České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra radioelektroniky

Využití SDR v LoRaWAN sítích

Utilization of Software Defined Radio in LoRaWAN networks

BAKALÁŘSKÁ PRÁCE

Vypracoval: Jan Zlevor
Vedoucí práce: Doc. Ing. Stanislav Vítek, Ph.D.
Rok: 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zlevor** Jméno: **Jan** Osobní číslo: **483696**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití SDR v LoRaWAN sítích

Název bakalářské práce anglicky:

Utilization of Software Defined Radio in LoRaWAN networks

Pokyny pro vypracování:

1) Prostudujte komunikační protokol LoRa a přístupové vrstvy LoRaWAN
2) Navrhněte a implementujte dekodér rádiové části technologie LoRaWAN
3) Pro implementaci využijte dostupný HW (LimeSDR) a SW (GNURadio)
4) Ověřte komunikaci v síti
5) Navrhněte možné aplikace SDR v LoRaWAN sítích a diskutujte vhodnost zvoleného přístupu

Seznam doporučené literatury:

[1] SENEVIRATNE, Pradeeka. Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power Wireless IoT Networks. Apress, 2019.
[2] JEŘÁBEK, Ondřej.Signálová analýza LoRa s využitím SDR. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky.
[3] TRIPATHY, B. K.; ANURADHA, J. (ed.). Internet of things (IoT): technologies, applications, challenges and solutions. CRC Press, 2017.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Stanislav Vítek, Ph.D., katedra radioelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **03.02.2021** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2022**

doc. Ing. Stanislav Vítek, Ph.D. podpis vedoucí(ho) práce	doc. Ing. Josef Dobeš, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

V Praze dne ....

Jan Zlevor

Poděkování

Děkuji Doc. Ing. Stanislavu Vítkovi, Ph.D. za podnětné návrhy při vedení mé bakalářské práce a za notnou dávku trpělivosti.

Jan Zlevor

Název práce:
Využití SDR v LoRaWAN sítích

Autor: Jan Zlevor

Studijní program: Elektronika a komunikace
Druh práce: Bakalářská práce

Vedoucí práce: Doc. Ing. Stanislav Vítek, Ph.D.
Katedra radioelektroniky

Abstrakt: Cílem práce je sestrojit funkční LoRa transceiver na platformě LimeSDR za využití programových prostředků v GNU Radio. Jádro transceiveru je implementováno jako bloky pro kodér a dekodér v GNU Radio. Blok pro dekodér byl převzat z projektu uvedeného ve zdrojích k práci. Blok pro kodér byl sestrojen v jazyce C++ jako OOT modul. Stojí na základě popisu v teoretické části práce a na analýze zdrojového kódu pro dekodér. V teoretické části práce je především rozebrána fyzická vrstva LoRa z matematického hlediska a stručný popis principů sítí LoRaWAN.
Klíčová slova: LoRa, LoRaWAN, SDR, GNU Radio

Title:
Utilization of Software Defined Radio in LoRaWAN networks

Author: Jan Zlevor

Abstract: This thesis aims to create a functional LoRa transceiver on LimeSDR platform using GNU Radio software. The core of the transceiver is implemented as blocks for encoder and decoder in GNU Radio. Block for the decoder was taken from project given in the thesis resources. The block for encoder was written in C++ programming language as an OOT module. It is based on the theoretic part of the thesis and the analysis of the decoder source code. The theoretic part of this thesis mainly deals with physical layer of LoRa from mathematical point of view and brief description of LoRaWAN networks.
Key words: LoRa, LoRaWAN, SDR, GNU Radio

Obsah

Seznam použitých zkratek

Zkratka	Význam
LoRa	Long Range
CSS	Chirp Spread Spectrum
LPWAN	Low Power Wide Area Network
SDR	Software Defined Radio
GNU	GNU's Not Unix!
SF	Spreading Factor
BW	Bandwidth
CR	Coding Rate
CRC	Cyclic Redundancy Check
USB	Universal Serial Bus
OOT	Out Of Tree

Seznam obrázků

Úvod

LoRa je jedním zástupcem technologie pro komunikaci v internetu věcí. Existuje vedle dalších standardů, kupříkladu vedle Sigfox či ZigBee. Označuje tak protokol, jímž se světem komunikují koncové body v síti. Sítě založené na LoRa se vyznačují velice nízkou spotřebou bateriově napájených koncových zařízení a vysokou vzdáleností na kterou je možné komunikaci uskutečnit. Z těchto vlastností LoRa spadá do kategorie sítí LPWAN (Low Power Wide Area Network).

Cílem práce je popsat princip LoRa a LoRaWAN, a na tomto základě implementovat vysílač a přijímač pro LoRa na platformě LimeSDR.

Práce se v teoretické části zabývá popisem fyzické vrstvy LoRa a pohledem na sítě LoRaWAN. Důraz je kladen na matematický popis signálu a na popis kanálového kódování v LoRa. Dále je principiálně popsána technologie LoRaWAN s odvoláním na specifikaci a popis softwarově definovaného rádia (SDR).

V praktické části je navržena a implementována vysílací a přijímací část LoRa transceiveru v softwaru GNU Radio. Následuje ověření komunikace s pokusem o zdůvodnění výsledků testů.

Kapitola 1

Teoretická část

1.1 LoRa

Jako LoRa je označována modulační technika rádiového přenosu založená na principu rozprostřeného spektra (CSS). Jedná se o proprietární technologii vyvinutou francouzskou firmou Cycleo, následně koupenou společností Semtech v roce 2012. Název technologie pochází z anglického ***Lo****ng* ***Ra****nge*. Z názvu plyne jedna bezesporná výhoda modulace LoRa, a sice, že komunikaci je možno uskutečnit i na vzdálenosti v řádu kilometrů. LoRa disponuje touto vlastností díky použití variace metody rozprostřeného spektra, konkrétně CSS - Chirp Spread Spectrum, o které bude řeč dále.

1.1.1 Metoda rozprostřeného spektra

Pod označením metody rozprostřeného spektra se ukrývá vícero modulačních technik. Společnou vlastností těchto technik je šířka pásma rádiového signálu, která je násobně širší než by vyžadovala informace v něm obsažená. Podle [1] je možné aproximovat Shannon-Hartleyův teorém pro malé hodnoty SNR na

$$\frac{C}{B} \approx \frac{S}{N}, \tag{1.1}$$

kde C je informační propustnost kanálu v bit/s, B je šířka pásma kanálu v Hz a $\frac{S}{N}$ je poměr signálu ku šumu. Z uvedeného vztahu je vidět, že chceme-li zachovat určitou informační propustnost pro velmi malé $\frac{S}{N}$, tak je zapotřebí zvětšit šířku pásma kanálu B, aby zůstaly poměry v 1.1 zachovány. Důsledkem tedy je, že za cenu větší šířky pásma lze komunikovat i za velice špatných podmínek v přenosovém prostředí (komunikace je možná s výkonem přijatého signálu i pod úrovní šumu).

Technik pro rozprostření spektra existuje několikero a principálně se mezi sebou liší. Následující přehled dává velice stručný popis základních používaných technik (často se používají i jejich kombinace které zde zmíněny nebudou). Pro další části bude stěžejní technika CSS. Pro hlubší vysvětlení jednotlivých metod je vhodná kniha [2].

DSSS - Direct Sequence Spread Spectrum

K rozšíření spektra dojde pomocí násobení datového signálu pseudonáhodnou posloupností, jejíž bitová rychlost je násobně větší než rychlost datového signálu.

Pro správné přijetí dat je zapotřebí znát na přijímací straně přesnou podobu posloupnosti použité pro vysílání. DSSS se využívá například v navigačních systémech GPS, Galileo a GLONASS nebo ve standardu IEEE 802.11b

FHSS - Frequency Hopping Spread Spectrum

Jak název napovídá, v této metodě je princip rozprostření založen na postupné změně nosného kmitočtu v určitém intervalu (o délce šířky pásma). Pro úspěšný příjem je opět nutné znát posloupnost která udává v jakém čase se bude vysílat na jakém kmitočtu. Metoda klade nároky na přesnost hodin pro synchronizaci přijímače. Výhodou je odolnost proti úzkopásmovému rušení, nebo jistá odolnost proti odposlechu. Využití FHSS je například ve standardu Bluetooth či ve vojenské komunikaci.

THSS - Time Hopping Spread Spectrum

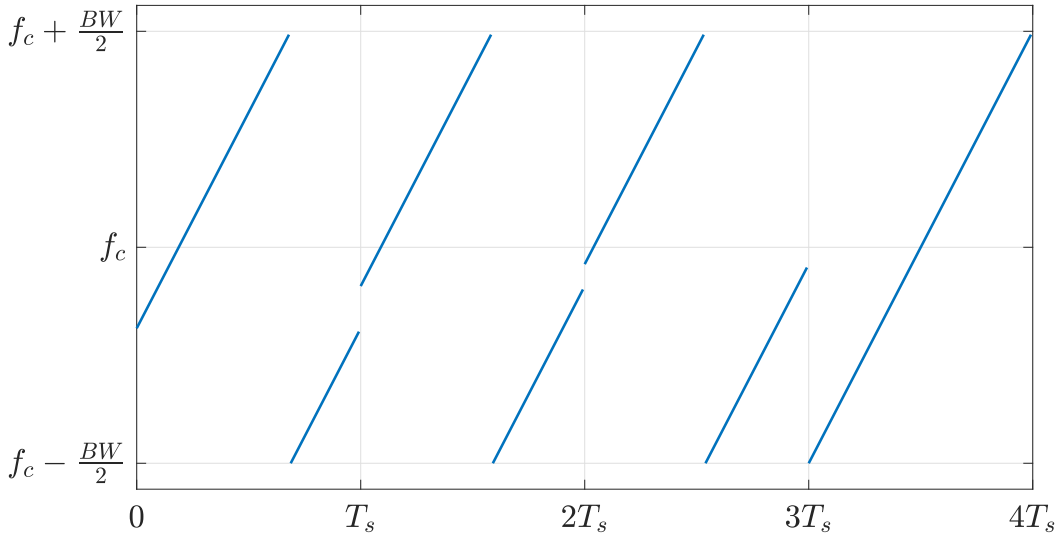
Zde je doba datového symbolu rozdělena na menší segmenty a podle pseudonáhodné posloupnosti je vždy v symbolové době vybrán jeden segment pro vysílání symbolu, ve zbytku segmentů se nevysílá. Tímto procesem je zrychlen přenos dat, ovšem jen po dobu časového segmentu, takže symbolová rychlost zůstane nezměněná oproti ryhclosti původních dat. Zrychlení přenosu má za následek rozšíření spektra signálu.

CSS - Chirp Spread Spectrum

V této technice je užito signálu s lineárně se měnícím kmitočtem, tzv. *chirpem*, kde rozdíl mezi jeho nejvyšším a nejnižším kmitočtem je šířka pásma signálu. Na takovýto typ signálu jsou poté modulována data. CSS byla vyvinuta především pro oblast radarové techniky.

1.1.2 Chirp Spread Spectrum v LoRa

Jak bylo zmíněno, standard LoRa ke své činnosti využívá modulační metodu rozprostřeného spektra CSS. Modulace je založena na vysílání signálu s lineární časovou závislostí kmitočtu, který se nazývá chirp. V LoRa mají využití jak chirpy s postupně rostoucím kmitočtem (up-chirp), tak i s klesajícím kmitočtem (down-chirp). Každý jeden chirp přenáší jeden symbol, jehož bitová délka je určena parametrem *SF*, viz 1.1.3. Pro namodulování dat na chirp je celá šířka pásma rozdělena na 2^{SF} částí, takzvaných chipů. Data se pak v chirpu projeví jako frekvenční pozice prvního chipu v symbolu. Pro názornost je tento proces zobrazen na 1.1. Časová osa je rozdělena do čtyř částí, každá o trvání symbolové doby T_s, tedy doby, za kterou se přenese jeden chirp. Poslední chirp (od $3T_\mathrm{s}$ do $4T_\mathrm{s}$) přenáší symbol o hodnotě 0. Počáteční kmitočet takového signálu roste od minimálního kmitočtu $f_\mathrm{c} - \frac{BW}{2}$ do $f_\mathrm{c} + \frac{BW}{2}$ bez žádné nespojitosti tak, jako je tomu u ostatních hodnot symbolů. V případě například prvního symbolu je počáteční kmitočet vyšší než minimální $f_\mathrm{c} - \frac{BW}{2}$. Jelikož je symbolová doba pro všechny symboly stejná, tak signál dosáhne hodnoty maximálního kmitočtu $f_\mathrm{c} + \frac{BW}{2}$ dříve než v čase T_s. Jakmile je dosaženo maximálního kmitočtu, stejnou strmostí roste kmitočet dále od hodnoty minimálního kmitočtu, jak je z obrázku patrné.



Obrázek 1.1: Časový průběh kmitočtu signálu LoRa

1.1.3 Parametry modulace LoRa

Běžně se sítě LoRa provozují v bezlicenčních kmitočtových pásmech ISM, lišících se mezi kontinenty. Typicky se jedná o kmitočty 868 MHz, 915 MHz, 433 MHz. Celou modulaci popisují tři parametry: *SF*, *CR* a *BW*.

BW - Šířka pásma

Šířka pásma *BW* popisuje, jak široké intervaly frekvencí lze pro modulaci použít. Typicky jsou to hodnoty 125 kHz, 250 kHz a 500 kHz, nicméně existují implementace i s nižšími *BW*. Se zvyšující se šířkou pásma přímo úměrně roste přenosová rychlost komunikace.

SF - Spreading factor

Parametr *SF* popisuje, na kolik chipů bude rozdělen jeden chirp, což mimo jiné říká, kolik bitů bude reprezentovat jeden symbol. Se zvyšujícím se *SF* klesá přenosová rychlost, avšak komunikace je odolnější vůči rušení. Důležitým faktem je, že dva signály na stejném kmitočtu ale s různým *SF* jsou vzájemně ortogonální a tudíž nedojde k vzájemnému rušení při současném vysílání více signálů. V LoRa může parametr *SF* nabývat hodnot $\{6, 7, 8, 9, 10, 11, 12\}$.

CR - Coding rate

CR udává, jaká část z přenesené informace je redundantní, sloužící k zajištění větší spolehlivosti komunikace. Parametr *CR* nabývá hodnot z $\{1, 2, 3, 4\}$, toto číslo se vztahuje vždy ke čtveřici bitů (nibble). Užitečná informace je tedy pouze $\frac{4}{4+CR}$ přenášené informace, zbytek jsou paritní bity získané Hammingovým kódováním.

1.1.4 Matematický popis signálu

Nejprve bude uveden vztah pro nemodulovaný up-chirp (poslední chirp na předchozím obrázku). Časový vývoj kmitočtu takového signálu je pouhou úsečkou a tedy pro něj platí:

$$f(t) = \frac{BW}{T_\mathrm{s}}t - \frac{BW}{2} \Longrightarrow \varphi(t) = 2\pi \int_0^t f(\tau)\,d\tau = 2\pi\left(\frac{BW}{2T_\mathrm{s}}t^2 - \frac{BW}{2}t\right), \quad 0 \leq t < T_\mathrm{s}, \tag{1.2}$$

kde $\varphi(t)$ je fáze signálu. Signál se nachází v základním pásmu.

Ze znalosti fáze již lze uvést vztah pro up-chirp.

$$x_\mathrm{up}(t) = e^{j\varphi(t)} = e^{j2\phi(\frac{BW}{2T_\mathrm{s}}t^2 - \frac{BW}{2}t)}. \tag{1.3}$$

Komplexní signál představuje spojení dvou složek $I = \mathrm{Re}\{x_\mathrm{up}(t)\}$ a $Q = \mathrm{Im}\{x_\mathrm{up}(t)\}$.

Po zakomponování hodnoty symbolu do chirpu se vztah trochu komplikuje, podle [3] takový signál vypadá následovně:

$$x_\mathrm{S}(t) = \begin{cases} e^{j2\pi(\frac{BW}{2T_\mathrm{s}}t^2 + (\frac{BW}{2^{SF}}S - \frac{BW}{2})t)}, & 0 \leq t < t_\mathrm{fold}, \\ e^{j2\pi(\frac{BW}{2T_\mathrm{s}}t^2 + (\frac{BW}{2^{SF}}S - \frac{3BW}{2})t)}, & t_\mathrm{fold} \leq t < T_\mathrm{s}. \end{cases} \tag{1.4}$$

Člen t_fold představuje čas, ve kterém je dosaženo kmitočtu $\frac{BW}{2}$ a platí $t_\mathrm{fold} = T_\mathrm{s}\frac{2^{SF}-S}{2^{SF}}$. S je číslo z $\{0, 1, 2, 3, ..., 2^{SF} - 1\}$.

Při úvaze, že doba trvání symbolu $T_\mathrm{s} = 2^{SF}T_\mathrm{chip}$ a doba trvání chipu $T_\mathrm{chip} = \frac{1}{BW}$, dostaneme vztah $T_\mathrm{s} = \frac{2^{SF}}{BW}$. Jelikož nás zajímá digitální podoba signálu, je nutné vyjít z navzorkovaného signálu 1.4. Pro navzorkování stačí říci, že $t = nT_\mathrm{samp} = \frac{n}{f_\mathrm{samp}}$, kde $n \in \mathbb{N}_0$. Při výše zmíněném dostaneme novou podobu signálu:

$$x_\mathrm{S}[n] = \begin{cases} e^{j2\pi(\frac{BW^2}{{f_\mathrm{samp}}^2 2\cdot 2^{SF}}n^2 + (\frac{S}{2^{SF}} - \frac{1}{2})\frac{BW}{f_\mathrm{samp}}n)}, & 0 \leq n < n_\mathrm{fold}, \\ e^{j2\pi(\frac{BW^2}{{f_\mathrm{samp}}^2 2\cdot 2^{SF}}n^2 + (\frac{S}{2^{SF}} - \frac{3}{2})\frac{BW}{f_\mathrm{samp}}n)}, & n_\mathrm{fold} \leq n < 2^{SF}. \end{cases} \tag{1.5}$$

Stanovením vhodné hodnoty f_samp je možno docílit jednoduššího tvaru 1.5. Takovou hodnotou je BW. Takový vzorkovací kmitočet neodporuje Shannon-Nyquestově vzorkovací podmínce, ježto nejvyšší kmitočet v signálu nabývá právě hodnoty $\frac{BW}{2}$.

$$x_\mathrm{S}[n] = \begin{cases} e^{j2\pi(\frac{n^2}{2\cdot 2^{SF}} + \frac{S}{2^{SF}}) - j2\pi\frac{1}{2}n}, & 0 \leq n < n_\mathrm{fold}, \\ e^{j2\pi(\frac{n^2}{2\cdot 2^{SF}} + \frac{S}{2^{SF}}) - j2\pi\frac{3}{2}n}, & n_\mathrm{fold} \leq n < 2^{SF}. \end{cases} \tag{1.6}$$

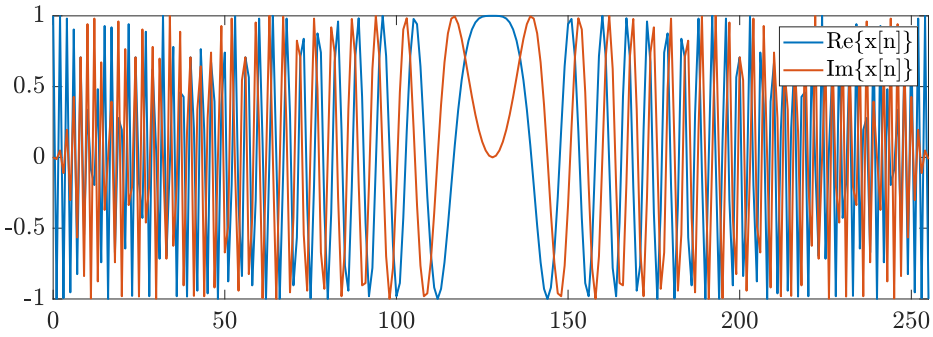
V posledním vztahu je vidět, že fáze v druhém případě ($n > n_\mathrm{fold}$) je posunutá o $-2\pi n$ oproti prvnímu případu ($n < n_\mathrm{fold}$). Protože je tento posun vždy celočíselným násobkem 2π, můžeme napsat výsledný vztah pro navzorkovaný modulovaný chirp vzorkovacím kmitočtem $f_\mathrm{samp} = BW$:

$$x_\mathrm{S}[n] = e^{j2\pi(\frac{n^2}{2\cdot 2^{SF}} + (\frac{S}{2^{SF}} - \frac{1}{2})n)}, \quad n \in \{0, 1, 2, 3, 4, ..., 2^{SF} - 1\}. \tag{1.7}$$

Na obrázku 1.2 je zobrazen vygenerovaný up-chirp v časové oblasti podle vztahu 1.7.

Na závěr ještě stojí za povšimnutí vztah mezi přenosovou rychlostí, šířkou pásma a SF. Pro symbolovou dobu platí $T_\mathrm{s} = \frac{2^{SF}}{BW}$. Za symbolovou dobu je přeneseno právě SF bitů, tedy přenosová rychlost je:

$$C = \frac{SF}{T_\mathrm{s}} = \frac{SF}{2^{SF}}BW. \tag{1.8}$$



Obrázek 1.2: Časový průběh up-chirpu s $SF = 8$

Ze vztahu je patrné, jak s rostoucím SF klesá přenosová rychlost, a naopak s rostoucím BW se přenosová rychlost úměrně zvyšuje.

1.1.5 Demodulace

Při příjmu přijímač detekuje signál ve tvaru 1.7. Pro zjištění hodnoty symbolu je zapotřebí vypreparovat hodnotu S v uvedeném vztahu pro přijatý signál. Toho je docíleno vynásobením přijatého chirpu jedním nemodulovaným down-chirpem, tím je získáno:

$$e^{j2\pi(\frac{n^2}{2\cdot 2^{SF}}+(\frac{S}{2^{SF}}-\frac{1}{2})n)}e^{-j2\pi(\frac{n^2}{2\cdot 2^{SF}}-\frac{n}{2})} = e^{j2\pi\frac{S}{2^{SF}}n}. \tag{1.9}$$

Výsledek ve vztahu 1.9 odpovídá posloupnosti s jedním kmitočtem, který je úměrný symbolové hodnotě S. Tudíž hodnota S je získána aplikací diskrétní Fourierovy tranformace na takový signál.

$$X_{\mathrm{k}} = \mathrm{DFT}\{e^{j2\pi\frac{S}{2^{SF}}n}\} = \frac{1}{2^{SF}}\sum_{n=0}^{2^{SF}-1} e^{j2\pi\frac{S}{2^{SF}}n}e^{-j2\pi\frac{k}{2^{SF}}n} = \begin{cases}1, & k = S\\ 0, & k \neq S\end{cases} \tag{1.10}$$

Maximální hodnota posloupnosti X_{k} je tedy na indexu k, který odpovídá právě symbolové hodnotě S. S tímto lze vyjevit vztah pro demodulaci chirpu:

$$S = \mathrm{argmax}\{\mathrm{DFT}\{x[n]x_{\mathrm{down}}[n]\}\} \tag{1.11}$$

Existují i jiné způsoby demodulace, například v článku [4].

1.1.6 Formát komunikace v LoRa

Doposud byl popsán princip nejnižší vrstvy komunikace – fyzické. LoRa nicméně také stanovuje formát, ve kterém má komunikace probíhat, tedy operace nad samotnými daty pro zajištění co možná nejvyšší možné spolehlivosti a tvar rámců jakožto nejmenších datových entit v síti.

Pro větší spolehlivost přenosu dat, je na datech aplikován řetězec operací, známý též pod pojmem *Forward Error Correction* (FEC). Pro zakódování se jedná o následující operace v uvedeném pořadí:

- Hammingovo kódování
- Whitening

- prokládání (interleaving)
- Grayovo kódování

Pro dekódování je sled operací obrácen, ovšem s inverzními funkcemi. Nyní budou zevrubně popsány jednotlivé operace s pomocí zdrojů [5], [6], [4].

1.1.7 Hammingův kód

Jedná se lineární kód, který může za jistých okolností detekovat či dokonce opravit chybně přenesené bity během komunikace. Klíčovým parametrem je zde parametr CR (viz 1.1.3). Ke každé čtveřici datových bitů je na základě generující matice vypočítáno CR paritních bitů a následně připojeno k datům.

Zásadní prvek v Hammingově kódu je generující matice. Je to matice $4 \times (4 + CR)$, která po vynásobení datového vektoru zleva vygeneruje kódové slovo. Celý proces je pro $CR = 4$ znázorněn v 1.12.

$$\underbrace{\begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix}}_{\vec{d}} \underbrace{\begin{pmatrix}1 & 1 & 1 & 0 & 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1\\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1\\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0\end{pmatrix}}_{\mathbb{G}_{4/8}} = \underbrace{\begin{pmatrix}1 & 0 & 1 & 0 & 1 & 0 & 1 & 0\end{pmatrix}}_{\vec{c}} \tag{1.12}$$

Do matice $\mathbb{G}_{4/8}$ vstupuje zleva datové slovo $\vec{d}$. Kódové slovo $\vec{c}$ je pak tvořeno lineární kombinací řádků matice $\mathbb{G}_{4/8}$, které maskuje vektor $\vec{d}$. V uvedeném příkladu je $\vec{c}$ tvořen součtem 1., 2. a 4. řádku generující matice ($\vec{d}$ má na uvedených pozicích jedničky, jinde nuly). Řádky generující matice jsou lineárně nezávislé a tvoří bázi podprostoru kódových slov v prostoru $(\mathbb{Z}_2)^n$ (v příkladu 1.12 je $n = 8$).

Při příjmu kódového slova je nejprve pomocí kontrolní matice vypočítán takzvaný syndrom $\vec{s}$ Kontrolní matice je složena z řádků, které generují ortogonální doplněk k prostoru kódových slov, a platí že: $\mathbb{G}^T\mathbb{H} = \mathbb{O}$, kde $\mathbb{H}$ je kontrolní matice. Jestliže bylo kódové slovo přijato bezchybně, pak bude syndrom $\vec{s} = \mathbb{H}\vec{c}^T$ roven nulovému vektoru. Vyjde-li syndrom nenulový, došlo k chybě během přenosu. Podle hodnot paritních bitů je poté zjistitelné, na jakých pozicích došlo k chybě při přenosu a je možnost i chybu opravit. Pro hodnoty CR v LoRa je možné opravit maximálně 1 chybu, a odhalit maximálně 3 chyby ($CR = 4$). Více o Hammingově kódu lze nalézt například v [7].

1.1.8 Whitening

Technika, která z dat odstraňuje dlouhé úseky jedniček nebo nul – jde o odstranění datové 'DC složky". Odstranění je docíleno aplikováním tzv. whitening sekvence na data pomocí operace exklusivního součtu - XOR. Tato sekvence je jiná pro různé CR a různé čipy. V [4] je dokumentováno, jak byla získána whitening sekvence pro konkrétní čip SX1272.

1.1.9 Prokládání

Pomocí prokládání se společně s Hammingovým kódem cílí na větší odolnost vůči rušení během přenosu. Zvyšuje pravděpodobnost možné opravy naskytlé chyby.

Myšlenkou je, vytvořit z datových slov nová prokládaná slova, kde každé jedno prokládané slovo obsahuje určitou informaci ze všech datových slov. Dosahuje se toho tak, že například druhé prokládané slovo je složeno ze všech druhých bitů datových slov.

V LoRa je tento proces implementován následovně. Nejprve se seřadí právě SF datových slov pod sebe a utvoří takzvanou prokládající matici. Datová slova čítají $4 + CR$ bitů, tedy matice nabývá rozměrů $SF \times (4 + CR)$. Poté jsou jednotlivé sloupce orotovány směrem dolů (směrem k LSB) o počet pozic vycházející z indexu sloupce v matici, tedy třetí sloupec bude rotován o tři pozice směrem k LSB. Po transponování takto vzniklé matice tvoří její řádky již výsledné hodnoty prokládání. Řádky transponované matice mají SF sloupců, tedy prokládaná slova mjí SF bitů. Proces prokládání je znázorněn v 1.13 pro parametry $SF = 7$, $CR = 4$, řádků není sedm z toho důvodu, že se jedná o případ prokládání hlavičky v LoRa rámci která je poněkud specifická.

$$\begin{matrix} 0&0&0&0&1&0&0&1 \\ 0&0&0&1&0&1&1&0 \\ 0&0&1&0&1&1&0&1 \\ 0&1&0&0&0&0&1&0 \\ 1&0&1&1&0&1&0&1 \end{matrix} \longrightarrow \begin{matrix} 0&0&0&0&0&0&0&0 \\ 0&0&1&0&1&1&0&1 \\ 0&0&0&1&0&1&1&1 \\ 0&0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1 \end{matrix} \longrightarrow \begin{matrix} 0&0&0&0&1 \\ 0&0&0&0&1 \\ 0&1&0&0&1 \\ 0&0&1&0&1 \\ 0&1&0&0&1 \\ 0&1&1&0&1 \\ 0&0&1&0&1 \\ 0&1&1&0&1 \end{matrix} \tag{1.13}$$

Jeví se jako výhodné spojit Hammingův kód s prokládáním. Dojde-li k porušení jednoho symbolu, pak dojde k porušení jednoho bitu ve všech datových slovech (v nejhorším případě), což je situace která se dá díky použitému Hammingově kódu napravit a ke ztrátě dat nedojde.

1.1.10 Grayův kód

Jedná se o kód s masivním uplatněním v celé radiotechnice. Jeho klíčovou vlastností je, že každá dvě kódová slova se od sebe liší vždy právě v jednom bitu. Stejně tak v konstelačním diagramu jsou body indexovány Grayovým kódem takovým způsobem„ že všechny sousední body určitého symbolu se liší v jednom bitu. Když se přijatý symbol interpretuje místo vysílaného na jeden z jeho bezprostředního okolí, je zaručeno, že se přijatá hodnota liší jen v jednom bitu. Takovou poruchu lze mnohdy opravit. Tabulka 1.1 pro ukázku znázorňuje tříbitový Grayův kód. V protokolu LoRa se kódují slova o délce SF bitů, tedy 6 b až 12 b.

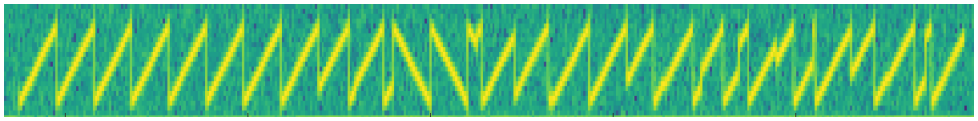
1.1.11 Tvar LoRa rámce

Na obrázku 1.3 je spektrogram reálného signálu LoRa ($SF = 8$, $CR = 4$ a $BW = 125$ kHz), na kterém jsou patrné jednotlivé části rámci v komunikaci LoRa. Prvních osm symbolů jsou nemodulované up-chirpy které slouží jako preambule k inicializaci komunikace, nicméně délka preambule je volitelným parametrem. Následují dva modulované up-chirpy nesoucí synchronizační slovo, značící pod jakou síť rámec spadá. V LoRaWAN síti je to typicky hodnota 0x34, v privátních sítích se hodnoty liší. Pokud přijímač detekuje příchozí rámec s hodnotou synchronizačního slova jinou, než do které sám spadá, tak dále komunikaci ignoruje. Další v pořadí

Tabulka 1.1: Tříbitový Grayův kód

bin	Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

jsou dva nemodulované down-chirpy a jedna čtvrtina down-chirpu. Tato část slouží k frekvenční synchronizaci přijímače. Dále již následuje datové pole.



Obrázek 1.3: Spektrogram zachyceného signálu LoRa s parametry $BW = 125$ kHz, $SF = 8$ a $CR = 4$

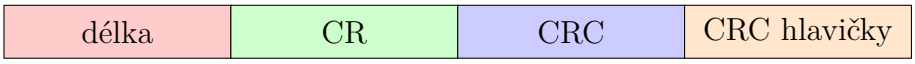
Formát datového pole se skládá ze tří částí:

- Hlavička
- Data
- CRC dat

Kromě dat, nejsou ostatní části povinné. Pole Data je datová posloupnost, na kterou byl aplikován řetězec operací pro FEC, CRC je pouhý kontrolní součet o délce 16 bitů, jehož implementace je k nalezení v [8]. Z uvedených částí bude dále popsána hlavička.

Hlavička

Je-li nastavena komunikace s hlavičkou, tak nehledě na nastavenou hodnotu CR je použité CR pro hlavičku vždy 4. To zaručuje co nejspolehlivější přenos metadat k samotným datům. Formát hlavičky je na obrázku 1.4.



Obrázek 1.4: Formát hlavičky v protokolu LoRa

- **délka** - Pole o délce 8 bitů. Ukládá informaci o délce dat v bytech.
- **CR** - Pole o délce 3 bitů. Nese informaci o použitém parametru CR aplikovaném na data.
- **CRC** - Jeden bit. Je-li v jedničce, k datům je připojen i kontrolní součet.

- **CRC hlavičky** - Kontrolní součet hlavičky o délce 8 bitů.

Celkem je hlavička tvořena 20 bity, po Hammingově kódování se velikost zdvojnásobí na 40 bitů. Hlavičku tvoří vždy $SF - 2$ symbolů, což odpovídá bloku $(SF - 2)SF$ bitů. Nevyužité místo se zaplní daty z datového pole a zakóduje společne s hlavičkou tak, aby byly všechny symboly zaplněny.

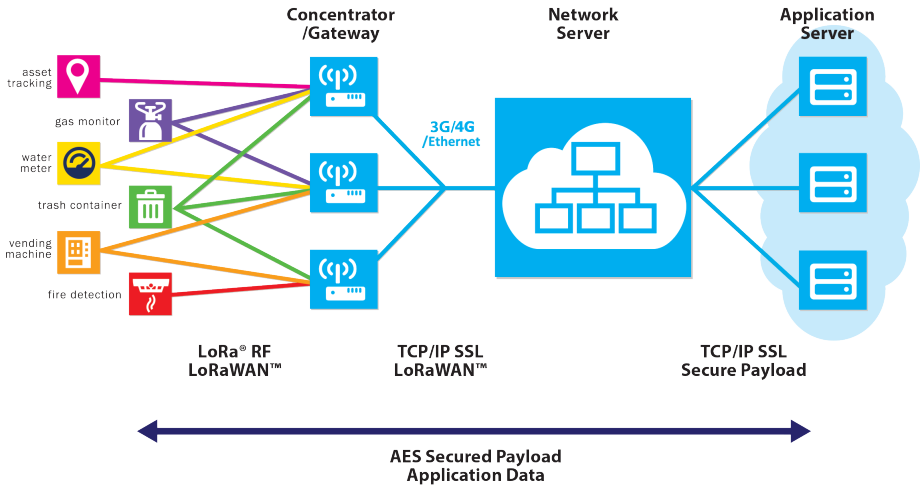
1.2 LoRaWAN

Zatímco komunikační protokol LoRa je proprietární technologií firmy Semtech, tak sítě typu LoRaWAN představují otevřený standard přístupové vrstvy nad protokolem LoRa. Tento standard zaštiťuje sdružení LoRa Alliance, čítající takové firmy jako IBM, Cisco, HP, Foxconn, a další.

LoRaWAN je linkovou vrstvou v ISO/OSI modelu nad protokolem LoRa. Typem sítě spadá pod LPWAN (Low Power Wide Area Network), které se vyznačují malou energetickou spotřebou. LoRaWAN definuje síť rozšířené hvězdicové topologie (Obrázek 1.5), ve které koncová zařízení komunikují protokolem LoRa s bránami. Brány takto sbírají data a přeposílají je například pomocí protokolu TCP/IP síťovému serveru, který komunikuje se serverem aplikačním, jenž určuje zpracování dat. Komunikace mezi koncovými zařízeními a bránami se dělí do časových oken, ve kterých lze komunikovat směrem buď k bráně do koncového zařízení, či obráceně, ale nikdy ne najednou. Z tohoto pohledu se jedná o half-duplexní komunikaci.

LoRaWAN také určuje konkrétní frekvence, na kterých bude komunikace probíhat. Každé koncové zařízení se pro každé vysílání náhodně přeladí na určitý komunikační kanál. Toto přelaďování předchází interferencím způsobených současným vysíláním více zařízení v jednom kanálu se shodným SF.

LoRaWAN popisuje její specifikace [9], [10], tudíž zde nebudou vyloženy všechny aspekty LoRaWAN, ale popis se zaměří především na základní principy.



Obrázek 1.5: Síť LoRaWAN

1.2.1 Koncová zařízení

Koncová zařízení jsou různé senzory či aktuátory schopné komunikovat v LoRaWAN síti. Každé zařízení je opatřeno 32 bitovou adresou zajištěnou síťovým serve-

rem pro identifikaci v síti. Formát paketů vysílaných koncovými zařízeními je popsán v uvedené specifikaci ([9]). LoRaWAN určuje celkem tři třídy koncových zařízení, lišících se především v pohotovosti komunikace a spotřebě zařízení.

Třída A

Třída, kterou musí podporovat všechna zařízení pro LoRaWAN. Vyznačuje se nejmenší spotřebou a absolutní asynchronicitou. Komunikaci zahajuje vždy koncové zařízení směrem k bráně (uplink). Poté následují dvě časová okna pro komunikaci směrem od brány ke koncovému zařízení (downlink). Server nikdy nemůže zahajovat komunikaci, tudíž pokud chce komunikovat s koncovým zařízením, musí od něj počkat na zprávu, a až poté všechny zprávy určené k odeslání poslat. Zařízení ve třídě A jsou z principu nejméně energeticky náročná, protože nemusí udržovat komunikaci a vysílají/přijímají jen když potřebují.

Třída B

Oproti třídě A je třída B synchronizována. Podle nastaveného plánu v určený čas do sítě posílá tzv. beacon rámce a tím otevírá přijímací časová okna. Síťový server tedy stále musí čekat na zahájení komunikace koncovým zařízením, ale doba po kterou je nucen vyčkávat, je pevně daná.

Třída C

Energeticky nejnáročnější třída. Po celou dobu kdy koncové zařízení nevysílá, tak poslouchá. Síťový server tak nemusí čekat na zahájení komunikace a kdykoliv, kdy koncové zařízení nevysílá, mu server může posílat data.

1.2.2 Gateway

Je zařízení v síti, které komunikuje s koncovými zařízeními, především od nich sbírá data a přeposílá je dále. Koncové zařízení nijakým způsobem nepatří k jedné gatewayi, když pošle data, všechny gatewaye v dostupném okolí paket přijmou a přepošlou dále. Takto například není problém mít v síti mobilní koncové uzly. Největší nároky na gateway jsou kladeny na rádiovou část a propustnost dat. Po přijetí dat z koncových uzlů gateway tato data přepošle pomocí rychlostního rozhraní (Ethernet, LTE, Wi-Fi) do síťového serveru. Gateway musí zvládat zpracovat data ze značného množství koncových uzlů a přeposílat rychlostní linkou data dále. Z tohoto důvodu je prakticky nutné, aby byla gateway připojena na nebateriové napájení.

1.2.3 Síťový server a aplikační server

Síťový server spravuje veškeré prostředky v síti LoRaWAN. Vyřizuje MAC příkazy docházející z koncových zařízení, nastavuje přenosovou rychlost a jiné. Je zodpovědný za přeposílání dat z uplink komunikace do aplikačního serveru, tak i za downlink komunikaci z aplikačního serveru ke koncovým zařízením. Pro downlink komunikaci je nutné přesné časování, protože pro downlink pakety jsou vyhrazena dvě časová okna po ukončeném vysílání koncového zařízení.

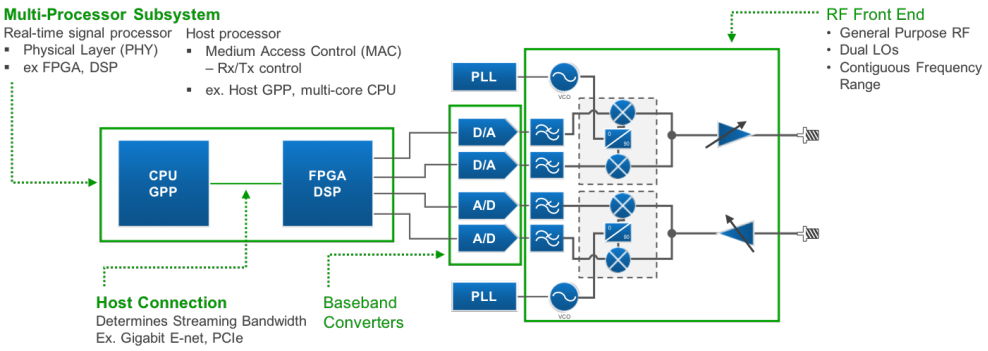
Aplikační server poté přijímá samotná data od síťového serveru a poskytuje aplikační služby uživatelům. Aplikačních serverů může být více. Do jakého aplikačního serveru poslat data je úkolem síťového serveru.

1.3 SDR

SDR je architektura rádiových zařízení určených pro příjem/vysílání, ve které jsou určité funkce implementovány jako programovatelný kód, nikoliv implementovány neměnně v hardwaru zařízení. Typicky jsou to funkce týkající se zpracování signálu, modulace, demodulace atp. Velkou výhodou je bezesporu univerzálnost takových zařízení. Stejným přijímačem lze dekódovat signál AM, FM, DVB-T, Wi-Fi či družicové vysílání, záleží jen na použitém programu. To ze softwarově definovaných rádií činí opravdu mocné nástroje s uplatněním v široké škále odvětví především v radioamatérství či radioastronomii. Následujících nekolik odstavců bude věnováno popisu SDR.

1.3.1 Koncepce SDR

Na obrázku 1.6 je zobrazeno principiální blokové schéma obecného SDR. Dále



Obrázek 1.6: Principiální schéma SDR [1]

budou popsány jednotlivé bloky.

Anténa

Zařízení sloužící k navázání elektromagnetických vln či k jejich emisi. Důležitým parametrem je zisk antény který říká, kolikrát větší energii je anténa schopná v určitém směru přijmout/vyslat než referenční anténa. Jedná se o typickou logaritmickou veličinu: $G = 10 \log \frac{P(\phi,\theta)}{P_R(\phi,\theta)}$, kde $P(\phi,\theta)$ představuje výkon vysílaný z antény, $P_R(\phi,\theta)$ je výkon vysílaný z referenční antény a (ϕ,θ) jsou úhly ve sférických souřadnicích. Referenční anténou je zpravidla izotropní zářič nebo ideální půlvlnný dipól. V případě izotropního zářiče jako referenční antény je jednotkou zisku dBi, v případě půlvlnného dipólu je jednotkou dBD [11]. Vztah mezi dBi a dBD je: 0 dBD = 2,15 dBi.

[1] `https://ni.scene7.com/is/image/ni/SDR_architecture_no_background?scl=1`

Up/Down konvertor

Obvody Up konvertoru i Down konvertoru jsou ve své podstatě obvodem směšovače. Směšovač je obvod se dvěma vstupy a jedním výstupem. Na vstupy je nejčastěji připojen signál a výstup lokálního oscilátoru. Na výstupu budou podle typu směšovače produkty o kmitočtech $mf_0 \pm nf_S$, kde f_0 je kmitočet lokálního oscilátoru, f_S je kmitočet signálu a m, n jsou celá čísla. Nicméně pro účely Up/Down konvertoru jsou žádoucí jen produkty $f_0 \pm f_S$, zbytek je třeba odfiltrovat. V případě sčítání kmitočtů se jedná o Up konvertor s výstupem na kmitočtu $f_0 + f_S$. Up konvertor je zařazen před vysílací anténou aby kmitočtově posunul výstupní signál ze základního pásma na kmitočet vhodný k vysílání. Down konvertor je zařazen naopak v cestě přijímače kde plní opačnou funkci, a sice dostat přijatý signál do základního pásma.

A/D a D/A převodník

A/D převodník (Analog-to-Digital converter) je obvod který z analogového spojitého signálu vytvoří číslicový signál, a to tím způsobem, že nejdříve analogový signál diskretizuje v čase pomocí vzorkovače, a poté aplikuje diskretizační proces i na hodnoty navzorkovaného signálu, čemuž se říká kvantování. Tyto dvě operace vnášejí do signálu nežádoucí složky a je na to nutné brát zřetel. Pro vzorkování je nutné dodržet Shannon-Nyquistův vzorkovací teorém, který stanovuje vzorkovací kmitočet nejméně dvakrát větší, než je nejvyšší kmitočet obsažený ve vzorkovaném signálu. Nedodržením této podmínky dojde k prolínání spektra, tzv. *aliasingu*. Druhým zdrojem nežádoucích složek je proces kvantování, při kterém dochází ke změnám hodnot vzorků, a tudíž kvantovaná posloupnost představuje jiný signál než nekvantovaná. To se projevý i ve spektru kvantovaného signálu tzv. kvantizačním šumem.

Zpracování digitálního signálu

Zpracování signálu zpravidla probíhá na PC. SDR je k PC připojeno přes sběrnici USB, PCI nebo jinou. Dále se již dá signál zpracovat jak programátor uzná za vhodné. Existují specializované programy typu Pothos nebo GNU Radio, jež se vyznačují tzv. *flowchart* prostředím, ve kterém se dají propojovat bloky které zajišťují různé operace prováděné nad signálem, např. filtry, převzorkovače, modulátory, demodulátory a mnoho dalších.

Kapitola 2

Praktická část

V následujících několika sekcích bude zevrubně popsána implementace protokolu LoRa na platformu LimeSDR Mini za pomoci softwaru GNU Radio.

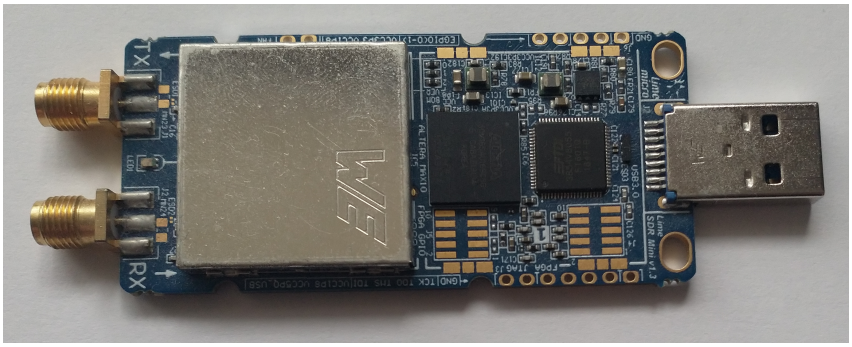
2.1 LimeSDR Mini

LimeSDR Mini (obrázek 2.1) je softwarově definované rádio firmy Lime microsystems. Jedná se o menší a levnější verzi LimeSDR.

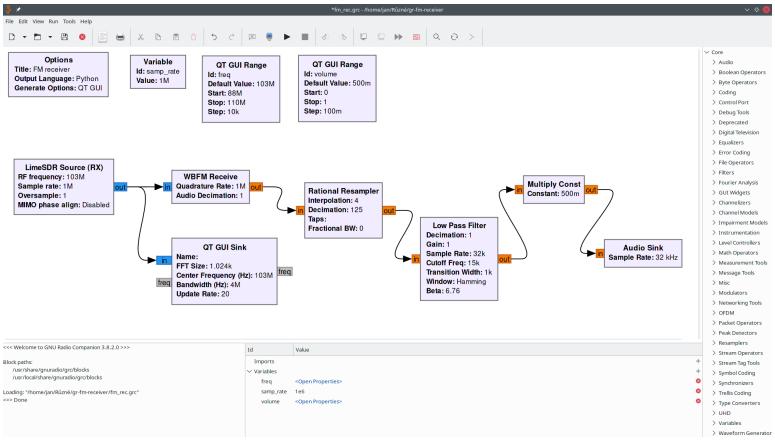
LimeSDR Mini disponuje dvěma rádiovými rozhraními, jedno pro přijímání, druhé pro vysílání. Na desce jsou tato rozhraní vyvedena na SMA konektory. Celou rádiovou část obsluhuje obvod LMS7002M [12]. Použitý obvod je použitelný ve škále kmitočtů od 100 kHz do 3,8 GHz.

2.2 GNU Radio

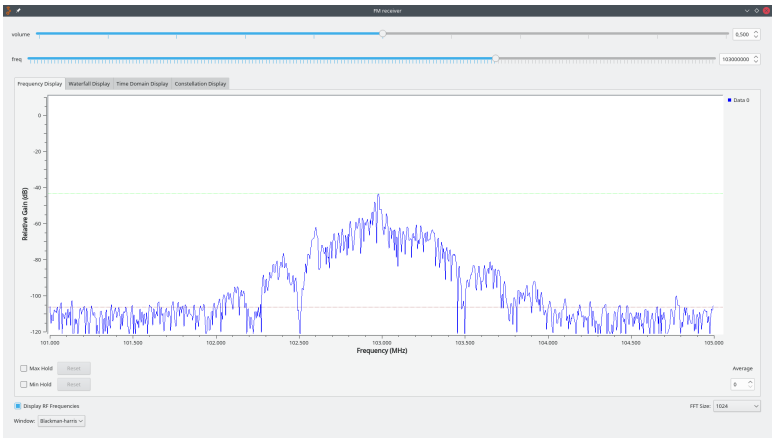
GNU Radio je software umožňující intuitivní implementaci rádiových přijímačů a vysílačů. Patří do široké rodiny svobodného softwaru pod licencí GNU General Public License. Je založeno na grafickém spojování bloků, které implementují konkrétní funkci v rádiovém zařízení. Spojováním takových bloků se dosahuje komplexnějších zapojení. Pospojované bloky s nastavenými parametry se nazývají flowgraph. GNU Radio pro takový flowgraph vygeneruje zdrojový kód ve skriptovacím jazyce Python. Kód v Pythonu je však jen textová podoba flowgraphu, samotné bloky jsou pak interně naprogramovány v jazyce C++ a z Pythonu volány díky nástroji SWIG.



Obrázek 2.1: LimeSDR Mini



Obrázek 2.2: flowgraph FM přijímače v GNU Radio



Obrázek 2.3: Program implementující FM přijímač

Pro ukázku je na obrázku 2.2 prostředí GNU Radia se zapojením jednoduchého FM přijímače. Spuštěný program je ukázán na obrázku 2.3.

2.2.1 OOT moduly

Pod označením OOT (Out Of Tree) modulů je rozumněno rozšiřování stávajících funkčních bloků v GNU Radio o nové moduly obsahující nové bloky. Název plyne ze skutečnosti, že nové bloky nejsou obsaženy v základním stromě modulů v GNU Radio, ale doinstalovány zvlášť.

K vytvoření nového modulu poskytuje GNU Radio nástroj *gr_modtool*, který slouží k vytvoření vývojového souborového stromu se zdrojovými kódy jednotlivých bloků v modulu, jejich grafickou reprezentaci, dokumentaci, soubory pro překlad a další. Jednotlivé bloky lze programovat v jazyce C++ nebo v jazyce Python. Obecně se dá říci, že pro aplikace kladoucí důraz na výkon a rychlost je vhodnější užití kompilovaného jazyka (C++) namísto jazyka interpretovaného (Python). Dále se bude jako vývojový jazyk uvažovat pouze C++.

2.3 Existující řešení

Pro komunikaci LoRa pomocí softwarově definovaného rádia již existují hotová řešení volně k použití, nicméně se často jedná o neúplné implementace, kdy je hotový pouze přijímač [4], nebo je řešení nekompatibilní s novějšími verzemi GNU Radia. Jsou i implementace pro jiný software, např. pro Pothos, které se však v rámci zadání práce nebudou uvažovat.

2.4 Návrh řešení

Cílem práce bylo vytvořit zařízení, které bude schopno obousměrné komunikace pomocí protokolu LoRa, tedy transceiver, na platformě LimeSDR Mini programovými prostředky softwaru GNU Radio. Takový transceiver čítá dvě části - přijímač a vysílač. GNU Radio poskytuje podporu pro požadované SDR – LimeSDR, existují bloky LimeSDR Source (Rx) a LimeSDR Sink (Tx). Blok LimeSDR Source (Rx) posílá navzorkovaný signál přijatý z antény do dalších bloků, naopak blok LimeSDR Sink (Tx) plní přesně opačnou funkci, a sice posílá vzorky do antény.

Ke splnění cíle je tedy třeba získat bloky pro funkce LoRa kodéru a dekodéru.

Dekodér

K zajištění funkce dekodéru bude využito již hotového LoRa modulu [1], který obsahuje funkční dekodér pro LoRa komunikaci. Umožňuje poslouchat na vícero kanálech současně. Podporuje všechny možné hodnoty parametrů SF, BW a CR.

Kodér

Pro funkci kodéru bude potřeba napsat vlastní OOT modul v programovacím jazyce C++. Analýzou kódu pro dekodér bude možné implementovat inverzní operace k operacím v dekodéru – Hammingovo kódování, whitening, prokládání a Grayovo kódování.

2.5 Implementace

Cílem bylo vytvořit blok, který na vstupu přijímá zprávy v řetězcovém tvaru, a na výstupu odesílá modulované chirpy do bloku LimeSDR Sink (Tx), který signál pošle fyzicky do SDR.

V prvním kroku byl třeba zjistit tvar jednotlivých operací aplikovaných na přijatá data ve zdrojovém kódu pro dekodér a také vidět výstup dekodéru v debbugovacím módu na uživatelský vstup. V dalším kroku byly po analýze vytvořeny inverzní operace k operacím v dekodéru a aplikována na nemodulovaná data určená pro přenos. Dále bude více popsána implementace každé jednotlivé operace.

Hammingovo kódování

Pro implementaci Hammingova kódování bylo v prvé řadě zapotřebí zjistit generující matici. Tuto matici lze získat pozorováním výstupu z přijímače po přijmutí

[1] Dostupný z: `https://github.com/rpp0/gr-lora`

zprávy obsahující v jednotlivých nibblech číslice 0x01, 0x02, 0x04 a 0x08. Výstupy na tyto zprávy (v debuggovacím módu) představují jednotlivé řádky generující matice.

Tímto pozorováním byly zjištěny následující tvary generujících matic pro jednotlivé kódovací poměry:

$$\mathbb{G}_{4/5} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0\text{x}30 \\ 0\text{x}18 \\ 0\text{x}14 \\ 0\text{x}12 \end{pmatrix} \tag{2.1}$$

$$\mathbb{G}_{4/7} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0\text{x}61 \\ 0\text{x}19 \\ 0\text{x}55 \\ 0\text{x}52 \end{pmatrix} \tag{2.2}$$

$$\mathbb{G}_{4/8} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0\text{x}e1 \\ 0\text{x}99 \\ 0\text{x}55 \\ 0\text{x}d2 \end{pmatrix} \tag{2.3}$$

Z tvaru generujících matic je vidět, že datové bity se nacházejí na pozicích 2., 3., 4., a 6. Ostatní pozice jsou, podle použitého CR, obsazeny paritními bity, či bity nevyužitými.

Nutno podotknout, že tvar matice pro kódovací poměr 4/5 je sice platný, nicméně první tři znaky (byty) byly přijímačem vždy chybně interpretovány. Pro zbytek znaků již tvar matice 2.1 sedí. Hodnota chybně vyložených znaků se v průběhu opakování vysílání stejné zprávy měnila a nebylo možné ji staticky vykompenzovat. Stejný problém se vyskytl i při určení generující matice pro kódovací poměr 4/6. Zde se ovšem chybně neinterpretovaly pouhé tři byty, nýbrž naprostá většina, i z relativně dlouhých zpráv (desítky znaků). Z tohoto důvody nebylo doposud možné vypozorovat tvar generující matice pro kódovací poměr 4/6. Pro ostatní kódovací poměry se žádné chybně detekované znaky s uvedenými maticemi (4/7 2.2 a 4/8 2.3) nevyskytují.

Se znalostí tvaru generující matice bylo možné vytvořit funkci pro Hammingovo kódování, ve které se podle vstupního nibblu operací XOR kombinovali řádky generující matice. Následuje příklad kódu pro zakódování čtveřice bitů (pro kódovací poměr 4/7).

```
uint8_t data_enc = ((data_nibble & 0x08) ? 0x61 : 0x00) ^
                   ((data_nibble & 0x04) ? 0x19 : 0x00) ^
                   ((data_nibble & 0x02) ? 0x55 : 0x00) ^
                   ((data_nibble & 0x01) ? 0x52 : 0x00);
```

Jednotlivé bity v datovém slově (data_nibble) jsou testovány na svoji hodnotu, podle které je ternárním operátorem (() ? :) rozhodnuto, které řádky generující matice budou použity k vygenerování kódového slova (data_enc).

Whitening

Pro tuto operaci bylo využito stejného postupu jako v dekodéru. Whitening není nic jiného, než pozměnění dat pomocí funkce XOR. Jako vstupy této funkce jsou samotná data k pozměnění a pseudonáhodná posloupnost. Jelikož výstup funkce exkluzivního součtu, jehož dva argumenty tvoří totéž číslo, je vždy nula (a XOR

$a = 0$), je možné využít stejný postup jak pro whitening, tak pro de-whitening. Při whitening je jednou aplikována pseudonáhodná posloupnost na data, při de-whitening je tato posloupnost aplikována podruhé, čímž se vyruší s první aplikací a výstupem jsou původní data.

K implementaci je tedy pouze zapotřebí znát tvar pseudonáhodné posloupnosti. Ve zdrojových souborech dekodéru se nachází soubor tables.h, který obsahuje whitening sekvence pro všechny možné hodnoty CR. Jejich získání je popsáno v [4]. Tentýž soubor je pak využit pro implementaci whitening v kodéru. Ukázka kódu pro whitening vypadá takto:

```
const uint8_t *whitening_seq = (d_cr == 3 || d_cr == 4) ?
                               prng_payload_cr78          :
                               prng_payload_cr56;
for (int i = 0; i < len; i++)
    data[i] ^= whitening_seq[i];
```

Podle hodnoty CR je ze souboru tables.h vybrána pseudonáhodná posloupnost (prng_payload_crXX), která je dále aplikována na datovou posloupnost (délky len).

Prokládání

Cíl prokládání spočívá ve vytvoření m n-bitových slov z n m-bitových slov tak, že každé prokládané slovo částečně obsahuje informaci ze všech původních slov. V případě LoRa je $m = SF$ ($m = SF - 2$ pro hlavičku či mód reduced rate) a $n = 4 + CR$. Graficky lze prokládání znázornit jako vytvoření matice, jejíž řádky tvoří SF datových slov o délce $4+CR$. Nově vzniklá slova po prokládání jsou sloupce takto vzniklé matice, nicméně je potřeba každé takové nové slovo orotovat doprava o k pozic, kde k je index sloupce v matici ($k = 0, 1, 2, ..., 3 + CR$). Z analýzy dekodéru taký vyšlo najevo, že před prokládáním je nutné přeházet některé bity v datových slovech. Tabulka 2.1 ukazuje, na které pozice se nově dostanou původní bity, kupříkladu ze slova 1101001 se stane slovo 11110000. Po této záměně pozic již lze aplikovat prokládání.

Tabulka 2.1: Záměna pozic bitů před prokládáním, pozice 1. značí LSB

původní pozice	nová pozice
1.	6.
2.	1.
3.	2.
4.	3.
5.	5.
6.	4.
7.	7.
8.	8.

Následující kód implementuje prokládání v kodéru. Pro účely kratšího kódu byl kód poněkud poupraven, veškeré operace typu prokladání, změna pozic bitů ve slově atd. jsou ve vlastním kódu kodéru implementovány jako samostatné funkce, zde pro krátkost jsou sloučeny do sebe. Z těchto důvodů občas chybí některé deklarace proměnných.

```
// Zamena pozic v datovych slovech
for (size_t i = 0; i < data_size; i++)
```

```
    data[i] = ((data[i] & 0x01) << 5) |
              ((data[i] & 0x02) >> 1) |
              ((data[i] & 0x04) >> 1) |
              ((data[i] & 0x08) >> 1) |
               (data[i] & 0x10)       |
              ((data[i] & 0x20) >> 2) |
               (data[i] & 0x40)       |
               (data[i] & 0x80);

// Prokladani
uint8_t b_count = (is_header) ? 8 : d_cr + 4;
uint8_t w_count = (is_header) ? d_sf - 2 : d_sf;

for (int i = 0; i < b_count; i++) {
    symbols[i] = 0;
    for (int j = 0; j < w_count; j++)
        symbols[i] |= ((data[j] & (1 << i)) >> i) << j;


    // Rotace doprava
    uint8_t n_shift = i % w_count;
    uint8_t n_bits = w_count;
    symbols[i] = ((symbols[i] >> n_shift) & ((1 << n_bits) - 1)) |
                 ((symbols[i] & ((1 << n_shift) - 1)) << (n_bits -
   n_shift));
}
```

Zpočátku je ve všech prvcích datové posloupnosti délky data_size provedena změna pozic bitů. Dále následuje prokládání. Podle proměnné hodnoty is_header (bool is_header) je stanoven počet řádků (uint8_t w_count) a sloupců (uint8_t b_count) matice. Platí totiž, že hlavička se vždy kóduje s $CR = 4$ ale počet slov k pokládání je pouze $SF - 2$, nehledě na nastavení parametrů pro data. Do proměnných symbols[i] (uint16_t symbols[]) jsou pak ukládána slova po prokládání. Tato slova je nakonec potřeba ještě orotovat jak je patrné z kódu.

Grayovo kódování

V protokolu LoRa se při kódování zprávy ve skutečnosti používá dekódování Grayova kódu, nikoliv kódování které je použito u LoRa dekodéru. Algoritmus je prostý, pro n-bitový symbol je třeba $n - 1$ kroků, při kterých se původní symbol operací XOR sečte se stejným symbolem, bitově posunutým o k pozic doprava (k je proměnná v cyklu, $0 < k <= n - 1$). Nicméně u kódování hlavičky je drobný rozdíl oproti zbytku, a sice ten, že hodnoty se po Grayově kódování musí vynásobit čtyřmi (bitový posun doleva o dva bity). Výstupem jsou již symboly které budou následně modulovat referenční upchirp. Následující kód ukazuje realizaci Grayova dekódování v LoRa kodéru.

```
uint16_t j;
uint16_t result;

for (int i = 0; i < symbols_size; i++) {
    j = symbols[j];
    result = 0;

    while (j != 0) {
        result ^= j;
        j >>= 1;
```

```
11      }
12
13      symbols[i] = result;
14
15      if (is_header)
16          symbols[i] <<= 2;
17  }
```

Význam pole symbols[] je tentýž, jako v předchozím odstavci o prokládání.

Modulace

Symboly určené k přenosu jsou přenášeny prostřednictvím chirpů. K tomu je nutné vygenerovat referenční up-chirp resp. down-chirp, který představuje symbol o hodnotě 0, tedy signál s rovnoměrně rostoucím resp. klesajícím kmitočtem od $-\frac{BW}{2}$ do $\frac{BW}{2}$. Tento referenční signál je následně modulován jednotlivými symboly tak, že hodnota na indexu 0 modulovaného chirpu (počáteční hodnota) odpovídá hodnotě na indexu *symbol* referenční chirpu. Další hodnoty kopírují toto schéma, nejlépe to vystihne následující zápis:

$$x_S[k] = \begin{cases} x_{\text{ref}}[k+S], & k = 0, 1, 2, ..., 2^{SF} - 1 - S, \\ x_{\text{ref}}[k - (2^{SF} - S)], & k = 2^{SF} - S, ..., 2^{SF} - 1. \end{cases} \tag{2.4}$$

Člen S je hodnota přenášeného symbolu, x_S je modulovaný chirp, x_{ref} představuje referenční chirp.

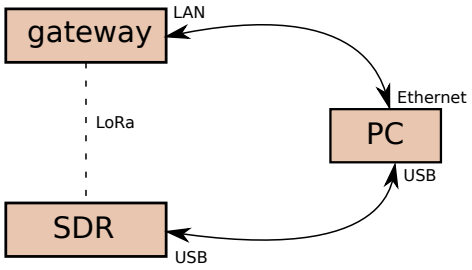
Referenční chirp je vygenerován podle podle vzorce 1.7 jenž představuje up-chirp, pro vygenerování down-chirpu stačí vzít komplexně sdruženou posloupnost k posloupnosti up-chirpu. Následující část kódu znázorňuje generování referenčního signálu a následnou modulaci taohoto signálu symbolem.

```
1  // Vygenerovani referencnich signalu - up-chirpu a down-chirpu
2  for (int n = 0; n < d_nfft; n++) {
3      angle = M_PI * (double(n*n)/d_nfft - n);
4      d_up_chirp.push_back(gr_complex(std::polar(1.0, angle)));
5      d_down_chirp.push_back(gr_complex(std::polar(1.0, -angle)));
6  }
7
8  // Priklad modulace payload na up-chirpy
9  for (int n = 0; n < paylen; n++)
10     for (int i = 0; i < d_nfft; i++)
11         d_modulated_symbols.push_back(d_up_chirp[(payload[n] + i) %
       d_nfft]);
```

Proměnná angle (double angle) představuje fázi signálu, d_nfft (uint16_t d_nfft) je počet prvků jednoho chirpu (2^{SF}), d_up_chirp (std::vector<double> d_up_chirp) je posloupnost referenčního up-chirpu (d_down_chirp posloupnost pro referenční down-chirp). Posloupnost d_modulated_symbols už obsahuje navzorkovaný signál určený k vysílání. V příkladu je do ní uloženo *paylen* symbolů. Z príkladu je také vidět, že vztah 2.4 je elegantně splněn použitím operace modulo.

2.6 Ověření funkce

K ověření funkce poslouží LimeSDR Mini a LoRa gateway.



Obrázek 2.4: Propojení jednotlivých prvků v testovacím zapojení

2.6.1 LoRa gateway

Jako testovací zařízení byla použita LoRa gateway LG01-N od firmy Dragino, pracující v pásmu 868 MHz. Gateway skýtá jisté nedostatky týkající se specifikace LoRaWAN, komunikace může probíhat pouze na jednom kanálu. LoRaWAN vyžaduje podporu osmi kanálů, na kterých se jednotlivá koncová zařízení střídají (viz [13]). Proto je vhodné tuto gateway provozovat jen ve vlastní, nenáročné síti.

Zvolená gateway běží na operačním systému OpenWrt, což je linuxová distribuce určená pro směrovací zařízení (název pochází z Open Wireless router). Takový systém je mocným nástrojem poskytující běh všemožných linuxových nástrojů a vlastních skriptů. Gateway může přijatá data z koncových zařízení LoRa odesílat pomocí protokolů TCP/IP či MQTT dále na síťový server, nebo může data zpracovat na základě uživatelského skriptu.

Gateway disponuje rozhraním Wi-Fi, LoRa, USB a dvěma (LAN, WAN) ethernetovými porty. Rozhraní pro LoRa zajišťuje čip Semtech SX1276 pro verzi gateway pracující v pásmu 868 MHz.

Aby gateway rozlišila jednotlivé klienty, je nutné opatřit každého z nich tzv. channel ID. Klient zařadí toto ID před každou zprávu ve formátu <channel_id>zpráva.

2.6.2 Testovací zapojení prvků

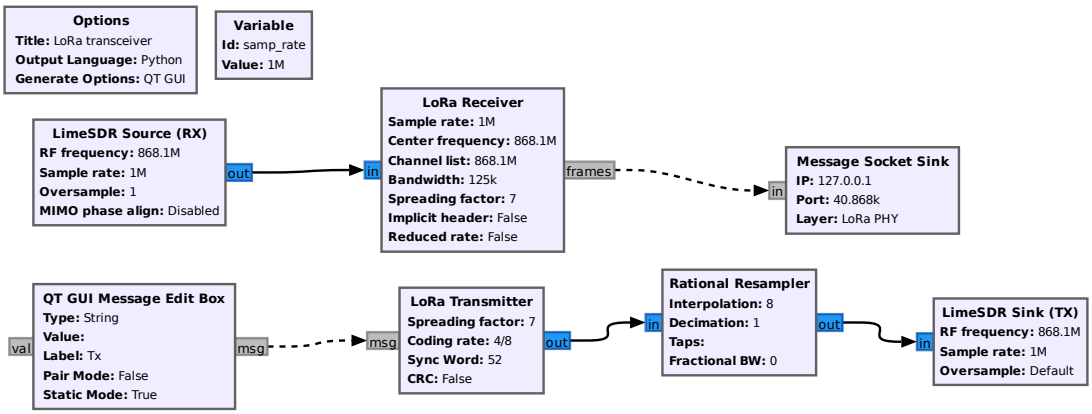
Obrázek 2.4 ukazuje použité zapojení SDR, gateway a PC. PC je přes přes ethernet připojen do LAN rozhraní na gateway. SDR je připojeno přes USB do PC, kde je ovládáno programem v GNU Radio. SDR a gateway spolu mohou komunikovat pouze bezdrátově pomocí LoRa.

2.6.3 Testovací program

Za účelem ověření komunikace bylo potřeba sestrojit program v GNU Radio obsluhující SDR a zároveň nastavit zpracování dat na gateway.

Program obsluhující SDR

Na obrázku 2.5 je zobrazeno zapojení bloků v GNU Radio implementující LoRa vysílač a přijímač. Blok pro demodulaci LoRa rámců (LoRa Receiver) je napojen přímo na výstup z přijímací antény (LimeSDR Source (RX)). Výstup z demodulátoru je napojen na blok "Message Socket Sink", který zobrazí přijaté zprávy a pošle je na loopback rozhraní. Na straně vysílače se jako první blok nachází "QT GUI Message Edit Box", který vytvoří grafické okno s polem pro zadání textu, která se odešle po stisknutí klávesy Enter. Tuto textovou zprávu převezme blok pro LoRa



Obrázek 2.5: Flowgraph testovacího zapojení

modulaci, který, dle zadaných parametrů, vytvoří navzorkovaný modulovaný signál určený k vysílání. Po něm ještě následuje blok "Rational Resampler", díky kterému lze měnit šířku pásma modulovaného signálu. Z převzorkovače jsou vzorky již vysílány díky bloku LimeSDR Sink (Tx).

Obsluha na gateway

K příjmu dat na gateway poběží skript, který přijaté zprávy nahraje do databáze na serveru, který beží na testovacím PC. Na databázi přijatých dat bude možno pohlédnout skrze jednoduché webové rozhraní. k vysílání dat slouží vestavěný program *lg02_single_rx_tx*, jenž umožňuje vysílat textové zprávy s nastavitelnými parametry SF, CR, BW.

2.6.4 Testování komunikace

Testování přijímací části SDR

Zprávy byly vysílány z gateway příkazem *lg02_single_rx_tx* v režimu vysílání. Pro každou jednotlivou hodnotu SF se vyzkoušeli všechny hodnoty CR. Tento postup byl poté aplikován i pro tři hodnoty BW (125 KHz, 250 kHz a 500kHz). Ačkoliv tvůrci projektu *gr-lora* uvádí, že přijímač je prakticky na 100 % schopen dekódovat zprávy pro všechny SF, měření s gateway DRAGINO toto tvrzení nepotvrdilo. Na vině může být fakt, že primárně byl dekodér testován pro vysílač postavený na čipu Microchip RN2483, kdežto DRAGINO gateway využívá čip Semtech SX1276.

Při testování se $SF = 6$ a zaplou volbou "implicit header"došlo vždy k chybě "Segmentation fault". Chyba bude pravděpodobně způsobena faktem, že program počítá s hlavičkou o velikosti $SF - 2$ prvků a zároveň celá hlavička musí obsahovat 20 bitů (tvořené 5 čtveřicemi), které vyžadují 5 prvků v poli pro hlavičku. Tyto dvě podmínky jsou v rozporu, protože pro pole hlavičky je vyhrazeno pole o 4 prvcích.

Pro hodnoty $SF = 7$, 8, 9 a 10 byly zprávy přijaty bezchybně.

Se $SF = 11$ byly sice zprávy detekovány, ale po dekódování obsahovaly naprosto jiné údaje oproti vysílané zprávě. Pro $SF = 12$ nebyly zprávy detekovány.

Zprávy byly dekódovány pro všechny hodnoty BW i CR. Výsledky testování jsou shrnuty v tabulce 2.2, kde ANO znamené bezchybně přijatou zprávu na SDR, NE nikoli.

Tabulka 2.2: Test komunikace (Rx) v závislosti na parametrech CR a SF

	$CR = 1$	$CR = 2$	$CR = 3$	$CR = 4$
$SF = 6$	NE	NE	NE	NE
$SF = 7$	ANO	ANO	ANO	ANO
$SF = 8$	ANO	ANO	ANO	ANO
$SF = 9$	ANO	ANO	ANO	ANO
$SF = 10$	ANO	ANO	ANO	ANO
$SF = 11$	NE	NE	NE	NE
$SF = 12$	NE	NE	NE	NE

Tabulka 2.3: Test komunikace (Tx) v závislosti na parametrech CR a SF

	$CR = 1$	$CR = 2$	$CR = 3$	$CR = 4$
$SF = 6$	NE	NE	NE	NE
$SF = 7$	ANO	NE	ANO	ANO
$SF = 8$	ANO	NE	ANO	ANO
$SF = 9$	ANO	NE	ANO	ANO
$SF = 10$	ANO	NE	ANO	ANO
$SF = 11$	NE	NE	NE	NE
$SF = 12$	NE	NE	NE	NE

Testování vysílací části SDR

Ověření vysílací části LoRa SDR transceiveru proběhlo v následujících krocích. Postupovalo se od nejnižších možných hodnot paramterů CR, SF a BW k nejvyšším a pozorovalo se, zda se odeslana zpráva zobrazí v databázi na PC. Výsledky jsou patrné z tabulky 2.3, ve které ANO značí úspěšně přijatou zprávu jež vyslalo SDR, NE naopak. Již bylo zmíněno, že pro $CR = 2$ (4/6) nebylo možné najít generující matici, tudíž přenos zpráv se nemohl uskutečnit. Neočekávané výsledky se dostavily se změnou SF. S hodnotami 7, 8, 9 a 10 vše pracuje tak, jak má. Při hodnotách 6, 11 a 12 gateway zprávy detekovala, nicméně výstup po dekódování obsahoval naprosto odlišné hodnoty od vysílaných.

Závěr

Praktickým cílem práce bylo, na základě teoretického rozboru, vytvořit funkční LoRa přijímač/vysílač na platformě LimeSDR v softwaru GNU Radio. Tento cíl byl splněn, avšak ne pro všechny hodnoty parametrů modulace.

V implementaci vysílací části nebyl nalezen tvar generující matice pro hodnotu $CR = 2$ (kódovací poměr 4/6), což mělo za následek funkčnost vysílače pouze pro hodnoty $CR = 1, 3, 4$. Dále byl problém s parametrem SF, pro jehož hodnoty $SF = 6, 11$ byly sice vyslané zprávy na testovacím zařízení detekovány, ale jejich obsah byl naprosto jiný nežli obsah zpráv vyslaných. Pro hodnotu $SF = 12$ nebyly zprávy ani detekovány. Pro všechny ostatní hodnoty ($SF = 7, 8, 9, 10$) došly zprávy úspěšně. S různými šířkami pásma se problémy nevyskytly.

Přijímací část byla převzata z [14]. Ani tento blok se nevyhnul problémům s přenosem. Zprávy byly přijímány bezchybně pro všechny kódovací poměry ($CR = 1, 2, 3, 4$), nebyl problém ani se změnami šířky pásma. Přijímač přestal fungovat pro $SF = 6$ s vypnutou volbou "implicit header". Při hodnotě $SF = 12$ nebyly zprávy vůbec detekovány, při hodnotě $SF = 11$ byly zprávy detekovány, ale dekódovaná zpráva byla odlišná od vyslané. Pro ostatní hodnoty ($SF = 7, 8, 9, 10$) pracoval přijímač spolehlivě.

Nefunkční vysílání pro $CR = 2$ by mohlo mít příčinu v nevhodné whitening sekvenci použité při kódování. Všechny whitening sekvence byly převzaty z kódu pro přijímač, které ovšem odpovídají whitening sekvencím pro čip Microchip RN2483. Zařízení, na kterém byl LoRa transceiver testován (LoRaWAN gateway DRAGINO LG01-N), obsahuje čip SX1276.

Takto navržený LoRa transceiver by mohl najít uplatnění v různých analýzách provozu LoRaWAN sítí, avšak po přidání plné funkcionality (provoz na všech SF a CR). Výhodou je potenciální možnost přijímání/vysílání na vícero kmitočtech, veškerá omezení jsou daná propustností dat z SDR do PC a navrženým programem. V takové aplikaci by SDR nebylo prvkem sítě, pouze by se snažilo dekódovat veškerou LoRa komunikaci kterou zachytí. K plné analýze by bylo třeba znát šifrovací klíče AES šifry. Bez znalosti klíčů lze provést analýzu jako v práci [15], která statisticky vyhodnocovala komunikaci v LoRaWAN sítích z pohledu počtu komunikujících zařízení, používaných hodnot parametrů SF, CR, délky dat, a podobně.

Dále by zařízení s SDR mohlo pracovat jako prvek v síti LoRaWAN, ať jako koncové zařízení, či gateway. Využití by našlo kupříkladu jako testovací zařízení pro hledání zranitelností v LoRaWAN penetračními testy. Kdyby SDR účinkovalo jako gateway se síťovým serverem, mohl by jít podvrhnout pravý síťový server a jemu náležící koncová zařázení by byla spravována SDR s PC. Z pozice koncového zařízení by zase bylo možné zamezit funkci gatewaye v síti, pomocí útoku DoS (viz [16]).

Bibliografie

1. SEMTECH. *LoRa Modulation Basics*. 2015. Dostupné také z: https://web.archive.org/web/20190718200516/https://www.semtech.com/uploads/documents/an1200.22.pdf. [cit. 2021-08-09].
2. TORRIERI, Don. *Principles of Spread-Spectrum Communication Systems*. Springer, 2018. ISBN ISBN 978-3-319-70569-9.
3. GHANAATIAN, Reza; AFISIADIS, Orion; COTTING, Matthieu; BURG, Andreas. Lora Digital Receiver Analysis and Implementation. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, s. 1498–1502. Dostupné z DOI: 10.1109/ICASSP.2019.8683504.
4. ROBYNS, Pieter; QUAX, Peter; LAMOTTE Wim a Thenaers, William. A Multi-Channel Software Decoder for the LoRa Modulation Scheme. In: 2018, s. 41–51. Dostupné z DOI: 10.5220/0006668400410051.
5. MARQUET, Alexandre; MONTAVONT, Nicolas; PAPADOPOULOS, Georgios Z. Towards an SDR implementation of LoRa: Reverse-engineering, demodulation strategies and assessment over Rayleigh channel. *Computer Communications*. 2020, roč. 153, s. 595–605. ISSN 0140-3664. Dostupné z DOI: https://doi.org/10.1016/j.comcom.2020.02.034.
6. KNIGHT, Matthew; SEEBER, Balint. Decoding LoRa: Realizing a Modern LPWAN with SDR. *Proceedings of the GNU Radio Conference*. 2016, roč. 1, č. 1. Dostupné také z: https://pubs.gnuradio.org/index.php/grcon/article/view/8. [cit. 2021-08-09].
7. MOON, Todd K. *Error Correction Coding*. John Wiley & Sons, Inc, 2005. ISBN ISBN 0471648000.
8. SEMTECH. *SX1276/77/78/79 Datasheet*. 2015. Dostupné také z: https://www.datasheetq.com/datasheet-download/643974/1/Semtech/SX1276. [cit. 2021-08-09].
9. LORA ALLIANCE, INC. *LoRaWAN 1.1 Specification*. 11. 10. 2017. Dostupné také z: https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/. [cit. 2021-08-09].
10. LORA ALLIANCE, INC. *TS2-1.1.0 LoRaWAN Backend Interfaces Specification*. 19. 10. 2020. Dostupné také z: https://lora-alliance.org/resource_hub/ts002-110-lorawan-backend-interfaces/. [cit. 2021-08-09].
11. ANTENNA GAIN. *Antenna gain – Wikipedia, The Free Encyclopedia*. 2021. Dostupné také z: https://en.wikipedia.org/wiki/Antenna_gain. [cit. 2021-08-09].

12. *LMS7002M Datasheet*. 2017. Dostupné také z: https://limemicro.com/app/uploads/2017/07/LMS7002M-Data-Sheet-v3.1r00.pdf. [cit. 2021-08-09].

13. DRAGINO. *LG01N/OLG01N LoRa Gateway User Manual*. 2020. Dostupné také z: https://www.dragino.com/downloads/index.php?dir=LoRa_Gateway/LG01N/. [cit. 2021-08-09].

14. ROBYNS, Pieter; QUAX, Peter; LAMOTTE Wim a Thenaers, William. *gr-lora: An efficient LoRa decoder for GNU Radio*. Zenodo, 2017. Dostupné z DOI: 10.5281/zenodo.853201. [cit. 2021-08-09].

15. JEŘÁBEK, Ondřej. *Signálová analýza LoRa s využitím SDR*. Brno, 2018. Dipl. pr. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky.

16. YANG, Xueying; KARAMPATZAKIS, Evgenios; DOERR, Christian; KUIPERS, Fernando. Security Vulnerabilities in LoRaWAN. In: *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. 2018, s. 129–140. Dostupné z DOI: 10.1109/IoTDI.2018.00022.

Přílohy

V přiloženém CD k práci je k nalezení zdrojový kód pro OOT modul LoRa kodéru. Dále obsahuje skript pro odesílání přijatých zpráv z gateway na server, zdrojový kód pro zpracování přijatých zpráv na serveru a elektronickou podobu této práce.