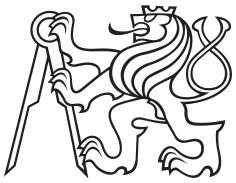
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra telekomunikační techniky

Přehledový přijímač / monitor rádiových sítí IoT

Ondřej Šulc

Školitel: Ing. Pavel Troller, CSc.
Obor: Komunikační systémy a sítě
Leden 2019

Poděkování

Děkujeme ...

Prohlášení

Fakt sám ...

Abstrakt

Rozvíjíme ...

Klíčová slova: IoT, SDR-RTL, LoRa, Sigfox, Přehledový přijímač

Školitel: Ing. Pavel Troller, CSc.
Pestitelský ústav,
Zárivá 232,
12000 Praha 2

Abstract

We develop ...

Keywords: IoT, SDR-RTL, LoRa, Sigfox, Scanner

Title translation: Scanner/Monitor of IoT radio networks

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Nejlepsi prace na svete

Část I

Rešerše

Kapitola 2
Internet věcí

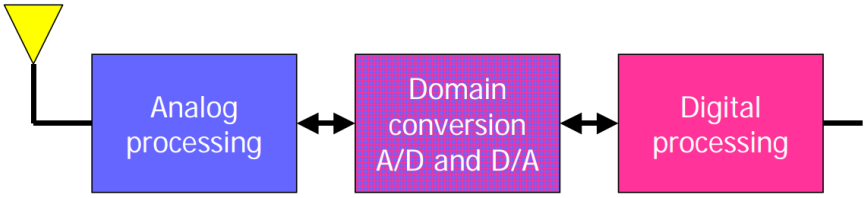
Kapitola 3
Softwérově definované rádio

3.1 Úvod do SDR

Značná část historie použití rádiových vln využívala rádia, která byla implementována čistě v hardware. Pro každou úpravu vlastností rádia, tak bylo zapotřebí ho fyzicky upravit. S rozmachem výpočetní techniky však přišel vývoj i v oblasti rádia a na světlo světa se tak nejdříve dostala SCR (Software controled radio), která umožňovala v omezené míře ovlivňovat některé funkce a pozdější i rádia SDR, která mohou být použita univerzálně díky digitálnímu zpracování signálu (DSP -Digital signal processing).
První známe nasazení SDR v praxi bylo v armádě, jednalo se o systém SPEAKeasy. Systém vyvinula DARPA v roce 1991 a umožňoval díky softwarové implementaci komunikovat prostřednictvím 10 různých vojenských protokolů na frekvencích od 2 MHz po 2 Ghz. SPEAKeasy byl připraven i na přidání dalších modulací a protokolů. O rok později již vyšel první článek o SDR na IEEE, napsal ho Joe Mitola a je tak považován za kmotra SDR. [Nut10]
I přes tyto slibné začátky a podchycení jeho teoretických možností se SDR rozšiřovalo poměrně pomalu a jeho význam narostl až v podlední době s rozmachem levných výkoných integrovaných čipů. Velkou motivací využití SDR do budoucna jsou tzv. kognitivní rádia, která se dokáži přizpůsobit aktualnímu stavu spektra.

3.2 Fungování SDR

Základním znakem každého SDR je, že je značná jeho část realizována jako software běžící na programovatelném a configurovatelném HW zařízení. Toto zařízení můžeme nazvat rádiovou platformou (radio platform) a SW část jako aplikační rámec (aplication framework). Pokud je použitý jednotný standard tak tyto části mohou tvořit univerzální a znovupoužitelné komponenty a tak ušetřit prostředky a usnadnit jakékoliv upgrady.
Jedním z možných standardů je armádou používaná Softwarová komunikační architektura (Software Communication Architecture) vyvinutá v programu JTRS (Joint Tactical Radio System). Tuto architekturu používá většina armádních SDR. [EB06] Jinou možností je de facto standard pro rádio amatéry



Obrázek 3.1: Implementace rádia na základě SDR[EB06]

- GNU Rádio, kterému je v této práci věnována samostatná kapitola **??**.
Pokud na rádio nechceme koukat jako na krabičku ke které se připojí anténa, můžeme identifikovat jednotlivé funkční bloky:

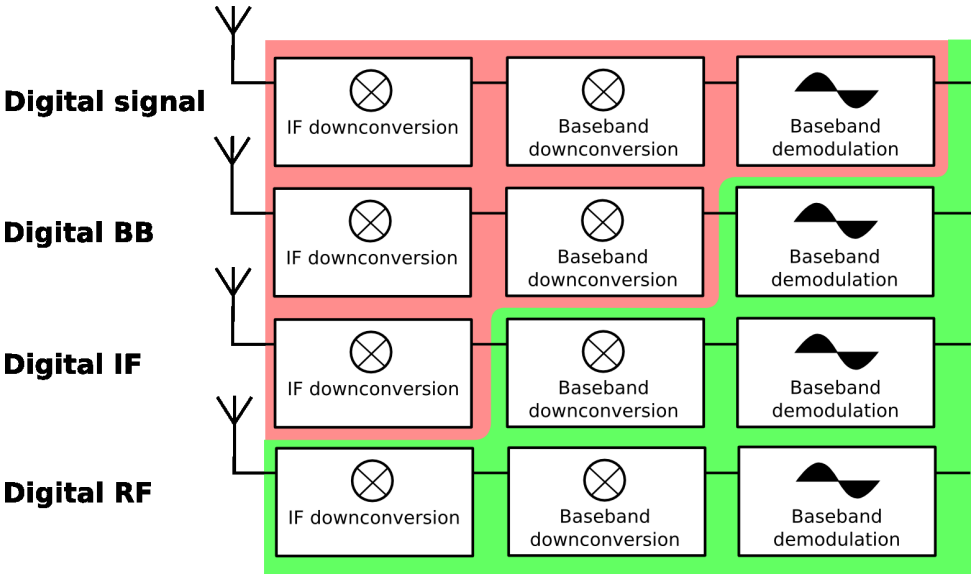
1. Převod z RF (rádiová frekvence) do IF (mezifrekvence)
2. Převod do základního pásma
3. Demodulace
4. Uživatelské rozhraní

Pokud se jedná o SDR můžeme ale zvolit i jednodušší vysokoúrovňový pohled a identifikovat tak tři prvky modelu: Zpracování analogového signálu (front-end), konverze domény (A/D, D/A) a digitální zpracování signálu (back-end), viz obrázek 3.1
Fron-end se stará o převedení rádiového signálu na frekvenci a šířku pása, kterou dokáže zpracovat digítální back-end. Tvoře ho analogové zesilovače, směšovače filtry a oscilátory.
Konvertor domény jak již název napovída převádí anologový signál na digitální a obráceně. Souži mu k tomu vysokorychloostní širokopásmové A/D a D/A převodníky, jejich vlastnosti zásadně ovlivňují možnosti výsledného SDR.
Poslední prvek tedy backend kde probíhá digitální zpracování je založen na FPGA a/nebo DSP programovatelných počítačích, na kterých běží software.
Převod mezi analogovou a digitální doménou se může odehrát v různých místech zpracování viz obrázek 3.2 a podle toho lze systémy kategorizovat.

Digitální signál (Digital signal) V toto případě se v podstatě nejdná o SDR, vše je implementováno v HW. Výstupní signál je však digitální.

Digitální základní pásmo (Digital Baseband) Zde se již část zpracování signálu odehrává v SW. Signál v základním pásmu je navzorkován a modulace se tak odehrává pomocí DSP. Podobné systémy požívají rádio amatéři například pro příjem BPSK31 pomocí rádiového přijímače a zvukové karty počítače na kterém běží SW pro demodlaci.

Digitální mezifrekvence (Digital IF) V této kategorii probíhá vzorkování již na mezifrekvenci a předchází mu konverze z rádiové frekvence

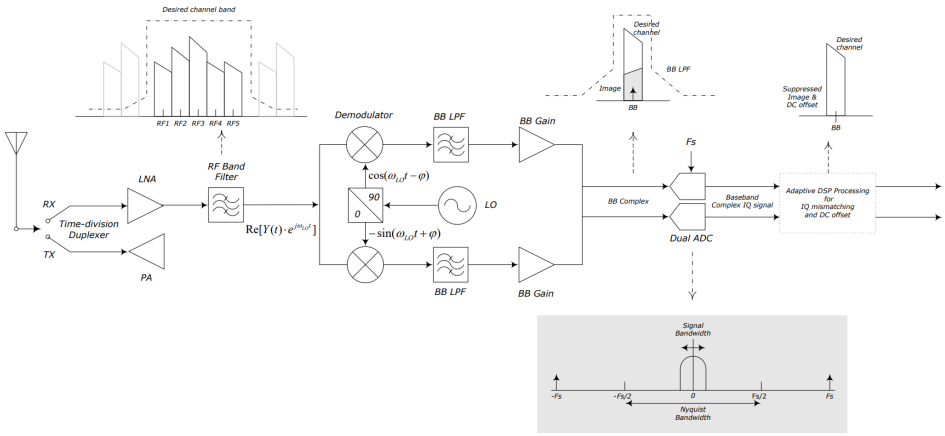


Obrázek 3.2: Klasifikace SDR dle místa konverze domény[**?**]

a filtrování implementované v HW. Podobné systémy jsou využívány radioamatéry nebo například v námořních rádiích, DSP zde slouží k filtrování šumu a demodulaci.

Digitální rádiová frekvence (Digital RF) I při vzorkování přímo RF je potřeba signál nejdříve zesílit a filtrovat až poté je možný jeho převod pomcí ADC. Následně probíhají všechny úkony již v SF. Příkladem systému z této kategorie může být HPSDR Mercury [tap] .

Toto rozdělení však nepočítá s přímou kvadraturní konverzí RF kde je vynechán mezikrok převodu na IF a převádí se rovnou na BB. V takovém případě se RF signál s reálnými hodnotami po zesílení a filtraci smíchá s výstupem oscilátoru s komplexním výstupem (sínus a kosínus), prožene fitrem dolní propust který odstraní vysoké frekvenční komponenty jako boční pásmo vzniklé smišením a poté navzrokuje dvěma ADC.[**?**]
Popsaný postup je výhodný zejména díky své jednoduchosti, není potřeba filtrování IF a cena tak může být nižší. Další výhoda spočívá v tom, že Nyquistova vzorkovací fekvence pro komplexní I/Q signál je dvojnásobná oproti reálnému signálu a tak je možné vzorkovat mnohem širší pásmo.
Mezi nevýhody a také důvody proč se dříve používali slotižejší systémy patří možnost projevení se zrcadlového obrazu signálu a stejnosměrné složky v přijatém signálu. Zrcadlový obraz je způsoben rozdílem ve fázi a amplitudě mezi i a Q kanály. Tyto vady výrazně degradují přijatý signál avšak je možné je efektivně eliminovat korekcí rovnováhy IQ v digitálním zpracování.
Tento systém jsem takto podrobně rozepsal zejména protože ho používá většina dostupných SDR pro rádio amatéry včetně RTL-SDR použitého v této práci.



Obrázek 3.3: Blokové schéma SDR s přímou kvadraturní konverzí RF na BB[wN13]

Možnost přidat výhody a nevýhody http://www.winradio.com/home/facts.htm

3.3 RTL-SDR

Pokud si v minulosti chtěl někdo pořídit SDR musel počítat s částkami mnoha tisíc nebo i desítek tisíc korun a musel mít zároveň mít k dispozici velmi výkonný HW na kterém provádět výpočty. To se změnilo v roce 2012, kdy na scénu přišlo RTL-SDR - původně DVB-T USB tunner, který však s alternativními ovladači může být použit jako SDR a dá se pořídit za cenu do 500 kč.
Celé to začalo v roce 2008, kdy Realtek představil chipset RTL2832U s DVB-T COFDM demodulátorem, hlavním učelem tohoto čipu a na něm postavených USB donglů byl příjem evropského standardu pro pozemní digitální televizní vysílání. Nabízel však i příjem digitálního rádia DAB a analogového FM. Tento malý detail se později stal velmi podstatným.
V roce 2010 totiž Eric Fry při pokusech o napsání ovladačů tohoto DVB-T donglu odposlouchával USB pakety během používání FM aplikace pro Windows. Eric zjistil, že narozdíl od přijmu DVB-T, kdy demodulace probíhá přímo na čipu, při příjmu FM, DAB a DAB+ demodulace probíhá až v SW počítače a přes USB se tedy přenáší I/Q vzorky, jeho primárním cílem však byli ovladače pro Linux a tak nezačal s vývojem pro použití jako SDR.
Informace se sice rozšířila, ale až do roku 2012 nenastal žádný větší pokrok. V tomto roce se o tento DVB-T přijímač od Realteku začal zajímat Antti Palosaari. Ten potvrdil možnost využití jako SDR díky přístupu k 8-bitovým I/Q vzorkům a začal s vývojem potřebného SW. Díky navázání spolupráce s organizací Osmocom, která se v té době vyvýjela vlastní SDR založené na tuneru E4000, což byl jeden z tunerů používaných v DVB-T přijímačích

Realtek, práce nabrala obrátek a brzy byl vyvinut driver SW pro jednoduché použití těchto USB donglů jako SDR. Nejdůležitějším krokem bylo nejspíše vyvinutí ovladače pro linux, o to se v organizaci Osmocom postaral Steve Markgraf. [rok16] [**?**]
Tento počin nastartoval éru RTL-SDR a na jeho základě tak vzniklo obrovské množství projektů všeho druhu, jmenovat budu jen pár těch zajímavějších (pokud čtete tu práci ve formátu PDF můžete kliknutím na projekt přejít na stránky, které se mu věnují):
Spektrální analyzér, Odposlech GSM, Sledování letadel pomocí ADSB, Generátor nahodných čísel, Příjem dat meterologických balónů, Sledování meteoritů, nebo Odposlech vysílaček městské policie

3.4 Další dostupná SDR

RTL-SDR je sice i v současné době to nejlevnější SDR, jeho kvalita však může být pro mnohá použití nedostatečná a tak v posledních letech vzniklo mnoho dalších dostupných SDR. Ta jsou sice o něco dražší, ale díky tomu, že jde o HW pro použití jako SDR navržený, mají větší rozsah, rozlišení, vzorkovací frekvenci a některá mohou kromě přijmu i vysílat.

USRP (Universal Software Radio Peripheral) USRP je celá řada produktů od společnosti Ettus Research. Hlavním vývojářem je Matt Ettus a první USRP spatřilo světlo světa již v roce 2008. V současnosti začínají ceny okolo 20 tisíc korun a v základu má USRP rozsah od 70 MHz po 6 GHz, tento rozsah však lze rozšířit pomocí přídavných desek. Starší hardware je uvolněn jako opensource a veškeré ovladače také. Z těchto důvodů jsou rádia USRP populární ve vědě, na universitách i mezi amatéry. USRP je kromě jiných podporované v GNU-Radio a Matlab SimuLinku a kromě příjmu zvládá i vysílat. [Res]

bladeRF Již v roce 2012 v reakci na RTL-SDR začal vývoj bladeRF. Momentálně se nachazí již ve svojí druhé verzi a zajímavostí je, že jedna z variant obsahuje i FPGA pro HW akceleraci digitálního zpracování. Cena začíná na 11 tisích za verzi bez FPGA, verze s FPGA stojí 16 tisíc korum. [Nua]

HackRF Toto SDR vyvinuté Michaelem Ossmanem v roce 2013 se díky úspěšné kampani na Kickstarter.com začalo prodávat v roce 2014 a jeho cena je cca 7000 kč. Michael Ossman se již dříve proslavil vývojem zařízením pro odposlech bluetooth Ubertooth a HackRF navazuje jako další velmi kvalitní produkt. HackRF dokáže vysílat i přijímat a má rozsah má od 30 MHz po 6 GHz a zvládá i 2x2 MIMO. [Oss13]

AirSpy AirSpy je asi nejlevnější alternativa k RTL-SDR a podobně jako ono neumí vysílat. Velkou popularitu AirSpy získalo zejména díky úzkému propojení s SDR# což je software pro analýzu a demodulaci rádiových

signálů. V nejnovější verzi má rozsah od 24 MHz po 1,8 GHz a podporuje i externí hodiny pro použití vyžadující synchronizaci. [Air]

Název	USRP B205mini	bladeRF 2.0 Micro		HackRF One	AirSpy R2	RTL-SDR
Cena (Kč)	20 000	11 000	16 000	7000	5000	500
Frekvenční rozsah (MHz)	70-6000	47-6000		1-6000	24-1700	24-2200
ADC rozlišení (bits)	12	12		8	12	8
Šířka pásma (MHz)	56	56		20	10	16
Možnost vysílání	Full Duplex	Full Duplex 2x2 MIMO		Half Duplex	Ne	Ne
Přesnost hodin (PPM)	2	0,26		30	0,5	85
FPGA (kLE)	Ne (dražší verze Ano)	49	301	Ne jen CPLD	Ne	Ne

Tabulka 3.1: Přehled dospuných SDR

[Res] [rsa16] [ITE] [Kil13]

Foo	Bar
foo1	bar1
foo2	bar2

Tabulka 3.2: Foobar.

Kapitola 4
LoRa

4.1 Fyzická vrstva (LoRa PHY)

4.1.1 Modulace

Modulační schéma LoRa je založeno na Chirp Spread Spread Spectrum (Cvrlikající rozprostřené spektrum) modulaci (Goursaud and Gorce, 2015) a definuje jeden "cvrk" jako jeden symbol (Semtech, 2015a). Standardní nemodulovaný lineární cvrk se nazývá "základní cvrk" a může být matematicky popsán jako funkce času t takto (Mann and Haykin, 1991):

$$x(t) = e^{i(\varphi_0 + 2\pi(\frac{k}{2}t^2 + f_0 t))} \tag{4.1}$$

Kde φ_0 je počáteční fáze, k je rychlost změny frekvence a f_0 je počáteční frekvence. Pokud je šířka pásma kanálu BW, tak parametry f_0 a k jsou nastaveny tak, že se frekvence zvětšuje od $f_0 - \frac{BW}{2}$ po $f_0 + \frac{BW}{2}$ během periody T cvrku. Tím pádem je $f_0 = \frac{BW}{2}$ and $k = \frac{BW}{T}$. Doba trvání jednoho cvrku závisí na šířce pásma signálu a na parametru nazývaném činitel rozprostření (Spreading Factor - SF) dle vztahu $T = \frac{2^{SF}}{BW}$ (Seller and Sornin, 2014). Vzhledem k tomu, že $x(t + nT) = x(t)$ kde $n \in \mathbb{N}$, celočíselná hodnota $i \in \{0, 1\}^{SF}$ může být namodulována na základní cvrk pomocí časového posunu $\hat{t} = Gray^{-1}(i)\frac{T}{2^{SF}}$ aplikovaného na signál ve vztahu (**??**), kde $Gray^1$ je dekódování Grayova kódu (Gray, 1953). Touto cestou je symbol v podstatě kvantovaný na 2^{SF} časových intervalů rozdělujích šířku pásma, nazýváme je "chipy" a právě ony určují i. Při příjmu modulovaného cvrku s neznámým časovým posuvem $x(t + \hat{t})$, může být hodnota cvrku zrekonstruována navzorkováním signálu vzorkovací frekvencí chipů a výpočtem:

$$i = Gray(arg\max(|FFT(x(t + \hat{t}) \odot \overline{x(t)})|)) \tag{4.2}$$

Kde $\overline{x(t)}$ značí komplexně sdružený základní cvrk, $\odot$ značí multiplikaci po prvcích, $|FFT(x)|$ zančí velikost Rychlé Fourierovi transformace x, a $Gray$ je Grayovo kódování.

4.1.2 Prokládání

Jako v každé jiné modulaci, musíme i zde počítat s chybami způsobenými šumem, interferencí, a časovými nebo frekvenčními posuny. Tyto chyby mohou způsobit, že hodnota čipu nebude dobře odečtena z modulovaného symbolu. Například poryv šumu může posunout vrchol v FFT spektru na jinou hodnotu chipu a tak jej znehodnotit.
Aby bylo možné minimalizovat dopad poryvů šumu na chybu jen jednoho bitu v symbolu je použito prokládání. Několik chipů je dohromady vepsáno do mřížky $\{0,1\}^{SFx(4+CR)}$, kde CR (Coding Rate) značí počet paritních bitů a nabývá hodnot 1 až 4. Pokud tedy bude použit $SF = 7$ a $CR = 4$ dostaneme matici $\{0,1\}^{7x8}$, příklad je na obrázku **??**. K získání kódové slova je pak potřeba číst bity po diagonále matice. Na rozdíl od patentu LoRy (Seller and Sornin, 2014), kde se uvádí, že směr diagonálního čtení bitů z mřížky je směrem dolů, v praxi lze pozorovat opačný směr. Tímto způsobem tak první chip obsahuje všechny nejméně významné bity (LSB - Least significant) všech kódových slov, druhý čip všechny druhé bity všech slov a tak dále. Díky tomu v případě ztráty celého čipu dojde k chybě jen v jednom bitu na kódové slovo.
Dalším způsobem jak zvýšit odolnost proti rušení vysílání je použití módu redukované rychlosti (reduced rate mode). V případě použití tohoto módu jsou první dvě řady prokládací matice zahozeny a její rozměr se tak změní na $\{0,1\}^{SF-2x(4+CR)}$ což způsobí, že z ní nasledně vyčteme o dvě kódová slova méně. Zahozené řádky obsahují nejméně významné bity chipů, které jsou nachylnější k chybám protože odpovídají užším frekvenčním intervalům v FFT spektru. Z toho vyplývá, že mód redukované rychlosti obětuje rychlost přenosu dat ve prospěch odolnosti proti šumu. Hlavička fyzické vrstvy LoRa je v tomto módu vysílána vždy, kdežtkoo užitečná data je v případě použití SF 11 nebo 12.

4.1.3 Kódování

Po přečtení kódových slov z prokládací matice mají tato délku $4 + CR$. Kvůli zamezení vzniku stejnosměrné složky byla slova v části rámce s užitečnými daty XOR-ována 9-bitovým lineární posuvným registrem se zpětnou vazbou (LSFR Linear feedback shift register) (whitening). A proto musí po synchronizaci projít stejným procesem znovu. Přesný algoritmus není v patentu určen a jeho výběr je tedy na každém výrobci zvlášť.
Na několika testovacích zařízeních **??** reverzním inženýrstvým zjistilo použité upraveného $4/(4 + CR)$ Hammingova kódu. Ve výsledku tak z každého kódového slova po dekódování získáme 4 bity dat. Ta jsou pak naparsována du struktury rámce lora.

4.1.4 Struktura rámce

Na fyzické vrstě LoRa definuje rámec jako strukturu složenou z následujících polí. Pole jsou uvedena ve stejném pořadí jako v rámci. (Semtech, 2015b, p. 27–29)

Preambule Sekvence základních cvrků, která slouží k časové a frekvenční synchronizaci. Počet cvrků není pevně dán.

Symboly synchronizace rámce Dva modulované cvrky co mouhou být použity pro identifikaci sítě. Hardwarový přijímač zahodí rámcec, které obsahují synchronizační symboly co neodpovídají jeho nastavení.

Symboly synchronizace frekvence Dva sdružené cvrky následované sdruženým cvrkem s periodoou $\frac{T}{4}$ určené pro přesnou frekvenční synchronizaci.

Hlavička (nepoviná) Hlavička obsahuje délku užitečných dat, použitou přenosovou rychlost, indikuje použití Cyklického redundantního součtu (CRC - Cyclic redunduncy check) a jendobajtovou kontrolní sumu hlavičky. Pro modulaci hlavičky je vždy použito $CR = 4$ a mód redukované rychlosti. Pokud hlavička vysílána není (implicitní mód) musí mít jak přijímač tak vysílač předem schodně nastavený CR a také zdali je použito CRC.

Užitečná data Pole o proměnné délce obsahující data vrstvy přístupu k médiu (MAC - Media access control) a případné dvoubajtové CRC těchto dat.

4.1.5 Struktura hlavičky

Délka hlavičky není ve specifikaci nikdy přímo určena. Lze jí však vydedukovat z toho, že hlavička je vždy vysílána v módu redukované rychlosti, má $CR = 4$ a SF minimálně 7. Z toho vyplívá že hlavička se musí vejít do mřížky $\{0, 1\}^{7-2x8)}$ a to odpovídá 5 kódovým slovům. Každé slovo má 8 bitů a dohromady je to bitů 40. Jakékoliv zbývající bity jsou použity pro užitečná data.
Po dekódování díky redundantním bitům dostáváme $40\frac{4}{8} = 20$ bitů nebo 2,5 bajtu. V [RQLT18] experimentálně vyzkoušeli pořadí hlavičky. První bajt udává délku datového obsahu, následuje půlslabika udávající CR a přítomnost MAC CRC a poslední bajt obsahuje kontrolní součet hlavičky, z něj je však používá jen 5 LSB bitů.

4.2 Softwarová demodulace

?? dokázali implementovat kompletní PHY vrstvu LoRa ve frameworku GNU Radio. Jejich zdrojové kódy jsou open source a dostupné na Githubu. Funkčnost příjmu signálu LoRa mého scanneru vychází z jejich práce. V této kapitole je pospsán princip fungování.

4.2.1 Detekce a synchronizace

Aby mohl být signál demodulován musí být nejdříve detekován. K tomu slouží preamule která má dva opakující se cvrky čehož dokáže využít použitý

Schmidl-Cox algoritmus. Ten definuje dvě veličiny $P(d)$ a $R(d)$, ty jsou definované takto (Schmidl a Cox, 1997) [SC97]:

$$P(d) = \sum_{m=0}^{L-1} (x^*_{t+m} x_{t+m+L}) \tag{4.3}$$

$$R(d) = \sum_{m=0}^{L-1} |x_{t+m+L}|^2 \tag{4.4}$$

kde L je délka symbolu, t je index vzorku komplexnímo signálu x a x^* je jeho komplexně sdružený signál. Veličiny $P(d)$ a $R(d)$ jsou použity k výpočtu časové metriky $M(d)$:

$$M(d) = \frac{|P(d)|^2}{R(d)^2} \tag{4.5}$$

Časová metrika $M(d)$ v podstatě počítá normalizovanou autokorelaci délky L přes dva symboly, maximum bude mít ve cvhíli kdy v signálu budou za sebou dva totožné symboly. Díky tomu, že oba symboly jsou chybami způsobenými přenosem (interference, frekvenční odchylka nosné (CFO - Carrier frequency ofset), odchylka vzorkovací frekvence) ovlivněny stejně, tak tyto chyby téměř neovlivní výsledek korelace. Aby bylo možné efektivně počítat rovnice **??** a **??** v programu byla použita knihovna VOLK (Vector Optimized Library of Kernels), která implementuje SIMD (Single Instruction, Multiple Data) instrukce. Na obrázku **??** je vidět příklad výsledku použití časové metriky $M(d)$ na komplexním signálu LoRa. Kolem vzorku 2500 funkce dosahuje horní plošilny a poukazuje tak na existenci preambule.
I přesto že tento algorimus detekuje preambuli velmi dobře, není možné přesně určit počátek symbolu jen z horní plošiny časové metriky. Tým kolem Wanga **??** proto navrhl vylepšení kdy je od časové metriky $M(d)$ odečtena její opožděná verze $M_2(d)$, tím se z plošiny stává vrchol **??** a lze tak za začátek symbolu považovat vzorky odpovídající maximu této metriky. Nicméně ani to není jak je patrné z **??** dostatečně přesné pro signály LoRa.
Aby **??** tenhle problém vyřešili museli vymyslet nové řešení. To se zakládá použití Schmidl-Coxovi metriky pro přibližné určení okna ve kterém se nachází druhý symbol preambule a následném zpřesnění pomocí ideálního lokálně vygenerovaného cvrku. Jeho okamžitá frekvence $\omega_l(t)$ a normovaná okamžitá frekvence signálu Lora $\omega(t)$ jsou vzájemně korelovány (přes posuvné okno?) a index vzorku jež odpovídá maximální hodnotě této funkce je považován za počátek symbolu. Použití omažité frekvence místo komplexních hodnot je odůvodněno chybami CFO, které by bez korekce mohly ovlivnit přesnost synchronizace. Použitím okamžité frekvence jsou podobně jako u Schmidl-Coxova algoritmu tyto chyby zanedbatelné.

$$symbolstart = \underset{i \in \{0,1,\dots L\}}{\operatorname{argmax}} (\omega_l \star \omega)(i) \tag{4.6}$$

Výsledek je na **??**. Poslední součástí tohoto řešení je určení prahové hodnoty maxima korelačního koeficientu okamžitých frekvencí lokálnlně gerovaného cvrku a přijátého. Pokud je tato hodnota menší než prahová je daný rámec zahozen, protože se buďto jedná o falešně pozitivní detekci rámce nebo o nepovedenou synchronizaci.

4.2.2 Demodulace

Po úspěšné synchronizaci následuje fáze demodulace popsaná v předchozí kapitole. Oproti teorii má však v praxi FFT demodulace nevýhodu v tom, že je citlivá na odchylku frekvence, která způsobuje posun hodnot FFT a tím i odečítaných hodnot chipů. Tím pádem je potřeba přesná synchronizace frekvence, kterou je navíc potřeba aplikovat na každý kanál LoRa zvlášť. Separace kanálů a následná synchronizace každého z nich je však v softawaru příliš náročná operace a tak **??** přišli s novou metodou demodulace, která je nezávislá na frekvenci a umožňuje demodulaci na všech kanálech současně v reálném čase. V porovnání s FFT metodou je však méně robustní.
Nejdříve je potřeba spočítat okamžitou úhlovou fekvenci $\omega[t] = \frac{d\varphi[t]}{dt}$. Poté je potřeba $\omega[t]$ vyhladit a decimovat konstantním decimačním faktorem $\frac{s_f T}{2^{SF}}$ kde s_f je vzorkovací frekvence. Díky tomu je pak počet vzorků v $\omega[t]$ schodný s 2^{SF} následně je vypočítán digitální gradient f:

$$D_t[\omega[t]] = \omega[t+1] - \omega[t] \tag{4.7}$$

Tuto operaci si lze představit jako filtr horní propusť okamžité frekvence nebo jako druhou derivaci fáze. Protože frekvence základního cvrku se lineárně zvyšuje s k - $\omega(t) = kt + f_0$ je její derivace $\omega'(t)$ rovna k. Pro modulované cvrky se však v D_t objeví ostré šličky v místech přechodu mezi vysokou a nízkou frekvencí. Přítomnost takových špiček indikuje časový posun $\hat{t}$, nepřítomnost naopak idikuje časový posun 0 - základní cvrk.

Dalším problémem při demodulaci je zpožďování/předbíhání hodin v jednotlivých zařízeních. Kristalové oscilátory v LoRa vysílači a SDR se budou zákonitě navzájem předbíhat nebo zpožďovat, rozdíl jejich frekvencí je předem neznámý, ale v průbhu času se musí projevit. To může způsobovat problémy zejména v případě delšího datového obsahu v kombinaci s vyšším SF. V patentu LoRa jsou pro účely korekce tohoto jevu použity pilotní symboly, které pomohou sledovat časování. Ve skutečnosti se však zdá, že k jejich použití nebylo přistoupeno. Je tedy nutné využít techniku slepého odhadu, která využívá převzorkování přijatého signálu N-krát. Aby tato technika byla funkční je potřeba aby hodnota N odpovídala následujícímu vztahu $|\Delta t| < \frac{N}{2}$, kde Δt je chyba časování na symbol. Prvním krokem je synchronizace popsaná v **??**. Pokud je chyba časování na symbol $|\Delta t| < \frac{N}{2}$
, lze $|\Delta t|$ určit následujícím způsobem:

1. Symbol je demodulován běžným způsobem jak je popsáno v 4.2.2 a je tak získána hodnoto chipu i a časového posunu $\hat{t}$.

2. Na přijímači je lokálně generovaný základni cvrk modulován na i což způsobí časový posun $\hat{t}_1$ lokálního signálu.

3. Protože lokálně generovaný cvrk není ovlivněn rozdílem oscilátorů vysílače a přijímače můžeme vzájemné zpoždění oscilátorů definvat takto $\Delta t = \hat{t}_1 - \hat{t}$. Nyní stačí na příjimači opravit $\hat{t}$ připočtením vzájemného zpoždění k přijatému signálu.

Již zmíněná podmínka $|\Delta t| < \frac{N}{2}$ je daná tím, že při jejím nesplnění dekodér špatně určí hodnotu chipu i a chyba se tak bude šířit i do dalších symbolů. Hodnota N tak není určena přímo ale jako interpolace z $\hat{t}$ do $\hat{t} + \Delta t$. Vyšší hodnoty N by dále vylepšovali přesnost korekcí zpoždění ale také by zvyšovali náročnost výpočtu.

4.2.3 Dekódování

Ve fázi dekódování jsou hodnoty chipů zpětně proloženy a tím jsou získána kódová slova s 4 + CR bity. Prvních 8 kódových slov lze přímo dekódovat jako hlavičku fyzické vrstvy. Datový obsah však musí být nejprve XORován bělící posloupností. I přesto že dle výrobce LoRa Semtech má jít o sekvenci generovanou 9-bitovým LFSR ve skutečnosti je dle výzkumu [KNI16] a [Blu16] použita posloupnost mírně odlišná, která však není veřejně zdokumentovaná. Pokud však známe původní kódové slovo i přijaté vybělené lze snadno zjistit sekvenci použitou pro XORování následovně $w^{(j)} = c_w^{(j)} \oplus c^{(j)}$. Pro zjednodušení lze vyslat všechna kódová slova nulová a tím dostaneme rovnici $w^{(j)} = c_w^{(j)} \oplus 0$ a na výstupu tak máme přío samotnou bělící sekvenci, kterou můžeme uložit a poté náčítat z tabulky.
Po Odbělování přichází poslední krok a to Hammingovo dekódování kódových slov. U LoRa jsou datové bity umístěny na jiných pozicích než je běžné a to na indexech 0,1,2 a 3 bajtu místo indexů 1,2,3 a 5. Graficky je toto mapování zobrazeno na **??**. Po extrakci datových bitů jsou pak pomocí paritních bitů detekovány a případně opraveny chyby a tím získána původní data.

Kapitola 5

SigFox

5.1 Rádiový protokol Sigfox)

Základním poznávacím znamením technologie Sigfox je pužití velmi úzkého pásma (UNB - Ultra Narrow Band). To dovoluje vysílání na větší vzdálenosti díky úzkému zaměření vysílacího výkonu každého zařízení. Navíc díky použití UNB může na jednom místě operovat velké množství zařízení bez přílišného vzájemného rušení.
Sigfox používá různá modulační schémata pro uplink a downlink, důvodem je optimalizace využití spektra. Různá je i šířka pásma, přenosová rychlost, vysílací výkon i struktura rámce. Popíšu tedy oba směry zvlášť a zaměřím se přitom na parametry používané v Evropské unii. V jiných regionech se parametry lyší, důvodem jsou různé alokace spektra.

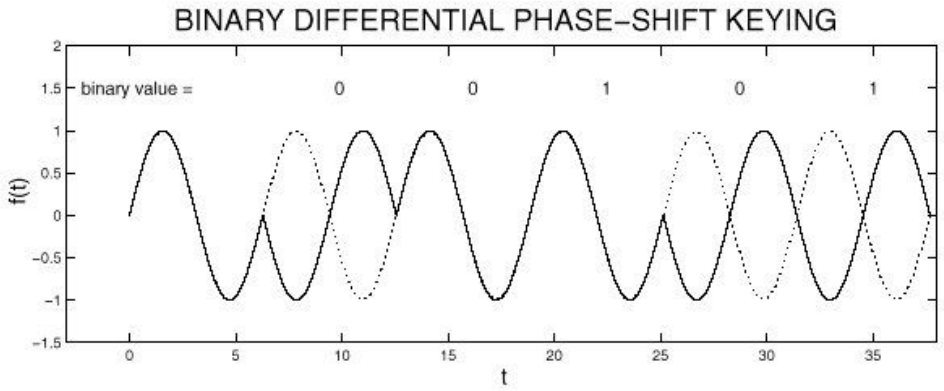
5.1.1 Fyzická vrstva uplinku

Sigfox pro Uplink používá modulaci typu DBPSK (Differential Binary Phase Shift-Keying). Tato modulace stejně jako standartní BPSK používá fázi pro určení symbolu, narozdíl od ní však nejen zaznamená přechod z 0 do 1, ale i bez referenční fáze rozlyší zda je daný symbol 0 nebo 1. Toto je možné díky tomu, že symbol neurčuje konkrétní fáze, ale její změna, odtud také plyne její pojmenování.
Na obrázku 5.1 je znázorněn příklad fungování této modulace. Pokud nenastává žádná změna ve fázi signálu identifikujme 1 pokud změna nastane pak je symbol určen jako 0. V tomto příkladu každý symbol trvá jednu periodu signálu a to by při frekvenci 868 MHz kterou Sigfox v evropě používá znamenalo přenosovou rychlost 868Mb/s což nejen nedává smysl pro daný účel, ale navíc tak Sigfox ani ve skutečnosti fungovat nemůže kvůli nemožnosti dekódobat podobný signál na jakoukoliv použitelnou vzdálenost.
Verze DBPSK Sigofxem používaná pro každý symbol sadu period signálu a hodnotu symbolu určí podle toho zda se průběhu vysílání této sady změní fáze (0) či nikoliv (1). Demodulace tak není tolik ovlivněna lokální proměnností fáze.
V případě evropské verze je přenosová rychlst uplinku 100 bit/s a frekvence signálu 868,1 Mhz. Z toho vyplívá že každý bit/symbol odpovídá 8 681 000 pe-

	Uplink	Downlink
Šířka pásma	100 Hz	1500 Hz
Přenosová rychlost	100 baud	600 baud
Modulace	DBPSK	GFSK
Vysílací výkon	25 mW	500 mW
Frekvence	868,0 - 868,6 Mhz	869,40 - 869,5 Mhz
Střída	1 %	10 %

[ZP16]

Tabulka 5.1: Parametry přenosu Sigfox pro Uplink a Downlink



Obrázek 5.1: Příklad modulace DBPSK [MWF]

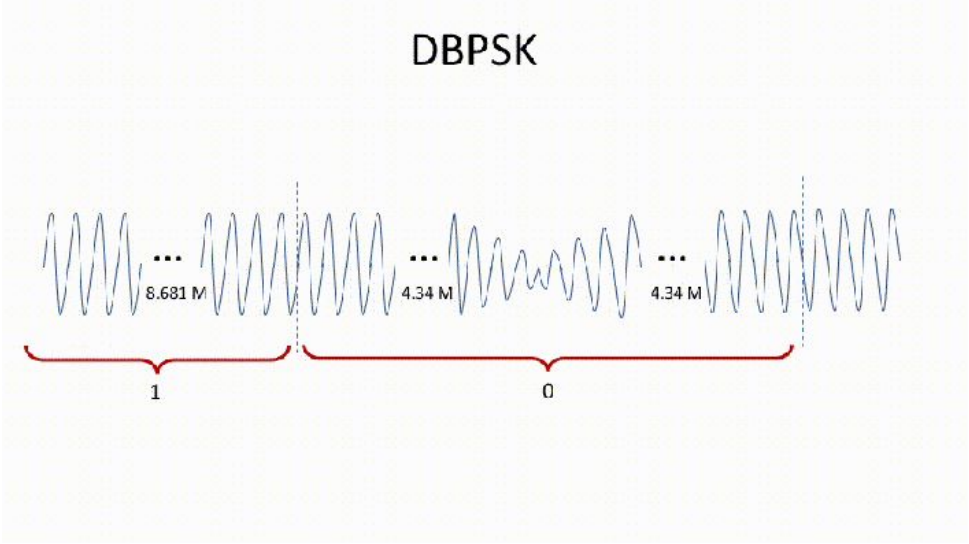
riodám. Velké množství period využívá příjimač pro statistické určení posunu fáze každého bitu. V případě 1 bude celá sada mít fázi stejnou a v případě 0 se fáze v prostřed posune.

Pokud by se však fáze posunula při plné amplitudě znamenalo by to velký negativní dopad na spektrální efektivitu. Aby se tento efekt zmenšil je také okolo posunu fáze modulována amplituda. Grafické znázornění této modulace je na obrázku 5.2.

5.1.2 Struktura rámce uplinku

Na fyzické vrstě Sigfox definuje rámec jako strukturu složenou z následujících polí. Pole jsou uvedena ve stejném pořadí jako v rámci. V prvním opakování není použito žádné specifické kódování a data tak jsou přímo čitelná. V druhém a třetím opakování jsou však pole F.TYPE, SEQ.ID, DEVICE.ID a PAYLOAD zakódovány.

Při druhém opakování hodnota každého bitu závisí na dvou předchozích, pokud jsou stejné tak je hodnota aktualního bitu prohozena. Při třetím opakování závisí hodnota aktuálních dvou bitů vždy na hodnotě dvou předchozích !!!!!!!! uplne nevim jak !!!! možná jen tak, že pokud je objeden dozadu nula tak prohazuji hodnotu.



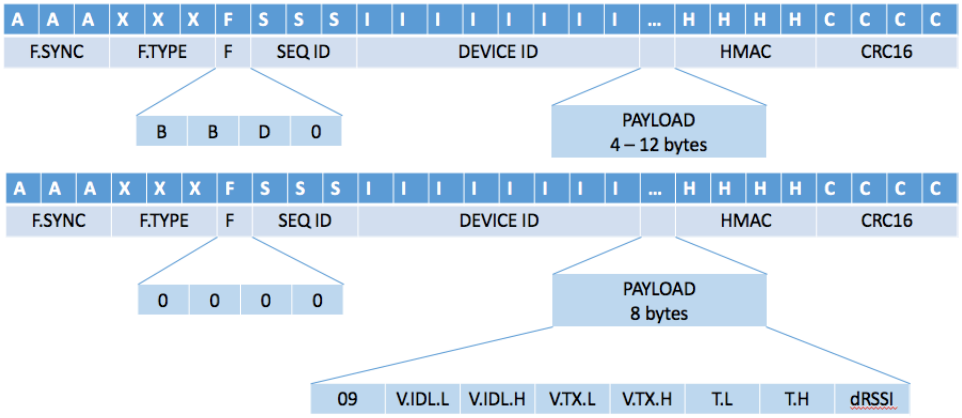
Obrázek 5.2: Modulace amplitudy při změně fáze [Dis17]

Následkem použití jiného kódování pro každé opakování, není i přes identický obsah nikdy vysílán stejný rámec třikrát. Změny fáze jsou vždy na jiných místech, to může pomoci při přijmu zejména v podmínkách kdy se signál ztrácí v šumu.
Speciálním typem rámce uplinku je rámec OOB (Out of Band) který slouží jako potvrzení příjmu rámce downlinku. Lyší se v několika ohledech, jednak není opakován a druhak nepřenáší uživatelsky užitečná data. Místo nich v PAYLOAD poli, které je vždy dlouhé 8 bajtů, zasílá informace o napětí během nečinosti i během vysílání, teplotu, RX RSSI.

F.SYNC Nebo také Frame Synchronization jak již název napovídá slouží pro synchronizaci. Objevuje se na začátku každého rámce. V rámci této části se střídá 0 a 1 po dobu 3 půlslabik (12 bitů), objevuje se tedy 6 změn fáze. Tyto změny fáze využívá příjímač k synchronizaci hodin aby později korektně identifikoval začátek každého bitu.
Synchronizace je ukončena ve chvíli, kdy se vzor změní z 01 na 00 nebo 11 v závislosti na typu rámce. Typ rámce je určen v následujícím poli F.TYPE.

F.TYPE Frame Type je část rámce, dle které lze rozeznat o kolikáté opakování (každý rámec se opakuje třikrát) vysílání se jedná a také délku užitečných dat. Přesné hodnoty nejsou nikde definovány, ale z pokusů o reverzní inženýrství [Dis17] vyplývá tabulka 5.2. (mohu/nemohu potvrdit) Zajímavé je, že opakování a délka dat nemají samostatné indikační bity, ale existují konstanty pro jejich kombinace.

F.FLAGS Tato půslabika je rozdělená do tří částí. První dva bity určují počet bajtů přidaných do rámce jako výplň pro dosažení jedné z předdefinovaných délek. To znamená, že pokud by délka zprávy vyšla na 6



Obrázek 5.3: Struktura rámce Sigfox pro Uplink (nahoře) a konkrétní podoba rámce OOB (dole). Každá buňka reprezentuje půslabiku (4 bity) [Dis17]

Délka zprávy	1. opakování	2. opakování	3. opakování
12 bajtů	0x94C	0x971	0x997
8 bajtů	0x611	0x6BF	0x72C
4 bajty	0x35F	0x598	0x5A3
1 bajt	0x08D	0x0D2	0x302
1 bit	0x06B	0x6E0	0x034
RX OOB	0xF67	-	-

Tabulka 5.2: Hodnoty F.TYPE v závislosti na délce zprávy a pořadí opakování [Dis17]

bajtů, tak bude odeslána v osmi bajtovém rámci a dva bajty tak budou přidány a hodnota těchto bitů bude 10 neboli 2.
Třetí bit značí zda je na tento rámec vyžadována odpověď ve formě downlinku. Pokud je tento bit 1 síť by měla v následujícím intervalu pro downlink odpovědět. Poslední bit je vždy 0.
V případě OBB rámce jsou všechny bity 0.

SEQ.ID Sequence ID je 12 bitové pole, které se inkrementuje s každým odeslaným rámcem. Slouží jako určitá ochrana proti znovuodeslání zachycených zpráv. Limitem této ochrany je, že se SEQ.ID každých 2048 zpráv opakuje a tak v určitých situacích nemusí backend znovuodeslanou zprávu zachytit. Bity jsou seřazeny od nejvýznamějšího po nejméně významný.

DEVICE.ID Toto pole je dlouhé 32 bitů a představuje unikátní ID Sigfox zařízení. Bajty jsou řazeny od nejméně významného, ale bity každého z nich jsou řazeny od nejvýznamějšího bitu.

PAYLOAD Toto pole obsahuje přenášená data a může být dlouhé 12, 8 nebo 4 bajty.

Preamble (91)	Frame Sync (13)	ECC (32)	Payload (0 - 64)	MAC (16)	FCS (8)

Tabulka 5.3: Pole rámce Downlinku (délka pole v bitech)

HMAC HMAC (hash-based message authentication code) je druh MAC projejíž výpočet je použitá kryptografická hešovací funkce a tajný klíč. Slouží jak ke kontrole integrity dat, tak k autentifikaci. Každé Sigfox zařízení má svůj tajný privátní klíč vypálený přímo na čipu. Délka pole je 2 bajty.

CRC Poslední pole je CRC s délkou 16 bitů.

5.1.3 Downlink

Pro komunikaci směrem ze sítě do zařízení Sigfox používá modulační schéma GFSK, vyšší výkon, širší pásmo a navíc má i větší poměr vysílacího času, viz 5.1. Mohlo by se zdát, že jsou Uplink a Downlink kanály nevyvážené ve prospěch Downlinku, ale opak je pravdou jelikož Downlink je vysílán pro všechna zařízení v dosahu a Uplink se počítá pro každé zařízení zvlášť. V Evropě po započtení všech omezení vychází pro každé zařízení 140 Uplink zpráv denně a jen 4 Downlink zprávy.
Přesný obsah rámce ani další detaily momentálně neznám. Pouze lze něco málo vyčíst z tabulky 5.3.

Část II

Přehledový přijímač

Kapitola 6

Hardware

V této kapitole shrnu veškerý potřebný hardware pro přehledový přijímač. Jedninou povinnou součástí určenou v zadání bylo SDR, ostatní komponenty však vyplynuly z potřebných vlastností přijímače, a tak nezbylo příliš prostoru pro invenci. Cíl bylo zhotovit přijímač tak aby byl mobilní, mohl funguvat na baterie a vyhověl zadání v tom smyslu, že kromě příjmu dat ze sítí IoT bude schopen je také předat po IP protokolech dál a sám přehledně zobrazit.

6.1 RTL-SDR

I přesto, že konkrétní druh SDR nebyl v zadání určen, byla jeho volba jen formální záležitostí. Jak už totiž vyplývá z kapitoly refchapter:sdr RTL-SDR je de facto stardard pro amatérské využití. Mezi ostatnímy SDR se vyjímá především cenou, která je desetinnásobně až stonásobně menší. Jeho hlavní nevýhodou oproti dražším variantám je nemožnost vysílání, tato nevýhoda však vzhledem k určení není relevantní. Dalším potencionální nevýhodou by mohl být rozsah. Vzhledem k tomu, že jsem se však rozhodl soustředit pouze dva druhy Iot sítí - LoRa a Sigfox a obě pracují v bezlicenčním pásmu ISM 868Mhz, je i tato nevýhoda bezvýznamná. Ostatní nedostatky oproti dražší konkurenci jsou již tak malé, že by pro účely této práce nedávalo smysl volit jinak.
V případě dalšího vývoje v budoucnosti je však možné, že bude potřeba SDR upgradovat. To může být motivováno technologií NB-Iot, kterou u nás provozuje operátor Vodafone. Tato technologie funguje na stejných pásmech jako LTE a tak se vzhledem k nemožnosti příjmu fekvencí nad 1766 Mhz může NB-IoT dostat mimo rozsah RTL-SDR. Upgrade by pak mohl proběhnout dvěma způsoby, buďto přidáním konvertoru, nebo výměnou samotného SDR. To už jsou však úvahy náležící pozdějším pracem. V obou případech se však díky využití GNU Radio bude jednat o jednoduchou výměnu, bez nutnosti větších zásahů do software přehledového přijímače.
Specifikovat vybrané SDR pouze jako RTL-SDR však nestačí. Toto označení se používá pro nepřeberné množství USB DVB-T tunerů používajících čipset RTL2832U. Ty se mezi sebou mohou lišit ve několika ohledech. Prvním je použitý tuner. Nejžádanější variantou pro účely SDR je tuner E4000 od firmy Elonics, má totiž nejširší rozsah z používaných tunerů.

citeosmocom Jeho nevýhodou však je dostupnost, tento tuner se již nevyrábí a tak jsou RTL-SDR s tím to tunerem ke koupi již jen ze zapomenutých zásob obchodů a nebo z druhé ruky. Jejich cena tak oproti ostatním variantám v poslední době stoupla i přesto, že ještě před několika lety se nijak nelišila. V poslední době je tak nejběžnějším tunerem R820T2, ten má sice rozsah o něco menší než E4000 (ale větší než ostatní tunery), nemá v něm však mezery a je tak i vzhledem k dostupnosti nejvhodnější variantou.
Druhým zásadním parametrem, kterým se můžou RTL-SDR lišit je konektor antény. Vzhledem k tomu, že byly původně určeny pro sledování televize, mají často konektor PAL nebo MCX. [**?**] To v zásadě nepředstavuje problém, ale pro práci s rádiem se častěji používá konektor SMA a proto existuje také větší množství a výběr antén k tomuto konektoru.
Posledním zásadním parametrem odlišujícím různá RTL-SDR je přitomnost teplotně kompenzovaného oscilátoru (TXCO). Ten je užitečný, protože při použití se RTL-SDR zahřívá a pokud není jeho oscilátor tepelně kompenzovám, tak se jeho frekvence mění. Zahřívání na plnou provozní teplotu trvá několik minut, během této doby zůstane nalazená frekvence u SDR-RTL s TXCO stabilní, kdežto u ustatních se bude posouvat.

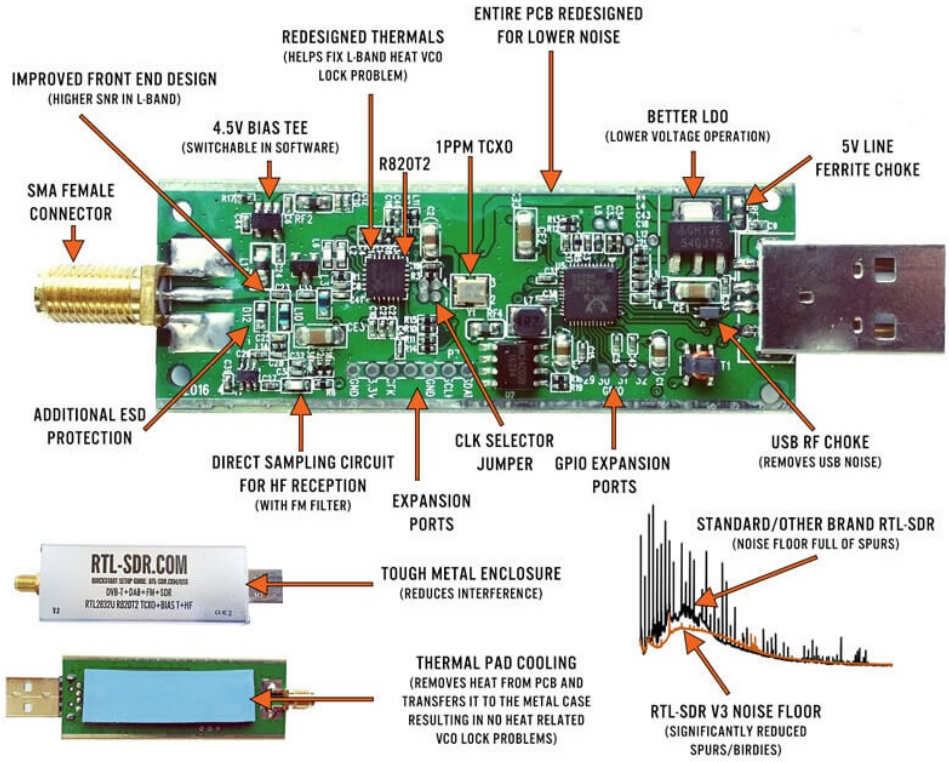
Po zvážení těchto skutečností a také kvůli dalším úpravám mířeným na zlepšení použití jako SDR jsem zvolil RTL-SDR Blog V3. Tato varianta RTL-SDR byla speciálně upravena pro použití ne jako DVB-T tuner ale jako SDR. Kromě tuneru R820T2, konektoru antény SMA a TXCO má i další vylepšení. Nejdůležitějším vylepšením pro použití v přehledovém přijímači je použití hliníkového krytu, který funguje jako chladič pro interní komponenty a zároveň jako elektromagnetický štít. To spolu s lepším návrhem PCB také snižuje množství šumu. Další vylepšení nejsou pro účely této práce až tolik podstatná spíše přispívají k znovupoužitelnosti tohoto SDR v dalších projektech. Konkrétně se jedná o softwarově spínatelné bias tee (napájení např. nízkošumového zesilovače z RTL-SDR), režim přímého vzorkování (zpřístupňuje fekvence 500 kHz - 24 MHz přímo přes SMA konektor) a mnoho dalších maličkostí, jejich kompletní výčet je v obrázku 6.1

6.2 Raspberry Pi

6.2.1 Displej

6.3 Napájení

Vzhledem k potřebě fungování přehledového přijímače na baterky bylo nutné řešit napájení složitěji, než jen zapojením Raspberry Pi do USB nabíječky. Bylo potřeba zvolit správnou baterii, DC-DC měnič s výstupem 5V pro napájení Raspberry PI (A přes něj ostatních komponent), způsob nabíjení baterií, vypínač a propojení těchto komponent.
Jako zdroj energie jsem zvolil články typu 18650 Li-Ion. Je to stejný typ



Obrázek 6.1: Výhody RTL-SDR blog V3 oproti běžným RTL DVB-T [**?**]

článků jako se používá v přenosných počítačích, elektromobilech, vaporizérech a power bankách. Využívá technologii Li-Ion a díky tomu netrpí paměťovým efektem jako starší Ni-Cd a má vysokou hustotu energie. K dispozici jsem měl články značky Samsung model ICR18650-26. Kapacita jednoho článku je 2,55 Ah a nominální napětí 3,7 V. Články jsem použil tři, vložil do zakoupeného držáku a zapojil paralelně. Dostal jsem tak baterii 1s3p s kapacitou 7,65 Ah a nomilnálním napětím 3,7 V (min 3V, max 4,2V) z čehož vyplývá kapacita ve watthodinách okolo 28 Wh.

Vzhledem k tomu, že Raspberry Pi vyžaduje napětí 5 V bylo potřeba napětí z baterie zvýšit na tuto hodnotu. Zároveň jsem potřeboval vyřešit jakým způsobem budu baterii nabíjet. Stejná napětí a problémy však řeší každá power banka a tak jsem se uchýlil k již připravenému řešení v podobě booster modulu určeného do powerbanky. Tyto moduly dokáží zvýšit napětí z baterie na 5V, ohlídat napětí na bateriích aby nekleslo pod 3V (při menších napětích dochází k poškození článků) a zároveň přri zapojení zdroje dokáží baterii nabíjet a ohlídat, že její napětí nepřesáhne hodnotu 4,2 V, nad kterou dochází k poškození článků a i případnému vzplanutí.

Vyzkoušel jem několik druhů (z Aliexpress se dají podobné moduly koupit za malé častky) a nakonec se rozhodl pro HCX-PCB-429. Tento modul nejen deklaruje maximální výstupní proud 2A, ale doopravdy ho i dosahuje, což

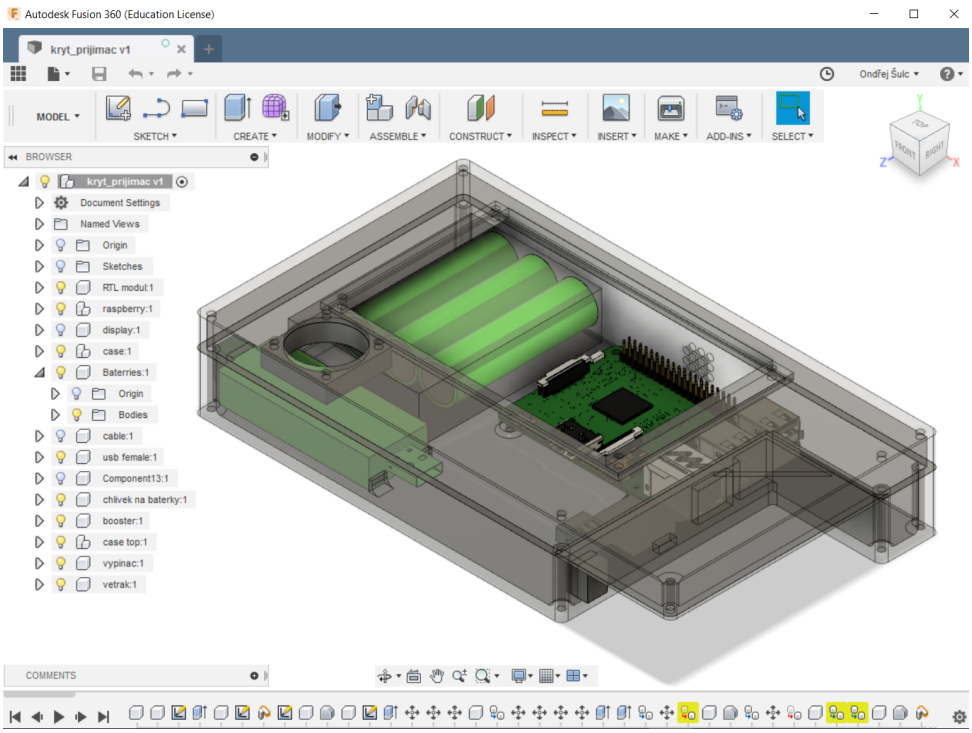
se většině ostatních nepodařilo. Další jeho výhodou je, že při připojování a odpojování nabíječky nepřeruší dodávku elektřiny na 5 V výstupu, to u těchto modelů bohužel nebývá běžné. Jeho posledí výhodou oproti dalším vyhovujícím je jeho velikost, ostatní kvalitnější moduly byly navrhnuty pro powerbanky s dvěma výstupy a měli tím pádem větší rozměry.
Raspberry Pi samotné nemá žádný vypínač, musel jsem tedy přidat vlastní. Použil jsem jednoduchý kolébkový vypínač a umístil ho v obvodu až za booster modul, díky tomu lze baterii napájet i ve vypnutém stavu a spotřeba samotného modulu je minimální, je na neustálé připojení k baterii stavěn.
Odstavec o výdrži.

6.4 Kryt

Abych z již představených jednotlivých komponent dokázal vyrobit přehledový přijímač, bylo potřeba je dohromady umístit do ochraného krytu. Přestože bývá pro účely závěrečných prací běžné sestrojit kryt z plexiskla s vyřezanými otvory, rozhodl jsem se jít jinou cestou - navrhnout a posléze na 3D tiskárně vytisknout kryt přizpůsobený na míru komponentům.
Vzhledem k tomu, že jsem neměl mnoho předchozích zkušeností s návrhem 3D modelů, dal jsem na doporučení kolegy Jaroslava Bartoše, který se 3D tiskem zabývá a pro návrh zvolil CAD software Autodesk Fusion 360. V rámci studia mají studenti všech vzdělávacích institucí k dispozici licenci zdarma

6.4.1 Autodesk Fusion 360

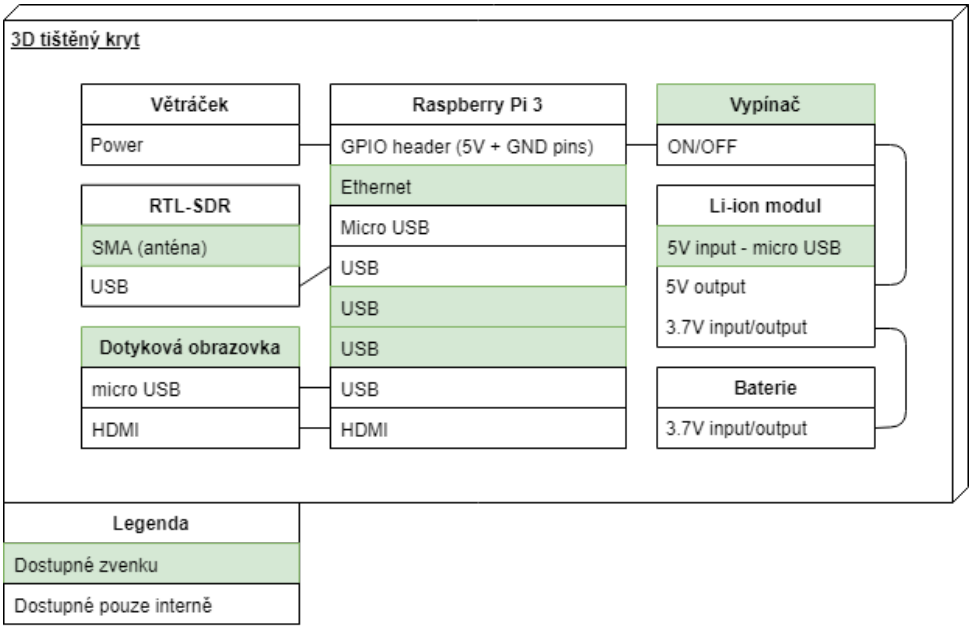
Fusion 360 of Firmy autodesk je varianta jejich CAD programu zaměřená zejména na kolaboraci a na návrh modelů určenách pro výrobu, ať už pomocí 3D tiskáren nebo obrábění, soustružení, řezaní vodním paprskem, laserem či plazmou [**?**]. Umožňuje také simulaci 3D modelů, kde jsou k dispozici nástroje jako pro statické zatížení, tepelnou analýzu, optimalizaci tvaru a podobně. Aplikace vznikla již v roce 2013 a od té doby jsou neustále přidávány nové funkce.
Princip celé aplikace je založen na cloudu, který je používán jako pro ukládání dat, tak pro náročnější výpočty. Samotná klientská aplikace je napsána v WebGL/HTML5 a je vidět snaha o maximální jednoduchost uživatelského prostředí. Kdo již někdy pracoval s jejím velkým příbuzným Autodesk Fusion Inventor, ten jistě najde hodně známých prvků. Podobně jako invertor také podporuje parametrické i přímé modelování [**?**].
Velkou výhodou, kterou jsem často využil při modelování komponent přijímače je také schopnost otevřít téměř libovolný formát 3D modelu, já jsem tak mohl importovat již hotové modely Raspberry Pi, Li-Ion článků nebo různých konektorů. Poslední velmi užitečnou funkcí je již v úvodu zmíněná podpora kolaborace, díky ní jsem mohl svůj návrh průběžně konzultovat s již zmíněným kolegou a provádět jeho úpravy tak aby šel výsledný model bez problému vytisknout.



Obrázek 6.2: Prostředí Autodesk Fusion 360 a navrhnutý model krytu

6.4.2 Model

Tvorbu modelu jsem započal vymodelováním všech komponent. Modeloval jsem je do různé úrovně detailu podle toho jak moc na jejich členění mohl záviset výsledný kryt. Například držak s Li-Ion články stačilo vymodelovat jako přibližný kvádr, do krytu jsem ho toiž jen vlepil na rovnou plochu. Oproti tomu Raspberry Pi které mělo mít některé konektory dostupné zvenku, bylo potřeba mít vymodelované do detailu, naštěstí jsem však našel velkmi podrobný model na internetu a vyhnul se tak zdlouhavé práci.
Po vytvoření modelů komponent následovalo jejich poskládání tak aby všechny potřebné porty byly dostupné zvenku, výsledný kryt se vešel na tisknutelnou plochu tiskárny a také aby dva největší producenti tepla - RTL-SDR a Raspberry Pi nebyly hned vedle sebe. Jako poměrně náročné se ukázalo, zpřístupnění LAN a volných USB portů zároveň s uschováním veškerých propojení komponent uvnitř krytu. Ztohoto důvodu nemá kryt tvar kvádru, ale je na jedné straně vykousnutý.
Výsledný kryt se skládá ze dvou částí. Ve vrchní části je zabudován displej a větráček a v dolní zbytek komponent. Ke spodní části bylo nutné ještě přilepit záslepku otvoru, který vznikl vykrojením původního kvádru. Záslepka byla vytištěna zvlášť, protože by ji bez opor nebylo možné vytisknout jako součást spodního ani horního dílu. Díly jsou spojeny šesti šrouby, celý kryt má tak velkou pevnost a působí robustně.



Obrázek 6.3: Diagram zapojení a dostupnosti portů přehledového přijímače

6.4.3 Tisk

O tisk se postaral můj kolega Jaroslav Bartoš. Použita byla tiskárna Průša MK2 a materiál petg. Tisk byl nastaven na nižší rychlost kvůli menší chybovosti a tisk vrchního dílu tak trval 6 hodin včetně záslepky a spodního 10 hodin. Vytisknutý kryt byl hned na první pokus použitelný bez větších úprav, jediný komponent co nešel zasadit bez úprav krytu byl displej jehož připravený výřez byl asi o pům milimetru menší, než displej samotný a bylo tak potřeba otvor trochu zvětšit.

6.5 Zapojení

Při zapojení se centrálním prvkem stalo Raspberry Pi a k němu jsou připojeny všechny ostatní komponenty. Konkrétní porty a dostupnost komponent a volných portů zvenku lze vyčíst z digramu na obrázku **??**.

Kapitola 7

Software

V této kapitole se budu věnovat veškerému software potřebnému k fungování přehledového přijímače ať už se bude jednat o SW vytvořený třetími stranami nebo mnou. Nejprve představím použité technologie, framworky a programy a poté nástíním jejich vzájemné propojení a logiku fungování celého přijímače.

7.1 Použité technologie a frameworky

7.1.1 GNU Radio

7.1.2 Flask

7.1.3 SocketIO

7.1.4 jQuery

Kapitola 8

Výsledky

Kapitola 9

Závěr

Literatura

[Air] Airspy. What is airspy?

[Blu16] Josh Blum. Lora modem with limesdr, jun 2016.

[Dis17] Paul Disk91. The sigfox radio protocol, nov 2017.

[EB06] Bertalan Eged and Benjamin Babják. Universal software defined radio development platform. In *Dynamic Communications Management*, 2006.

[ITE] ITEAD. Airspy r2.

[Kil13] Taylor Killian. Sdr showdown: Hackrf vs. bladerf vs. usrp, 2013.

[KNI16] MATT KNIGHT. Reversing lora: Exploring nextgeneration wireless. In *GRCon*, 2016.

[MWF] Jay Miller, Justin Wesley, and Matt Frazier. Differential binary phase shift keying.

[Nua] Nuand. bladerf 2.0 micro.

[Nut10] Nutaq. A short history of software-defined radio (sdr) technology, 2010.

[Oss13] Michael Ossmann. Hackrf, an open source sdr platform, 2013.

[Res] Etus Research. Usrp b205mini-i.

[rok16] roklobsta. Welcome to the rtlsdr.org wiki!, 2016.

[RQLT18] Pieter Robyns, Peter Quax, Wim Lamottea, and William Thenaers. A multi-channel software decoder for the lora modulation scheme. In *Conference: 3rd International Conference on Internet of Things, Big Data and Security*, 2018.

[rsa16] rtl-sdr.com admin. Review: Airspy vs. sdrplay rsp vs. hackrf, 2016.

[SC97] T.M. Schmidl and D.C. Cox. Robust frequency and timing synchronization for ofdm. *EEE Transactions on Communications*, 45:1613–1621, dec 1997.

[tap] tapr. Hpsdr mercury tech specification.

[wN13] Kyung wan Nam. Direct down-conversion system with i/q correction. *SLWU085–July 2013*, 2013.

[ZP16] Juan Carlos Zuniga and Benoit Ponsard. Sigfox system description. In *IETF 96*, jul 2016. https://www.ietf.org/proceedings/96/slides/slides-96-lpwan-10.pdf.