ str = "!STX:SET(M_INJ_CHAN"+n+"_STATE):VAL("+ena+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);

   }

   char* Form1::INJ_CHAN_DIFF(int n,bool enable){
   String^ ena="";
   if(enable) ena = "ON";
   else ena = "OFF" ;
   String^ str = "!STX:SET(M_ENABLE_OUTPUT):VAL(OUTPUT_"+n+":"+ena+");ETX!";
   return (char*)(void*)Marshal::StringToHGlobalAnsi(str);


   }
```

Komunikace pomocí protokolu Telnet

```
      char * Form1::telnet_sendandrecv(char *buffer2)
{


      send(s2,buffer2,1000,0);


      Sleep(1000);
      if ((recv_size2=recv(s2,server_reply2,10000,0))==SOCKET_ERROR)
      {
         this->richTextBox2->Text+="recv failed";

      }

      return server_reply2;
}
```

A.2.2 Profily a sekvence

NoiseProfile

Hlavička třídy NoiseProfile, *MeasureSeq* slouží k uložení sekvencí daného profilu. Jednotlivé důležité metody jsou popsány dále.

```
public ref class NoiseProfile{
public:

   String^ name;

   List <Sequence^>^ MeasureSeq;
   NoiseProfile(String^ jmeno);
   NoiseProfile();
   void ulozProfil(String^ soubor);
```

```
   void nactiProfil(Dictionary<String^,Dictionary<Int32,String^>^>^ wirelines,List<String^>^
        whitenoises,String^ soubor);
   void addToProfile(Sequence^ seq);
   Sequence^ getSequence(int k);
   void removeSequence(Sequence^);
   int seqCount();

   String^ getProfileName();
};
```

Uložení profilu Následující metoda slouží k uložení profilu do souboru, jehož název je vstupním parametrem. Na prvním řádku je název profilu, postupně jsou zapisovány sekvence a jejich parametry, tak jak byly vytvořeny. Každá hodnota je na vlastním řádku, jednotlivé parametry jsou odděleny řetězci: *seq files:,seq lenghts:,seq channels:,seq levels:*.

```
void NoiseProfile::ulozProfil(String^ soubor){
   try
   {
      StreamWriter^ sw = gcnew StreamWriter(soubor);
      sw->WriteLine(name);


      for(int i = 0;i<MeasureSeq->Count;i++){
         sw->WriteLine("seq files:");
         Dictionary<String^,Dictionary<Int32,String^>^>::KeyCollection^ stringFiles
              =MeasureSeq[i]->wirelines->Keys;

         for each(String^ s in stringFiles ){
            sw->WriteLine(s);
         }
         sw->WriteLine("seq lenghts:");
         List<int>^ wirelineLenghts = MeasureSeq[i]->getMesureList();
         for each(int l in wirelineLenghts){
            sw->WriteLine(l);
         }


         sw->WriteLine("seq channels:");
         for each(String^ s in stringFiles ){
            List<int>^ channels = MeasureSeq[i]->getSeqChannels(s);
            String^ line = "";
            for each(int c in channels)
            {
               line += c +"," ;

            }
            sw->WriteLine(line);
         }


         sw->WriteLine("seq levels:");


         for each(String^ s in stringFiles ){
```

```
            sw->WriteLine(MeasureSeq[i]->getLevel(s));
         }

      }
      sw->Close();
   }
   catch (Exception^ e)
   {
      if (dynamic_cast<FileNotFoundException^>(e))
         Console::WriteLine("file '{0}' not found", soubor);
      else
         Console::WriteLine(e->Message);
   }
}
```

Příklad souboru

```
defaultProfile
seq files:
D:\NoiseFiles\ETSI\ADSL2PlusFB_5B30v2-3-1\Annex_B\DnStream_atRem\ETSI_Lp1\ETSI-ADSL2PlusFB-B_R_Lp1
D:\NoiseFiles\ETSI\ADSL2PlusFB_5B30v2-3-1\White_Noise\White_Noise(EL135)_xtk.enc
seq lenghts:
50
250
500
750
1000
1250
1500
1750
2000
2250
2500
2750
3000
3250
3500
3750
4000
seq channels:
5,
5,
seq levels:
0
0
seq files:
D:\NoiseFiles\ETSI\ADSL2PlusFB_5B30v2-3-1\Annex_B\UpStream_atCo\ETSI_Lp1\ETSI-ADSL2PlusFB-B_C_Lp1
D:\NoiseFiles\ETSI\ADSL2PlusFB_5B30v2-3-1\White_Noise\White_Noise(EL135)_xtk.enc
seq lenghts:
50
250
500
750
1000
1250
1500
1750
2000
2250
```

```
2500
2750
3000
3250
3500
3750
4000
seq channels:
1,
1,
seq levels:
0
0
seq files:
D:\NoiseFiles\ETSI\ADSL2PlusFB_5B30v2-3-1\Annex_B\UpStream_atCo\ETSI_Lp1\ETSI-ADSL2PlusFB-B_C_Lp1
D:\NoiseFiles\ETSI\ADSL2PlusFB_5B30v2-3-1\Annex_B\DnStream_atRem\ETSI_Lp1\ETSI-ADSL2PlusFB-B_R_Lp1
D:\NoiseFiles\ETSI\ADSL2PlusFB_5B30v2-3-1\White_Noise\White_Noise(EL135)_xtk.enc
seq lenghts:
50
250
500
750
1000
1250
1500
1750
2000
2250
2500
2750
3000
3250
3500
3750
4000
seq channels:
1,
5,
1,5,
seq levels:
0
0
0
```

Načtení profilu Následující metoda slouží k načtení údajů do profilu ze souboru, jehož generování a příklad je uveden v předchozí části. Využívá pomocnou sekvenci *sq*, do níž jsou načítány údaje, pomocné slovníky *DicIn* a *DicOut* pomocí kterých je kontrolováno, zda jsou názvy souborů validní a jedná se o soubory, které jsou obsaženy v knihovně generátoru.

```
void NoiseProfile::nactiProfil(Dictionary<String^,Dictionary<Int32,String^>^>^
     wirelines,List<String^>^ whitenoises,String^ soubor){



   Dictionary<Int32,String^>^ dicIn = gcnew Dictionary<Int32,String^>();
```

```
List<int>^ lenList = gcnew List<int>();




try
{

   StreamReader^ din = File::OpenText(soubor);
   String^ str;
   this->MeasureSeq->Clear();
   name = din->ReadLine();

   din->ReadLine();
   do{

      List<List<int>^>^ channelsList = gcnew List<List<int>^>();
      Sequence^ sq = gcnew Sequence();
      List<double>^ noiseLevels = gcnew List<double>();
      while((str = din->ReadLine()) != "seq lenghts:" ){
         sq->noiseFiles->Add(str);
      }
      while((str = din->ReadLine()) !="seq channels:"){

         int hodnota;
         if(Int32::TryParse(str,hodnota)) sq->wirelineLenghts->Add(hodnota);
      }
      while((str = din->ReadLine()) !="seq levels:")
      {List<int>^ channels = gcnew List<int>();
      String^ delimStr = ",";
      array<Char>^ delimiter = delimStr->ToCharArray( );
      array<String^>^ split = str->Split(delimiter);
      for (int k = 0; k<split->Length-1;k++){
         int hodnota = Convert::ToInt32(split[k],10);
         channels->Add(hodnota);}
      channelsList->Add(channels);
      //channels->Clear();
      }
      while((str = din->ReadLine()) != nullptr){
         if(str== "seq files:")break;
         double hodnota = Convert::ToDouble(str);
         noiseLevels->Add(hodnota);
      }


      int myEnumC = 0;

      for each (String^ nfs in sq->noiseFiles ){

         Dictionary<Int32,String^>^ dicOut = gcnew Dictionary<Int32,String^>();


         if(wirelines->ContainsKey(nfs)){

            dicIn = wirelines[nfs];
            System::Collections::IEnumerator^ myEnum = sq->wirelineLenghts->GetEnumerator();
```

```
            while ( myEnum->MoveNext() )
            {
               int itemChecked = safe_cast<int>(myEnum->Current);
               dicOut->Add(itemChecked, dicIn[itemChecked]);

            }


            sq->addToSeq(nfs,dicOut,channelsList[myEnumC],noiseLevels[myEnumC]);

            myEnumC++;

         }else if(whitenoises->Contains(nfs)){

            List<int>^ mesureList = sq->wirelineLenghts;
            for each(int l in mesureList){
               if(!dicOut->ContainsKey(l))dicOut->Add(l,nfs);}

            sq->addToSeq(nfs,dicOut,channelsList[myEnumC],noiseLevels[myEnumC]);
            myEnumC++;



         }

      }
      this->addToProfile(sq);



   }while(str!= nullptr);
   din->Close();

}
catch (Exception^ e)
{
   if (dynamic_cast<FileNotFoundException^>(e))
      Console::WriteLine("file '{0}' not found", soubor);
   else
      Console::WriteLine(e->Message);
}



}
```

Přidání a odstranění sekvencí profilu Tyto metody slouží k manipulaci se sekvencemi daného profilu a využívají metody a vlastnosti *Add(), Count, Remove()* třídy *List<>*, jenž je součástí *.NET*.

```
void NoiseProfile::addToProfile(Sequence^ seq){
   this->MeasureSeq->Add(seq);
}
int NoiseProfile::seqCount(){
   return MeasureSeq->Count;
}

Sequence^ NoiseProfile::getSequence(int k){
   return MeasureSeq[k];
}

void NoiseProfile::removeSequence(Sequence^ k){
   MeasureSeq->Remove(k);
}
```

Sequence

Hlavička třídy *Sequence*. Proměnná *wirelines* slovníkového typu, slouží k uložení prvků sekvence a vzájemných vazeb mezi zástupným názvem souboru s údaji o rušení, měřenou vzdáleností a názvem obsahující délku vedení, který je načítán na výstup generátoru. Podobně fungují i *channels, noiseLevels* zajišťující uložení údajů o měřených kanálech a úrovních. Pomocné seznamy *wirelineLenghts, noiseFiles* slouží pro práci při manipulaci se sekvencemi.

```
public ref class Sequence{
private: Dictionary<String^,Dictionary<Int32,String^>^>^ wirelines;
      Dictionary<String^,List<int>^>^ channels ;

      List <int>^ wirelineLenghts;
      Dictionary<String^,double>^ noiseLevels;
      List <String^>^ noiseFiles;
public: Sequence();
     void addToSeq(String^ key,Dictionary<Int32,String^>^ lenghts,List<int>^ channelsList);
     void addToSeq(String^ key,Dictionary<Int32,String^>^ lenghts,List<int>^ channelsList, double
          noiseLevel );
     void dellFromSeq(String^ key);
     Dictionary<String^,Dictionary<Int32,String^>^>^ getSeqDictionary();
     List<int>^ getSeqChannels(String^ key);
     List<int>^ Sequence::getMesureList();
     double getLevel(String^ key);

};
```

Práce se sekvencemi Pro práci se sekvencemi jsou využívány třídy frameworku *.NET* a jejich metody, jedná se zejména o třídy *Dictionary a List*.

```
void Sequence::addToSeq(String^ key,Dictionary<Int32,String^>^ lenghts,List<int>^ channelsList){
   if(!wirelines->ContainsKey(key))wirelines->Add(key,lenghts);


   if(!channels->ContainsKey(key))channels->Add(key,channelsList);
```

```
}

void Sequence::addToSeq(String^ key,Dictionary<Int32,String^>^ lenghts,List<int>^
     channelsList,double noiseLevel){
   if(!wirelines->ContainsKey(key))wirelines->Add(key,lenghts);
   if(!channels->ContainsKey(key))channels->Add(key,channelsList);
   if(!noiseLevels->ContainsKey(key))noiseLevels->Add(key,noiseLevel);


}

void Sequence::dellFromSeq(String^ key){
   wirelines->Remove(key);
   channels->Remove(key);
   noiseLevels->Remove(key);
}

Dictionary<String^,Dictionary<Int32,String^>^>^ Sequence::getSeqDictionary(){
   return wirelines;

}

List<int>^ Sequence::getMesureList(){
   List<int>^ mesureList = gcnew List<int>();
   Dictionary<String^,Dictionary<Int32,String^>^>::KeyCollection^ stringKeys = wirelines->Keys;


   for each(String^ k in stringKeys){
      Dictionary<Int32,String^>::KeyCollection^ lenKeys = wirelines[k]->Keys;

      for each(int l in lenKeys){
         if(!mesureList->Contains(l)){
            mesureList->Add(l);
         }

      }

   }


   return mesureList;
}

List<int>^ Sequence::getSeqChannels(String^ key){
   return channels[key];

}

List<int>^ Sequence::getSeqChannels(){
   List<int>^ channs = gcnew List<int>();
   List<int>^ ch = gcnew List<int>();
   for each(List<int>^ ch in channels->Values){
      channs->AddRange(ch);
   }
   return channs;

}
```

```
double Sequence::getLevel(String^ key){
   return noiseLevels[key];

}
```

A.2.3 Měřicí proces

Měřicí proces je zahájen voláním funkce *mereniRuseni*, která vznikla úpravou původní funkce *mereni* [12].

```
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
            if (threadactive==false && threadactive2==false)
            {if(checkBox2->Checked){

      radioButton16_CheckedChanged(sender,e);
         ParameterizedThreadStart ^myThreadDelegate = gcnew
              ParameterizedThreadStart(this,&DSL_Measure_tool::Form1::mereniRuseni);//mereni
          trd = gcnew Thread(myThreadDelegate);
          trd->IsBackground = true;
          threadactive=true;
          this->button2->Text=" Peruit ";
          this->dataGridView1->Rows->Clear();
          this->dataGridView2->Rows->Clear();
          this->chart1->Series->Clear();
          this->chart2->Series->Clear();
          this->label26->Show();

               trd->Start(vybrany_profil);
```

Jelikož je funkce velice rozsáhlá, budou uvedeny pouze hlavní úpravy a rozdíly oproti té původní.

Kontrola spojení s generátorem

Před samotným měřením je provedena kontrola spojení s generátorem rušení, pokud je neúspěšná, tak je zobrazeno chybové hlášení a měření je ukončeno.

```
if(!Thread_isGenOnline()){
      Thread_Messagebox(" Men  bylo ukoneno.\\nDvod: Komunikace s genertorem  ruen  se nezdaila.
           \\nZkontrolujte spojen."
      ," Varovn ",MessageBoxIcon::Error);
         this->serialPort1->Close();
         ukonci_spojeni();
         threadactive=false;
         Thread_button_rename();
         trd->Abort();}
```

Informace, zda je komunikace aktivní, je získána zasláním známého příkazu a očekáváním známé odpovědi.

```
bool Form1::Thread_isGenOnline(){

        String^ testGen = marshal_as<String^>(telnet_sendandrecv(SOFTWARE_VERSION()));
        if(testGen->Contains("!STX:SET(REPORT):VAL(3.0.0);ETX!")){return true;}
        else{ return false;}



    }
```

Počet měření pro rušení

Pro připravení pole k uložení výsledků, které jsou součástí původního programu, je nutné znát počet měřících cyklů. Ten je stanoven z údajů všech sekvencí, které budou měřeny. Voláním *getMesureList()* je získán seznam měřených délek vedení a pomocí cyklu je stanoven počet měřených délek celého profilu, ten je násoben 2, protože je měření realizováno pro každou vzdálenost samostatně s rušením a bez rušení.

```
int pocet= 0;
     for(int i = 0;i<this->defaultProfile->seqCount();i++){
     List<int>^ mesureList = this->defaultProfile->getSequence(i)->getMesureList();
     pocet = pocet + mesureList->Count;
     }
     pocet = pocet*2;
```

Měřicí smyčka

Měřicí smyčka začíná cyklem `for`, ten se opakuje tolikrát, kolikrát je nastaveno v GUI, avšak měření dat je v cyklu provedeno dvakrát, jednou bez rušení a podruhé s rušením. V dalším cyklu `for` začíná měření jednotlivých sekvencí. Pomocí funkce *thread_nova_serie* jsou vytvořeny řady pro grafy, ve kterých jsou při měření zobrazována naměřená data. V následujícím cyklu `for` začíná měření na jednotlivých délkách získaných z pomocného seznamu *mesureList*, který je naplněný voláním metod *getSequence(i)*, *getMesureList()* nad globální proměnou *defaultProfile* ve které je uložen načtený profil. V následující části probíhá kontrola připojení k rozhraní simulátoru vedení, v případě neúspěchu jsou údaje o sekvenci a měřené vzdálenosti uloženy do chybového logu. Voláním funkcí z původního programu jsou *zprava_serveru* a *ThreadTask_vizualizace_dat* získány a zobrazeny naměřené údaje z DSLAMu.

```
for (int
     opakovani=1;opakovani<=System::Convert::ToInt32(this->numericUpDown4->Value)*2;opakovani+=2)
     {
```

```
kontrola=0;

for(int i = 0;i<this->defaultProfile->seqCount();i++){
thread_nova_serie(String::Concat("DataRate: seq",System::Convert::ToString(i)," Noise
    off"),"ChartArea1",0);
thread_nova_serie(String::Concat("Attenuation: seq",System::Convert::ToString(i)," Noise
    off"),"ChartArea2",0);
thread_nova_serie(String::Concat("SnrMargin: seq",System::Convert::ToString(i)," Noise
    off"),"ChartArea3",0);
thread_nova_serie(String::Concat("OutputPwr: seq",System::Convert::ToString(i)," Noise
    off"),"ChartArea4",0);

thread_nova_serie(String::Concat("DataRate: seq",System::Convert::ToString(i)," Noise
    on"),"ChartArea1",0);
thread_nova_serie(String::Concat("Attenuation: seq",System::Convert::ToString(i)," Noise
    on"),"ChartArea2",0);
thread_nova_serie(String::Concat("SnrMargin: seq",System::Convert::ToString(i)," Noise
    on"),"ChartArea3",0);
thread_nova_serie(String::Concat("OutputPwr: seq",System::Convert::ToString(i)," Noise
    on"),"ChartArea4",0);

Dictionary<String^,Dictionary<Int32,String^>^>^ wirelines =
    this->defaultProfile->getSequence(i)->getSeqDictionary();
Dictionary<String^,Dictionary<Int32,String^>^>::KeyCollection^ stringKeys = wirelines->Keys;

List<int>^ mesureList = this->defaultProfile->getSequence(i)->getMesureList();

for(int n = 0; n<mesureList->Count;n++){ //men na vech   dlkch   sekvence
  int len = find_length(mesureList[n]);

  Thread_Zmena_delky(len);
  Sleep(60000);
  Thread_zmena_intervalu(String::Concat("DataRate: seq",System::Convert::ToString(i),"
      Noise off"),"ChartArea1",index,opakovani);
  Thread_zmena_intervalu(String::Concat("Attenuation: seq",System::Convert::ToString(i),"
      Noise off"),"ChartArea2",index,opakovani);
  Thread_zmena_intervalu(String::Concat("SnrMargin: seq",System::Convert::ToString(i),"
      Noise off"),"ChartArea3",index,opakovani);
  Thread_zmena_intervalu(String::Concat("OutputPwr seq",System::Convert::ToString(i),"
      Noise off"),"ChartArea4",index,opakovani);



  pokus2=0;
  ThreadTask(marshal_as<String^>(profil[cislo_profilu].syntax.port_status)+ComboBoxItem+"\n");
  Kontrola_rozhrani_ruseni(profil[cislo_profilu]);
  if (kontrola==-1)
  {measureLog += "Nepodailo se pipojit k rozhrani:\n";
   measureLog += "Sekvence "+i+", dlka veden "+mesureList[n]+"m\n";
    while(len<=max)
    {
      Thread_Progress_bar(((max-min)/krok)+1);
      len+=krok;
    }
    pokus2=0;
    kontrola = 0;
    index++;
    break;
```

```
}


clear_array();

zprava_serveru(marshal_as<String^>(profil[cislo_profilu].syntax.rychlost_ds),ComboBoxItem,"\n");
ThreadTask(marshal_as<String^>(profil[cislo_profilu].syntax.rychlost_ds)+ComboBoxItem+"\n");
Sleep(50000);
ThreadTask_vizualizace_dat(len,0,index,cislo_profilu,opakovani,i);
clear_array();

if (this->radioButton57->Checked==true)
{
   thread_nova_serie(String::Concat("DataRate: seq",System::Convert::ToString(i)," Noise
        off"),"ChartArea1",1);
   thread_nova_serie(String::Concat("Attenuation: seq",System::Convert::ToString(i),"
        Noise off"),"ChartArea2",1);
   thread_nova_serie(String::Concat("SnrMargin: seq",System::Convert::ToString(i),"
        Noise off"),"ChartArea3",1);
   thread_nova_serie(String::Concat("OutputPwr: seq",System::Convert::ToString(i),"
        Noise off"),"ChartArea4",1);

   if (profil[cislo_profilu].syntax.typ=="Planet VC-820M")
   {
      zprava_serveru(marshal_as<String^>(profil[cislo_profilu].syntax.rychlost_us),ComboBoxItem,"\n");
      Sleep(40000);
      ThreadTask_vizualizace_dat(len,1,index,cislo_profilu,opakovani,i);
      clear_array();
   }else
   {
      zprava_serveru(marshal_as<String^>(profil[cislo_profilu].syntax.rychlost_us),ComboBoxItem,"\n");
      ThreadTask(marshal_as<String^>(profil[cislo_profilu].syntax.rychlost_us)+ComboBoxItem+"\n");
      Sleep(60000);
      ThreadTask_vizualizace_dat(len,1,index,cislo_profilu,opakovani,i);
      clear_array();
   }

}
Thread_Progress_bar(((max-min)/krok)+1);

index++;
```

Generovaní rušení

Způsob generování rušení je řešený trochu zvláštně, protože do poslední chvíle způsoboval problémy při určitých vstupních kombinacích zvolených souborů a požadovaných kanálů.

Pomocí seznamu *allChannels* a cyklu `for each` jsou procházeny veškeré kanály dané sekvence, v dalším cyklu `for each` jsou procházeny soubory dané sekvence a je zjišťováno, zda je pro dané číslo kanálu c nalezeno stejné číslo kanálu v seznamu kanálů *channels* pro daný soubor sekvence. Pokud je podmínka splněna, je daný soubor uložen na kanál c, pokud není, pokračuje vnitřní smyčka k dalšímu souboru.

Na konci vnější smyčky je pomocí funkce *Thread_genNoiseLoad* vygenerováno rušení všech požadovaných souborů k na kanál c.

```
List<int>^ allChannels= this->defaultProfile->getSequence(i)->getSeqChannels();
for each (int c in allChannels){

for each(String^ k in stringKeys){
 List<int>^ channels = this->defaultProfile->getSequence(i)->getSeqChannels(k);
 if(channels->Contains(c)){
   List<int>^ fakeChannels = gcnew List<int>();
   fakeChannels->Add(c);
 Thread_genNoiseOn(wirelines[k][mesureList[n]],fakeChannels,
 this->defaultProfile->getSequence(i)->getLevel(k));
 }}
List<int>^ fakeChannels2 = gcnew List<int>();
   fakeChannels2->Add(c);
Thread_genNoiseLoad(fakeChannels2);

}
```

Načtení souboru na kanál Funkce *Thread_genNoiseOn* je určena pro načtení zvoleného souboru pro konkrétní měřenou vzdálenost, na daném kanálu se zvolenou úrovní. Tyto vstupní parametry jsou předávány funkcím pro komunikaci s generátorem.

```
void Form1::Thread_genNoiseOn(String^ noisefile,List<int>^ channels, double level){

if (this->InvokeRequired == false){
   int fail = 0;

   for(int i = 0; i < channels->Count ; i++){
      int channel = channels[i];

   richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(INJ_CHAN(channel,1));
   richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
   richTextBox2->Text = richTextBox2->Text +
        marshal_as<String^>(telnet_sendandrecv(INJ_CHAN(channel,1)));
   richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
   richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(INJ_CHAN_DIFF(channel,1));
   richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
   richTextBox2->Text = richTextBox2->Text +
        marshal_as<String^>(telnet_sendandrecv(INJ_CHAN_DIFF(channel,1)));
   richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
   richTextBox2->Text = richTextBox2->Text +
        marshal_as<String^>(SELECT_OUTPUT("OUTPUT_"+channel));
   richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
   richTextBox2->Text = richTextBox2->Text +
        marshal_as<String^>(telnet_sendandrecv(SELECT_OUTPUT("OUTPUT_"+channel)));
   richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
   String^ fileName = "C:\\Program Files\\Spirent Communications\\DLS
        5800"+noisefile->Substring(2);
   richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(LOAD_FILE(fileName));
   richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
```

```
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(LOAD_FILE(fileName)));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(NOISE_GAIN(level));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(NOISE_GAIN(level)));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;

        }
    }
    else
    {
        DelegateThreadTask10 ^myThreadDelegate = gcnew
            DelegateThreadTask10(this,&Form1::Thread_genNoiseOn);
        this->Invoke(myThreadDelegate,noisefile,channels,level);
    }

}
```

Vygenerování rušení na kanál Funkce *Thread_genNoiseLoad* vygeneruje rušení z načtených souborů na daném kanálu pomocí funkcí pro komunikaci s generátorem. Na konci procesu generování je smazán *workspace* generátoru.

```
void Form1::Thread_genNoiseLoad(List<int>^ channels){

    if (this->InvokeRequired == false){
        for(int i = 0; i < channels->Count ; i++){
            int channel = channels[i];

        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(INJ_CHAN(channel,1));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(INJ_CHAN(channel,1)));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(INJ_CHAN_DIFF(channel,1));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(INJ_CHAN_DIFF(channel,1)));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(SELECT_OUTPUT("OUTPUT_"+channel));

        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(NUMBER_SAMPLES(131072));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(NUMBER_SAMPLES(131072)));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;

        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(CREST_FACTOR(1));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(CREST_FACTOR(1)));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(GENERATE_SAMPLE());
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
```

```
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(GENERATE_SAMPLE()));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(LOAD_OUTPUT());
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(LOAD_OUTPUT()));
        String^ lastString = marshal_as<String^>(telnet_sendandrecv(LOAD_OUTPUT()));


        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(CLEARWORKSPACE());
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(CLEARWORKSPACE()));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;

        }
    }
    else
    {
        DelegateThreadTask11 ^myThreadDelegate = gcnew
            DelegateThreadTask11(this,&Form1::Thread_genNoiseLoad);
        this->Invoke(myThreadDelegate,channels);
    }

  }
```

Restartovaní kanálů

Potom, co je měření na konkrétní vzdálenosti ukončeno, je pomocí následující funkce restartován výstup daných kanálů.

```
if (this->InvokeRequired == false)
     {

        for(int i = 0; i < channels->Count ; i++){
           int channel = channels[i];


        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(SELECT_OUTPUT("OUTPUT_"+channel));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(SELECT_OUTPUT("OUTPUT_"+channel)));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;



        richTextBox2->Text = richTextBox2->Text + marshal_as<String^>(RESET_CHANNEL());
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;
        richTextBox2->Text = richTextBox2->Text +
            marshal_as<String^>(telnet_sendandrecv(RESET_CHANNEL()));
        richTextBox2->Text = richTextBox2->Text +Environment::NewLine;}
    }
    else
    {
```

```
        DelegateThreadTask11 ^myThreadDelegate = gcnew
             DelegateThreadTask11(this,&Form1::Thread_genNoiseOff);
        this->Invoke(myThreadDelegate,channels);
     }

     }
```

Uložení výsledků a ukončení měření

Následující kus kódu slouží k uložení výsledků a ukončuje samotné měření.

```
Thread_Zmena_delky(0);
this->serialPort1->Close();
ukonci_spojeni();
DateTime now= DateTime::Now;
String^ cas=now.ToString("g");

if (System::IO::Directory::Exists("Mer")!=true)
{
   System::IO::Directory::CreateDirectory("Mer");
}
write_txt(String::Concat("Mer/",marshal_as<String^>(profil[cislo_profilu].name),
"_",cas->Replace(':',','),"_Downstream.txt"),download);
this->chart1->Serializer->Save(String::Concat("Mer/",marshal_as<String^>(profil[cislo_profilu].name),"_",
cas->Replace(':',','),"_Downstream.xml"));
     if (profil[cislo_profilu].syntax.typ=="DrayTek Vigor 3600")
     {
        write_bitload(String::Concat("Mer/",marshal_as<String^>(profil[cislo_profilu].name),
        "_",cas->Replace(':',','),"_bitload_us.txt"),tones_us);
     }
     if (this->radioButton57->Checked==true)
     {
        this->chart2->Serializer->Save(String::Concat("Mer/",
        marshal_as<String^>(profil[cislo_profilu].name),"_",cas->Replace(':',','),"_Upstream.xml"));
        write_txt(String::Concat("Mer/",marshal_as<String^>(profil[cislo_profilu].name),"_",
        cas->Replace(':',','),"_Upstream.txt"),upload);
        if (profil[cislo_profilu].syntax.typ=="DrayTek Vigor 3600")
        {
           write_bitload(String::Concat("Mer/",marshal_as<String^>(profil[cislo_profilu].name),
           "_",cas->Replace(':',','),"_bitload_ds.txt"),tones_ds);
        }
     }

     this->defaultProfile->ulozProfil(String::Concat("Mer/",marshal_as<String^>(profil[cislo_profilu].name),
     "_",cas->Replace(':',','),"_noiseprofile.txt"));

     if(measureLog->Length>0){
          try{StreamWriter^ w = gcnew
               StreamWriter(String::Concat("Mer/",marshal_as<String^>(profil[cislo_profilu].name),
          "_",cas->Replace(':',','),"_error_log.txt"));
             w->Write(measureLog);
             w->Close();
          }
```

B PŘILOŽENÉ CD

Obsah přiloženého CD

- *Bakalářská práce Pavel Röszler 2016.pdf*
  – elektronická verze bakalářské práce
- *Projekt Visual Studio*
  – složka s projektem Visual Studio 2010
- *Install DSL measure tool 2*
  – složka s instalačními soubory

SEZNAM PŘÍLOH