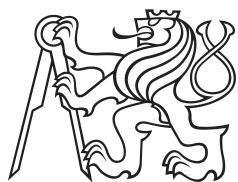


Diplomová práce



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**

Za podporu a pomoc při psaní této práce bych chtěl poděkovat svému vedoucímu Ing. Pavlu Trollerovi. Dále bych chtěl poděkovat své rodině, která mi byla velkou oporou během celého studia i při psaní této práce. A na závěr bych chtěl říct své díky i mému kolegovi Bc. Jaroslavu Bartošovi za konzultace při návrhu a 3D tisk krytu v této práci použitému.

## **Prohlášení**

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

## Abstrakt

Rozvoj sítí pro internet věcí je v poslední době v plném proudu, stejně tak rozvoj softwérově definovaného rádia a jeho použití. Tato práce se zabývá oběma těmito fenomény a následně je kombinuje ve formě přehledového přijímače pro síť internetu věcí založeném na RTL-SDR. V práci popisují všechny potřebné prvky tohoto přijímače jak z hlediska hardware, tak software.

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

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

## Abstract

Internet of things and its networks are big buzzwords of the last few years and the same goes for software defined radio and its new forms of usage. This work deals with both and combines them in form of a scanner of internet of things based on RTL-SDR. All necessary parts, both software and hardware, for realization of such device are described in this thesis.

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

**Title translation:** Scanner/Monitor of IoT radio networks

# Obsah

<b>1 Úvod</b>	<b>1</b>
<b>Část I</b>	
<b>Seznámení s problematikou</b>	
<b>2 Internet věcí</b>	<b>5</b>
2.1 Rozdělení technologií pro IoT	6
2.1.1 Dle využívaných frekvencí	6
2.1.2 Dle architektury sítě	6
2.1.3 Dle rozsahu sítě	7
<b>3 Technologie IoT</b>	<b>9</b>
3.1 LoRa	9
3.1.1 Obecné informace	9
3.1.2 Specifikace	10
3.1.3 Pokrytí	10
3.1.4 Fyzická vrstva (LoRa PHY)	11
3.1.5 Softwarová demodulace	15
3.2 SigFox	18
3.2.1 Obecné informace	18
3.2.2 Specifikace	19
3.2.3 Pokrytí	19
3.2.4 Rádiový protokol Sigfox	20
<b>4 Softwérově definované rádio</b>	<b>27</b>
4.1 Úvod do SDR	27
4.2 Fungování SDR	27
4.3 RTL-SDR	30
4.4 Další dostupná SDR	31
<b>Část II</b>	
<b>Přehledový přjímač</b>	
<b>5 Hardware</b>	<b>35</b>
5.1 RTL-SDR	35
5.2 Raspberry Pi	36
5.2.1 Displej	37
5.3 Napájení	37
5.4 Kryt	39
5.4.1 Autodesk Fusion 360	39
5.4.2 Model	39
5.4.3 Tisk	40
5.5 Zapojení	41
5.6 Testovací zařízení LoRa a Sigfox	41
<b>6 Software</b>	<b>43</b>
6.1 Požadavky a z nich vyplývající software	43
6.1.1 librtlsdr a její utility	43
6.1.2 GNU Radio	44
6.2 Architektura	47
6.3 Přehled používaných souborů	48
<b>7 Průběh práce a výsledky</b>	<b>51</b>
7.1 Postup práce	51
7.1.1 Počáteční testování	51
7.1.2 Vývoj kontrolní aplikace a webového rozhraní	52
7.1.3 Problémy s Raspberry Pi	52
7.1.4 Úprava knihovny gr-lora	53
7.1.5 Podpůrný hardware	53
7.2 Výsledky	53
<b>8 Závěr</b>	<b>55</b>
<b>Bibliografie</b>	<b>57</b>

## Obrázky

3.1 Specifikace LoRaWAN v jednotlivých regionech [5] .....	11
3.2 Pokrytí technologií LoRaWAN ve světě [4] .....	12
3.3 Pokrytí LoRaWAN Českými radiokomunikacemi [31] .....	13
3.4 Ukázka signálu lora [16] (upraveno) .....	14
3.5 Diagonální prokládání při $FS = 7$ a $CR = 4$ , hodnota čipu třetího symbolu je vyznačena tučně. [34] .	15
3.6 Vlevo synchronizace pomocí Schmidl-Cox algoritmu a vpravo algoritmus použitý v [34] .....	16
3.7 Pokrytí Sigfox ve světě[40] .....	20
3.8 Pokrytí Sigfox v České republice [28] .....	21
3.9 Příklad modulace DBPSK [21] .	22
3.10 Modulace amplitudy při změně fáze [9] .....	23
3.11 Struktura rámce Sigfox pro Uplink (nahoře) a konkrétní podoba rámce OOB (dole). Každá buňka reprezentuje půslabiku (4 byty) [9]	24
3.12 Časování jednotlivých opakování uplinku a následující downlink [18]	25
4.1 Implementace rádia na základě SDR[11] .....	28
4.2 Klasifikace SDR dle místa konverze domény[33] .....	29
4.3 Blokové schéma SDR s přímou kvadraturní konverzí RF na BB[22]	30
5.1 Výhody RTL-SDR blog V3 oproti běžným RTL DVB-T [37] .....	37
5.2 Prostředí Autodesk Fusion 360 a navrhnutý model krytu .....	40
5.3 Vnitřní uspořádání komponent .	41
5.4 Diagram zapojení a dostupnosti portů přehledového přijímače .....	42
6.1 Flowgraf pro příjem LoRa .....	46
6.2 Flowgraf pro příjem Sigfox .....	46
6.3 Diagram vzájemné komunikace jednotlivých bloků .....	48
7.1 Přehledový přijímač internetu věcí	54

## Tabulky

3.1 Parametry přenosu Sigfox pro Uplink a Downlink .....	21
3.2 Hodnoty F.TYPE v závislosti na délce zprávy a pořadí opakování [9]	23
3.3 Pole rámce Downlinku (délka pole v bitech) [50] .....	25
4.1 Přehled dospuňých SDR .....	32
5.1 Specifikace Raspberry Pi 3B+ [19]	38



# Kapitola 1

## Úvod

Spojení velkého fenoménu doby Internetu věcí s v poslední době rychle se vyvýjející oblastí softwarově definovaného rádia (SDR) nemusí být na první pohled logické. IoT sítě jdou totiž přesně opačnou cestou než SDR, maximálně specializují hardware aby dosáhl co nejlepšího výkonu při co nejmenší energetické náročnosti. Ve skutečnosti toto spojení však smysl dává, ne proto, že by senzory měli používat SDR, ale kvůli obrovskému množství vznikajících standardů a modulací pro jejichž monitoring a příjem by bylo potřeba stejně velké množství přijímačů. Díky SDR by mohl stačit přijímač jeden. A prvnímu pokusu o přesně takový přijímač se věnuje tato práce. Jejím výledkem by měl být přehledový přijímač pro internet věcí, který zvládne monitorovat okolní zařízení a zachytávat a zobrazovat jimy vysílané zprávy a to včetně údajů o použité frekvenci, sířce pásma nebo přibližné síle signálu.

První část textu této práce se věnuje teoretickému seznámení s problematikou IoT sítí a SDR. V první kapitole je krátce popsán vývoj a směřování IoT a definovány různé kategorizace těchto sítí. V další kapitole se trochu detajněji věnuji dvěma konkrétním sítím LoRa a Sigfox, které jsem zvolil jako první technologie, které by měl přehledový přijímač podporovat. Poslední kapitola této části se zabývá úvodem do světa SDR, jeho fungováním a porovnáním dostupných produktů se zaměřením na RTL-SDR.

V druhé části této práce se již věnuji samotnému přehledovému přijímači. Nejprve je popsán veškerý k realizaci potřebný hardware a způsob jeho použití a pak je řada na software. Zde jsou popsány požadavky na něj a výsledná architektura. Následuje poslední kapitola před závěrem, která shrnuje provedenou práci a dosažené výsledky.



## **Část I**

### **Seznámení s problematikou**



## Kapitola 2

### Internet věcí

Internet věcí je velmi rychle se rozvíjející trend v oblasti telekomunikační techniky. Vzhledem k snaze propojit do sítě stále více a více zařízení, která mají často specifické požadavky je potřeba vytvořit nové technologie, které by lépe vyhovovali tomuto použití. Tento proces již započal a na trhu existuje velké množství variant. Každá technologie je navržená pro určité způsoby využití, a tak se v mnohém liší. Mají však i společné atributy vyplývající z obecných požadavků pro IoT.

Je potřeba aby takové sítě byly velkokapacitní a mohly tak obsloužit velké množství zařízení. Dle odhadů bylo do sítí IoT v roce 2017 připojeno již přes dvacet miliard zařízení a jejich počet zatím bude nadále prudce stoupat. [43] I přesto že povětšinou se jedná o zařízení generující jen malý síťový provoz, celkový provoz sítě je díky jejich velkému množství obrovský.

Dále je, vzhledem k motivaci připojit zařízení co nejvíce, nutné, aby v síti mohla pracovat i velmi jednoduchá zařízení, která jde snadno a levně vyrobit. Jen tak bude možné vybudovat rozsáhlé senzorové sítě o tisících prvků nebo připojit opravdu většinu zařízení každé domácnosti.

Posledním univerzálním požadavkem, kterým se IoT od běžných sítí odlišují jsou velké nároky na energetickou úspornost. Připojená zařízení totiž musí často fungovat nezávisle na rozvodných sítích elektrické energie, a tak musí spoléhat na akumulátory. Ty mají omezenou velikost a kapacitu a často je jejich výměna či dobití nepraktické nebo dokonce nemožné. I přesto však musí být podobná zařízení schopná provozu i několik let.

Právě kvůli tak náročným požadavkům musí dělat standardy pro IoT mnoho kompromisů a každý standard je tak specializován na konkrétní použití. Z toho vyplývá, že nikdy nebude existovat jeden standard na kterém bychom mohli provozovat všechna IoT zařízení. A tady se nabízí použití softwarově definovaných rádií (SDR), ta by mohla komunikovat s IoT sítěmi všech standardů pomocí jednoho jediného zařízení jen změnou softwarové výbavy a šetřit tak náklady i životní prostředí. Prvním krokem v takovém počínání by mohl být přehledový přijímač internetu věcí, který je tématem této práce.

## 2.1 Rozdelení technologií pro IoT

### 2.1.1 Dle využívaných frekvencí

Velkým faktorem při výběru technologie pro použití pro účely internetu věcí jsou používané frekvence. Rádiové spektrum je totiž omezeným zdrojem, a tak jsou pro volné použití vyhrazeny jen určité frekvence, a i pro vysílání v těchto bezlicenčních pásmech existují omezení a pravidla. Další pásmata lze využívat na základě licence od Českého telekomunikačního úřadu. Sítě pro internet věcí jsou provozovány na obou typech frekvencí a z toho vyplývají pro danou síť vždy jak výhody, tak nevýhody.

Výhodou bezlicenčních pásem je především jednoduchost a cena nasazení. Není třeba žádat o žádné licence, čekat na jejich schválení, platit za ně. Na druhou stranu je pak ale nutné počítat s tím, že na stejných frekvencích může vysílat i kdokoliv jiný a způsobovat tak vzájemnou interferenci. Dále je nutné vždy dodržovat závazná pravidla pro použití dané frekvence. To se týká zejména maximálního vysílaného výkonu, což může ohrozit požadovaný dosah. Dalším omezením často bývá maximální povolená doba využívání daných prostředků. Můžeme tak například být omezeni počtem zpráv které je možné za určitý časový úsek přenést.

Náklady na provoz sítě pracující v licencovaném pásmu pak bývají větší, protože se do nich musí započítat poplatky za licenci. Na druhou stranu pak má provozovatel mnohem větší svobodu s nakládáním s daným pásmem a může tak mít vysílače s vyšším výkonem a nemusí se bát rušení od jiných zařízení, než od těch jejichž provoz sám v síti povolí.

### 2.1.2 Dle architektury sítě

Na základě předpokládaného využití je vhodné zvolit i architekturu a topologii sítě.

Nejčastější topologií bývá hvězda, její název je odvozen od tvaru hvězdy, kdy uprostřed je vždy centrální prvek a z něho se rozebíhají paprsky ke všem připojeným zařízením. Ta jsou propojena jen s centrálním prvkem, a ne spolu navzájem. Výhoda této topologie je, že pokud selže nebo se odpojí jedno z připojených zařízení nezpůsobí to žádné komplikace pro ostatní. Dále je v této topologii přítomný řídící prvek, který má většinou řádově vyšší výpočetní kapacitu a dostupné zdroje a může tak sít řídit a jednotlivé uzly mohou být o to jednodušší a úspornější. To také nahrává modelu, kdy provozovatel vytvoří pokrytí určitého území svou technologií a uživatelé se již nemusí o nic starat a jen za poplatek využívají v již vytvořené síti svá koncová zařízení. Na druhou stranu v případě selhání centrálního prvku přestává fungovat celá síť. To lze samozřejmě řešit redundancí takového prvku, avšak to dále přispívá, k již tak vysokým nákladům na budování infrastruktury pro síť tohoto typu.

Pravým opakem v tomto smyslu je síť postavená na smíšené topologii (mesh), kde jsou si většinou všechny prvky rovny a jsou propojeny navzájem mezi sebou každý aspoň s jedním sousedem. Taková síť často nevyžaduje žádnou

předpřipravenou infrastrukturu, v krajiném případě může opravdu jít jen o pouhé rozhození senzorů po krajině. Většinou také síť nevyřadí ani selhání jakéhokoliv jednotlivého prvku, jeho úlohu jednoduše převezme prvek jiný. Na druhou stranu neexistuje centrální prvek, který by měl automaticky přehled o všech ostatních a mohl je tak snadno řídit. Místo toho jsou potřeba složité algoritmy, které na základě informací od jednotlivých uzlů určují vhodné směrování zpráv síti, časování a podobně. I kdyby byly takové algoritmy sebedokonalejší vyžadují spolupráci a tím vyšší energetický výdej všech uzlů. Zároveň má tato topologie často nižší celkovou kapacitu, protože zprávy často musí putovat přes mnoho uzlů, než se dostanou k hraničnímu, který komunikuje s vnějším světem.

### **2.1.3 Dle rozsahu sítě**

Správnou technologii budeme posuzovat i na základě rozsahu. Naše požadavky se zásadně promění v závislosti na tom, zda vytváříme mezinárodní síť pokrývající velké území nebo chceme pokrýt jen jednu domácnost. Pro pojmenování rozsahu sítě jsou standardně používány tyto termíny:

**PAN (Personal area network)** Tento akronym popisuje síť, která zahrnuje osobní prostor člověka. V dnešní době to může znamenat spojení zařízení jako mobilní telefon, chytré hodinky, sluchátka, hrudní pás nebo i senzor cukru v krvi. Často se pro tento typ sítí používají technologie jako Bluetooth, Ant, MiWi nebo Zigbee.

**LAN (Local area network)** Místní síť je název pro síť, která je většinou fyzicky omezena jednou budovou, může to být domácnost, škola nebo menší firma. Typickými zástupci bezdrátových technologií této kategorie jsou WiFi a Z-Wave.

**MAN (Metropolitan area network)** Tento druh sítě propojuje zařízení v rámci větších oblastí jako jsou města a univerzitní kampusy. V tomto rozsahu jsou typickými zástupci WiMax a IQRF.

**WAN (Wide area network)** Síť pokrývající velká území, například celé státy. Zástupci z této kategorie jsou LoRaWAN, Sigfox nebo NB-IoT. A právě na tyto sítě se budu soutředit při vývoji přehledového přijímače.



## Kapitola 3

# Technologie IoT

Po předchozím průzkumu provedeném v rámci předmětu B2MPROJ6 - Projekt jsem se rozhodl zaměřit na dvě technologie pro IoT, jedná se o LoRaWAN a Sigfox. Tyto technologie jsem zvolil, protože jako jediné mají rozvinutou infrastrukturu v ČR a zároveň vysílají v pásmu, které se nachází v rozsahu RTL-SDR, které jsem plánoval pro přehledový přijímač použít.

V následující kapitole tyto technologie představím jak z hlediska obecného, tak z hlediska rádiové vrstvy.

### 3.1 LoRa

#### 3.1.1 Obecné informace

LoRaWan je klasická ukázka Low-Power Wide-Area Network (LPWAN) navržené pro prostředí internetu věcí. Vyniká tedy hlavně v oblasti životnosti na baterii, dosahu, ceně a celkové kapacitě sítě.

Každá implementace této technologie jako LPWAN se skládá ze dvou částí. LoRaWAN definuje komunikační protokol a síťovou architekturu a LoRa zajišťuje fyzickou vrstvu přenosu [4].

LoRa (Long Range) je fyzická vrstva používaná v sítích LoRaWAN. Je, jak již název napovídá, navržena pro komunikaci na velké vzdálenosti a na rozdíl od mnohých starších systémů, které využívají klíčování frekvenčním posunem (FSK), používá chirp rozprostřené spektrum [5]. Díky tomu je možné pomocí jen několika málo bran nebo základových stanic pokrýt velká území. Například celé město jednou jedinou branou. Z toho plyne možnost velmi rychle, s malými náklady a infrastrukturou pokrýt území celé země.

LoRaWAN definuje komunikační protokol a síťovou architekturu. Zvolená topologie síťové architektury je hvězda. To má oproti smíšené topologii výhodu v tom, že každý uzel sítě přenáší jen svoje data a nemusí tak využívat své zdroje pro přenos pro něj irrelevantních informací. Zároveň je taková síť jednodušší na správu a má vyšší celkovou kapacitu. Na druhou stranu je dosah omezen na dosah jednoho uzlu.

Takováto architektura sítě by se dala připodobnit klasické buňkové mobilní síti, ale na rozdíl od ní není nikdy uzel přiřazen specifické základové stanici, při pohybu uzlu tedy odpadá handover. Místo toho paket z každého uzlu při-

jmou všechny brány v dosahu a předají ho na společný server. Ten vyhodnotí přijaté pakety, duplikované zahodí, zvolí ideální bránu pro odpověď, adaptivní přenosovou rychlosť a podobně.

Další úspora baterie spočívá v časování komunikace. Uzly v síti nejsou synchronizované s bránami. K řízení provozu je použit protokol ALOHA. Zařízení tak vysílají jen pokud mají data k odeslání nebo v naplánovaných intervalech a nemusí se budit pro udržování synchronizace. Pro komunikaci směrem k uzlům existují tři způsoby podle tříd zařízení. Třída A má pro downlink definovaná okna jen vždy po ukončení uplinku. Třída B má navíc naplánovaná další periodicky se opakující okna, kdy je zařízení připraveno přijímat a třída C určená zejména pro zařízení, pro která není energetická náročnost tak kritická, je připravena na příjem kontinuálně.

Pro zachování velké celkové kapacity i při poměrně malém počtu bran musí tyto být schopné přijímat data od mnoha uzlů zároveň. Toho je docíleno využitím adaptivních rychlostí přenosu a vícekanálovým a vícemodemovým vysílačem. Díky použití rozprostřeného spektra lze také využít ortogonality signálů používajících jiný činitel rozprostření. Pokud by ani tak celková kapacita nestačila, je technologie LoRaWAN velmi dobře horizontálně škálovatelná. Přidáním dalších bran v oblasti mohou zařízení vysílat menším výkonem a vyšší přenosovou rychlosťí a tím omezit přeslechy a kapacita tak může být rozšířena 6-8krát.

I přes důraz na nízkou energetickou náročnost LoRaWAN řeší i bezpečnost provozu. Na síťové úrovni je řešena autenticita uzlů a bran a na aplikační úrovni je použito AES šifrování pro skrytí dat před neoprávněným odposlechem.

### **3.1.2 Specifikace**

- modulace: CSS - Cvrlikavé rozprostřené spektrum
- maximální velikost zprávy: 256 Bajtů
- počet zpráv za den: neomezený
- citlivost: -140dBm
- dosah v terénu: až 40km v terénu, 15 km v příměstském prostředí a 2 – 5 km ve městě
- zabezpečení: šifrování AES128
- výdrž na bateriích: 5-15 let (podle hustoty komunikace) [27]

### **3.1.3 Pokrytí**

Vzhledem k tomu, že LoRa aliance je otevřená a sestává se ze stovek členů zahrnujících mezinárodní telekomunikační operátory i výrobce techniky a senzorů a její účel je vytvořit mezinárodní síť s možností roamingu zařízení, je pokrytí téměř celosvětové. Sama aliance na svých stránkách publikuje tuto mapu 3.2. V Česku spravuje komerční LoRaWAN síť operátor České Radiokomunikace a pokrývá většinu území, viz 3.3

	Europe	North America	China	Korea	Japan	India
<b>Frequency band</b>	867-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz
<b>Channels</b>	10	64 + 8 +8				
<b>Channel BW Up</b>	125/250kHz	125/500kHz				
<b>Channel BW Dn</b>	125kHz	500kHz				
<b>TX Power Up</b>	+14dBm	+20dBm typ (+30dBm allowed)				
<b>TX Power Dn</b>	+14dBm	+27dBm				
<b>SF Up</b>	7-12	7-10				
<b>Data rate</b>	250bps- 50kbps	980bps-21.9kbps				
<b>Link Budget Up</b>	155dB	154dB				
<b>Link Budget Dn</b>	155dB	157dB				

**Obrázek 3.1:** Specifikace LoRaWAN v jednotlivých regionech [5]

### ■ 3.1.4 Fyzická vrstva (LoRa PHY)

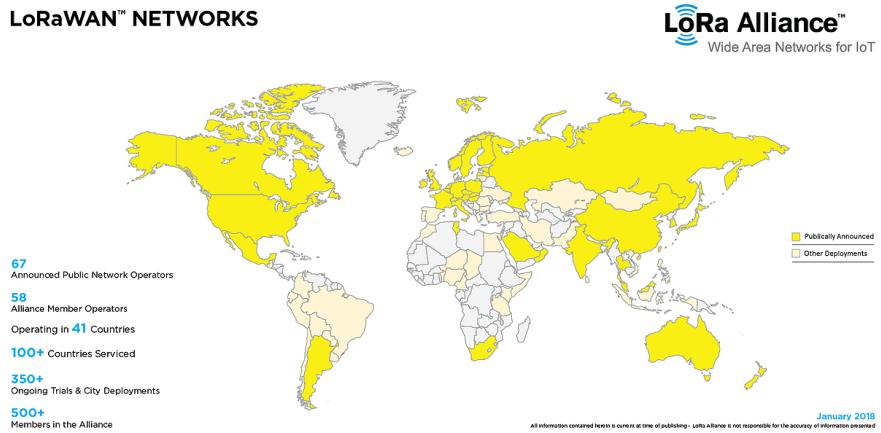
Následující část se zabývá protokolem LoRa. Vzhledem k tomu, že je protokol LoRa proprietární a neexistuje k němu tak veřejně dostupná dokumentace, vycházím zejména ze dvou prací [16] [34]. Tyto práce napsali výzkumníci, kteří pomocí reverzního inženýrství protokol analyzovali. Kromě reverzního inženýrství disponovali také zdroji ze stránek vlastníka technologie Semtech, které již v tuto chvíli nejsou k dispozici.

### ■ Modulace

Modulační schéma LoRa je založeno na Chirp Spread Spectrum (Cvrlikající rozprostřené spektrum) modulaci [12] a definuje jeden "cvrk" jako jeden symbol. Cvrk lze popsat jako signál, který v průběhu svého trvání lineárně mění svoji frekvenci a to buď na nižší nebo na vyšší a rozlyšujeme tak cvrk nahoru a dolů.

Část spektra kterou cvrk zabírá je jeho šířka pásma, v Evropě to může být 125 kHz nebo 250 kHz. To kolik kolik daný symbol/cvrk nese dat určuje činitel rozprostření, ten nabývá hodnot 7-12 bitů. Rychlosť jakou se frekvence cvrku mění (první derivace frekvence) je dáná vztahem  $k = \frac{BW}{2SF}$  kde  $k$  je rychlosť cvrku,  $BW$  je šířka pásma a  $SF$  činitel rozprostření.

Celkový počet možných symbolů je dán vztahem  $2^{SF}$  a konkrétní hodnota je na cvrk modulována pomocí časového posuvu cvrku. To znamená, že cvrk začíná na jiné frekvenci než cvrk základní a ve chvíli kdy dojde na konec šířky svého pásma pokračuje znova z druhé hranice pásma. Hodnota tohoto posuvu tak určuje hodnotu symbolu a každý cvrk je tak v podstatě kvantován na



**Obrázek 3.2:** Pokrytí technologií LoRaWAN ve světě [4]

$2^{SF}$  časových intervalů, těmto intervalů říkáme čipy.

### ■ 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 dobré 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  $\{0, 1\}^{SF \times (4+CR)}$ , 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  $\{0, 1\}^{7 \times 8}$ , příklad je na obrázku 3.5. K získání kódového slova je pak potřeba číst bity po diagonále matice. Na rozdíl od patentu LoRy, 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  $\{0, 1\}^{SF-2 \times (4+CR)}$  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 rychlosť přenosu dat ve prospěch odolnosti proti šumu. Hlavička fyzické vrstvy LoRa



**Obrázek 3.3:** Pokrytí LoRaWAN Českými radiokomunikacemi [31]

je v tomto módu vysílána vždy, kdežto užitečná data jen v případě použití SF 11 nebo 12.

## ■ Kódování

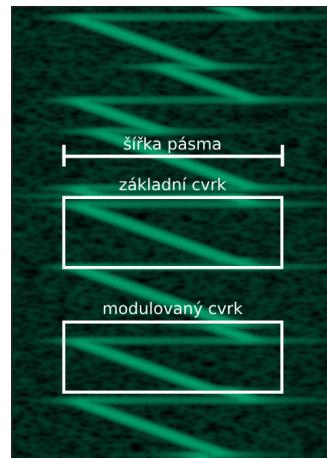
Po přečtení kódových slov z prokládací matice mají tato délku  $4 + CR$ . Kvůli zamezení vzniku stejnosměrné složky byla slova v části rámce s užitečnými daty XOR-ována 9-bitovým lineární posuvným registrem se zpětnou vazbou (LSFR Linear feedback shift register). Tento proces se nazývá whitening. A proto musí po synchronizaci projít stejným procesem znova. Přesný algoritmus není v patentu určen a jeho výběr je tedy na každém výrobci zvlášť. Na několika testovacích zařízeních [34] reverzním inženýrstvým zjistilo použití upraveného  $4/(4 + CR)$  Hammingova kódu. Ve výsledku tak z každého kódového slova po dekódování získáme 4 bity dat. Ty jsou pak naparovány do struktury rámce lora.

## ■ Struktura rámce

Na fyzické vrstvě LoRa definuje rámec jako strukturu složenou z následujících polí. Pole jsou uvedena ve stejném pořadí jako v rámci.

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

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



**Obrázek 3.4:** Ukázka signálu lora [16] (upraveno)

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

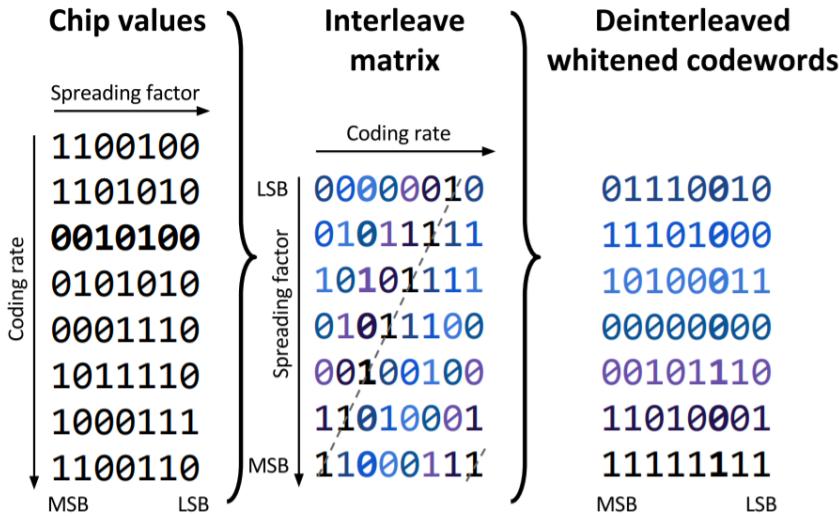
**Hlavička (nepoviná)** Hlavička obsahuje délku užitečných dat, použitou přenosovou rychlosť, indikuje použití cyklického redundantního součtu (CRC - Cyclic redundancy check) a jendobajtovou kontrolní sumu hlavičky. Pro modulaci hlavičky je vždy použito  $CR = 4$  a mód redukované rychlosti. Pokud hlavička vysílána není (implicitní mód) musí mít jak přijímač tak vysílač předem stejně nastavené 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.

## ■ 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 vyplývá že hlavička se musí vejít do mřížky  $\{0, 1\}^{7-2 \times 8}$  a to odpovídá 5 kódovým slovům. Každé slovo má 8 bitů a dohromady je to bitů 40. Jakékoli zbývající byty jsou použity pro užitečná data.

Po dekódování díky redundantním bitům dostáváme  $40 \frac{4}{8} = 20$  bitů nebo 2,5 bajtu. V [34] 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.



**Obrázek 3.5:** Diagonální prokládání při  $FS = 7$  a  $CR = 4$ , hodnota čipu třetího symbolu je vyznačena tučně. [34]

### 3.1.5 Softwarová demodulace

[34] dokázali implementovat kompletní PHY vrstvu LoRa ve framework 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 pospán princip fungování.

#### 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 [38]:

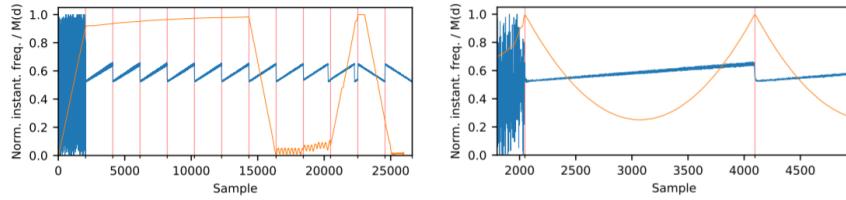
$$P(d) = \sum_{m=0}^{L-1} (x_{t+m}^* x_{t+m+L}) \quad (3.1)$$

$$R(d) = \sum_{m=0}^{L-1} |x_{t+m+L}|^2 \quad (3.2)$$

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

$$M(d) = \frac{|P(d)|^2}{R(d)^2} \quad (3.3)$$

Časová metrika  $M(d)$  v podstatě počítá normalizovanou autokorelací délky  $L$  přes dva symboly, maximum bude mít ve cvhlí kdy v signálu budou za sebou



**Obrázek 3.6:** Vlevo synchronizace pomocí Schmidl-Cox algoritmu a vpravo algoritmus použitý v [34]

dva totožné symboly. Díky tomu, že oba symboly jsou chybami způsobenými přenosem (interference, frekvenční odchylka nosné (CFO - Carrier frequency offset), odchylka vzorkovací frekvence) ovlivněny stejně, tak tyto chyby téměř neovlivní výsledek korelace.

Přestože tento algoritmus detekuje preambuli velmi dobře, není vzhledem k plošině maxima dostatečně přesný pro signály LoRa. Aby [34] tenhle problém vyřešili museli vymyslet nové řešení. To se zakládá na použití Schmidl-Coxovi metriky pro přibližné určení okna ve kterém se nachází druhý symbol preamble 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 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.

Výsledek je na obrázku 3.6. Poslední součástí tohoto řešení je určení prahové hodnoty maxima korelačního koeficientu okamžitých frekvencí lokálně generovaného cvrku a přijátého. Pokud je tato hodnota menší než prahová je daný rámec zahozen, protože se buď jedná o falešně pozitivní detekci rámce nebo o nepovedenou synchronizaci.

## ■ Demodulace

Po úspěšné synchronizaci následuje fáze demodulace. Při příjmu modulovaného cvrku s neznámým časovým posuvem  $x(t + \hat{t})$ , může být hodnota cvrku zrekonstruována navzorkováním signálu vzorkovací frekvencí chipů a výpočtem:

$$i = \text{Gray}(\arg \max(|FFT(x(t + \hat{t}) \odot \overline{x(t)})|)) \quad (3.4)$$

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

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 softwaru příliš náročná operace a tak [34] 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 frekvenci  $\omega[t] = \frac{d\varphi[t]}{dt}$ . 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  $f$ :

$$D_t[\omega[t]] = \omega[t+1] - \omega[t] \quad (3.5)$$

Tuto operaci si lze představit jako filtr horní propust 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 špičky v místech přechodu mezi vysokou a nízkou frekvencí. Přítomnost takových špiček indikuje časový posun  $\hat{t}$ , nepřítomnost naopak idíkuje 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 průběhu 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  $\Delta t$  je chyba časování na symbol. Prvním krokem je synchronizace popsaná v 3.1.5. 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 3.1.5 a je tak získána hodnota chipu  $i$  a časového posunu  $\hat{t}$ .
2. Na přijímači je lokálně vygenerován základní 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 definovat takto  $\Delta t = \hat{t}_1 - \hat{t}$ . Nyní stačí na přijímači 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 korekce zpoždění ale také by zvyšovali náročnost výpočtu.

## ■ 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 [16] a [7] 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 přímo 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 byty umístěny na jiných pozicích než je běžné a to na indexech 0,1,2 a 3 bajtu místo indexů 1,2,3 a 5. Po extrakci datových bitů jsou pak pomocí paritních bitů detekovány a případně opraveny chyby a tím získána původní data.

## ■ 3.2 SigFox

### ■ 3.2.1 Obecné informace

Sigfox je další LPWAN síť. Původně tato technologie vznikla ve Francii odkud se rozšířila do zbytku světa. Stejně jako LoRaWAN používá hvězdicové architektury a má i podobné využití a cíle. Na rozdíl od LoRaWAN však nevyužívá rozprostřené spektrum, ale vysílání na velmi úzké šířce pásmu.

Stejně je i to, že nepoužívá synchronizovanou síť a šetří tak zdroje normálně na synchronizaci vynakládané a místo synchronizace je použit náhodný přístup k médiu, kdy zařízení každou zprávu pošle třikrát v různých časech a na různých frekvencích. Na rozdíl od LoRaWAN však nemá třídy zařízení, a tak jediná možnost pro downlink je potom co zařízení ukončí uplink.

Dalším omezením je povolený počet zpráv, který je pro koncová zařízení limitován na obsazení pásmo 1 % času což při dvanácti bytových zprávách odpovídá 144 zprávám denně [41]. Brány mohou obsazovat médium po 10 % času. Při maximálním počtu zařízení to odpovídá 4 zprávám denně na každé zařízení. Tato omezení vyplývají z použité technologie a použití nekomerčních frekvencí na které není potřeba žádná licence.

Architektura sítě je velmi podobná LoRaWAN. Koncová zařízení komunikují přes Sigfox s bránami, které jsou připojeny přes veřejný internet k serverům Sigfox, které zajišťují správu stanic, spektra a celkového fungování sítě, zpracování zpráv, ukládání dat a také slouží k přístupu do sítě skrz webový portál nebo API. Servery Sigfox také zajišťují účtování a případnou analýzu dat.

Bezpečnost je zajištěna na několika vrstvách. Za prvé zařízení v síti nejsou přímo dostupná z internetu, což slouží podobně jako firewall. Za druhé je systém odolný proti replay útokům, díky použití unikátního ID pro každou

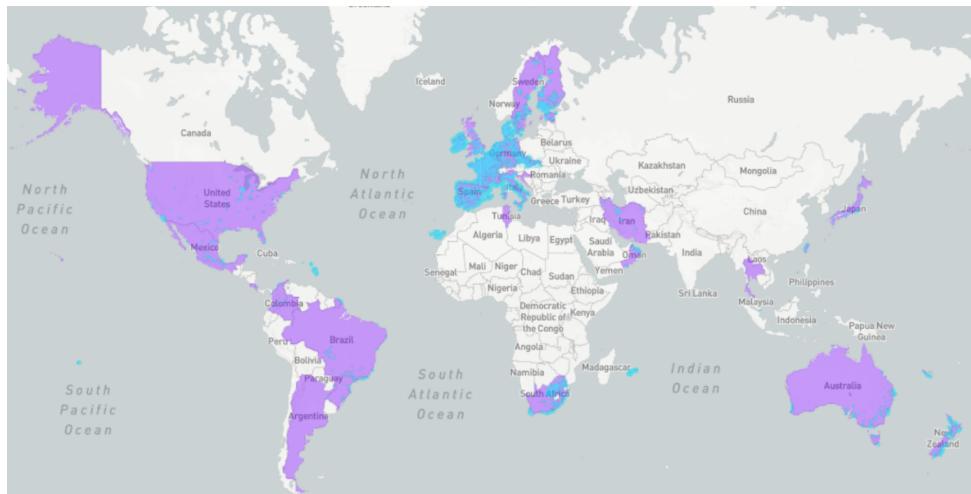
zprávu. Každé zařízení má také unikátní symetrický autentizační klíč, pomocí kterého je generován Message Authentication Code (MAC), který zajišťuje autenticitu a integritu každé zprávy. V případně nutnosti zašifrovat samotné zprávy je možné použít vlastní šifrování na aplikační úrovni nebo využít zabudovaný systém na bázi AES [39]. Od brány dál směrem k serverům a uživateli je použito standartní zabezpečení používané na internetu.

### 3.2.2 Specifikace

- technologie: UNB (Ultra Narrow Band)
- modulace: DBPSK
- způsob příjmu: bez synchronizace, MIMO
- velikost zprávy: 0-12 Bytů (96 bitů)
- rychlosť přenosu: 100 bitů/s
- doba přenosu a zpracování: 4-6 s
- frekvence: 868MHz (ETSI), 915 MHz (FCC)
- počet zpráv za den: 144
- maximální počet zpráv na BTS denně: 9 000 000
- vysílací výkon: 25mW / 14 dBm
- budget link: 162 dB
- zpětný kanál: 4 zprávy po 8 Bytech denně
- dosah v terénu: až 50 km v terénu, 3 km ve městě pro indoor
- spotřeba: 5 mA – 45 mA při vysílání, 0 mA v klidu
- výdrž na bateriích: 5-15 let (až 20 let na dvě AA baterie)
- zabezpečení: hash, šifrování možné na aplikační úrovni
- SLA: 99 % [29]

### 3.2.3 Pokrytí

V rámci celosvětového měřítka je SigFox rozvinut zejména v Evropě, kde je nejsouvislejší pokrytí. Dále se je síť ve výstavbě ve většině zemí Ameriky a v Austrálii. Na mapě jsou fialově vyznačeny země, kde probíhá budování sítě a modrou aktuální pokrytí. V České republice provozuje síť SigFox operátor SimpleCell ve spolupráci s operátorem T-mobile CZ. S výstavbou sítě začali již v létě 2015 v rámci testovacího provozu. Aktuálně pokrývají 96 % obyvatelstva a 94 % území a v plánu mají pokrýt 98 % obyvatelstva a 96 % území ČR.



**Obrázek 3.7:** Pokrytí Sigfox ve světě[40]

[42]. V mapě jsou červenou barvou označena místa, která pokrývají tři a více bran, což je ideální stav. Zelenou barvou jsou označena místa pokryta dvěma branami a modrou ta jen jednou.

### ■ 3.2.4 Rádiový protokol Sigfox

Stejně jako v případě LoRa je tato technologie uzavřená a chráněná patentem. Neexistuje tedy mnoho detailních informací o konkrétním fungování protokolu. V následujících řádcích jsem vycházel hlavně ze snah reverzního inženýrství provedeném v rámci [9]. Základním poznávacím známením technologie Sigfox je pužití velmi úzkého pásma (UNB - Ultra Narrow Band). To dovoluje vysílání na větší vzdálenosti díky úzkému zaměření vysílačního výkonu každého zařízení. Navíc díky použití UNB může na jednom místě operovat velké množství zařízení bez přílišného vzájemného rušení.

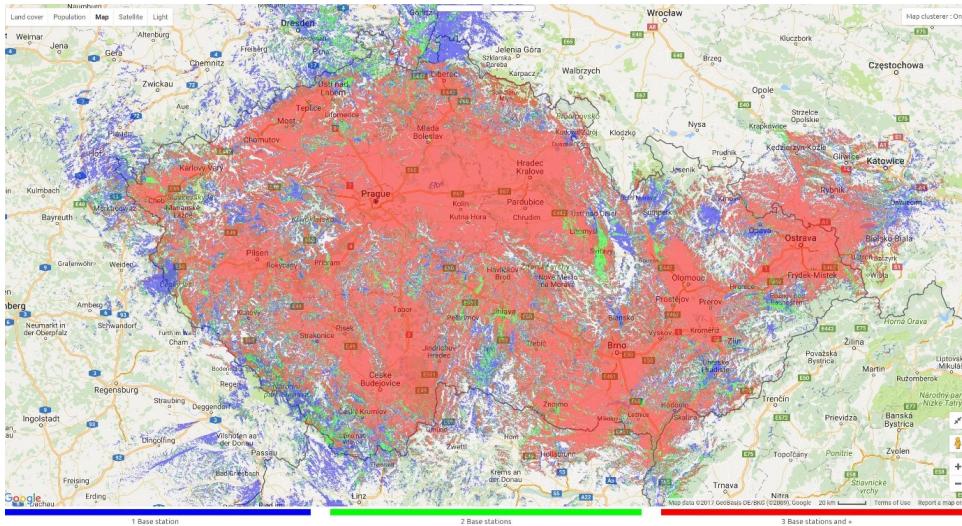
Sigfox používá různá modulační schémata pro uplink a downlink, důvodem je optimalizace využití spektra. Různá je i šířka pásma, přenosová rychlosť, vysílací výkon i struktura rámce. Popíšu tedy oba směry zvlášť a zaměřím se přitom na parametry používané v Evropské unii. V jiných regionech se parametry lyší, důvodem jsou různé alokace spektra.

### ■ 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 neurčuje hodnotu 0 a 1 přímo z fáze, protože to by vyžadovalo synchronizaci a použití referenční fáze, ale poradí si i bez ní. Toto je možné díky tomu, že symbol neurčuje konkrétní fáze, ale její změna, odtud také plyne její pojmenování.

Na obrázku 3.9 je znázorněn příklad fungování této modulace. Pokud nena-

### 3.2. SigFox



**Obrázek 3.8:** Pokrytí Sigfox v České republice [28]

	Uplink	Downlink
Šířka pásma	100 Hz	1500 Hz
Přenosová rychlosť	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 %

[49]

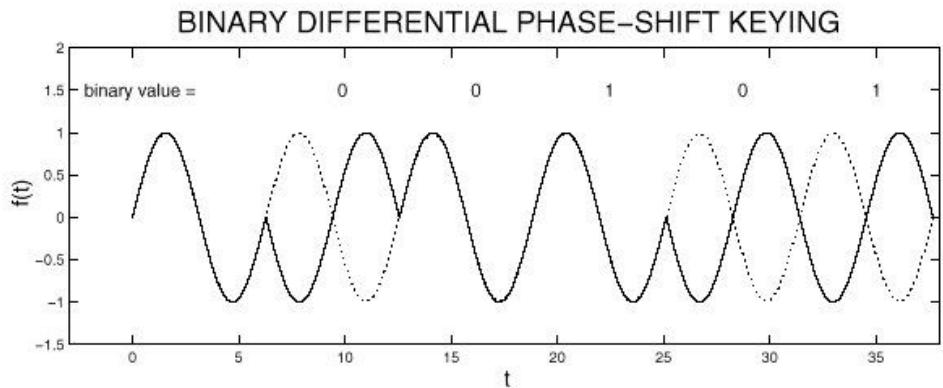
**Tabulka 3.1:** Parametry přenosu Sigfox pro Uplink a Downlink

stává 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 rychlosť 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 používá 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 proměností fáze.

V případě evropské verze je přenosová rychlosť uplinku 100 bit/s a frekvence signálu 868,1 Mhz. Z toho vyplívá že každý bit/symbol odpovídá 8 681 000 periodám. Velké množství period využívá přijímač pro statistické určení posunu fáze každého bitu. V případě 1 bude celá sada mít fázi stejnou a v případě 0 se fáze v prostředí 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



**Obrázek 3.9:** Příklad modulace DBPSK [21]

okolo posunu fáze modulována amplituda. Grafické znázornění této modulace je na obrázku 3.10.

### ■ Struktura rámce uplinku

Na fyzické vrstvě 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.

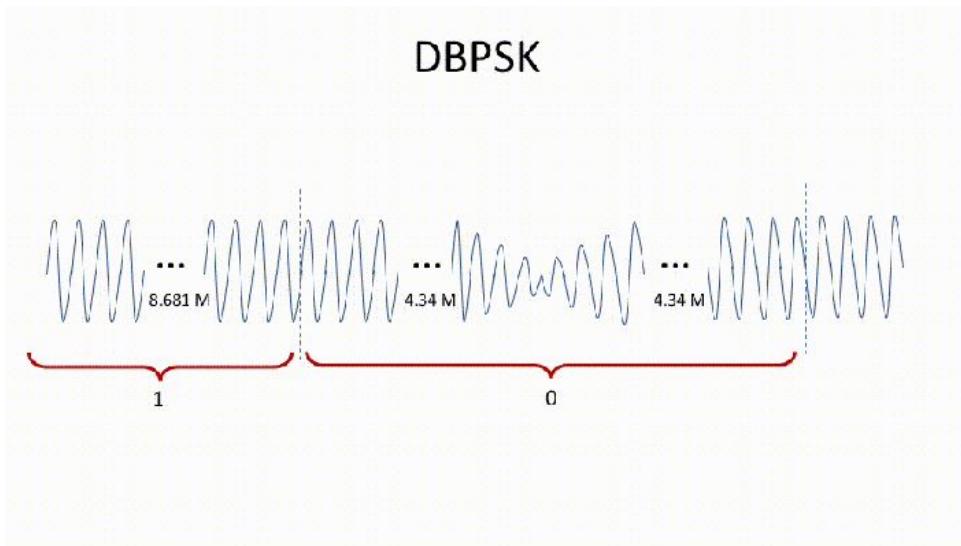
Při druhém opakování hodnota každého bitu závisí na dvou předchozích, pokud jsou stejné tak je hodnota aktuálního bitu prohozena. Při třetím opakování závisí hodnota aktuálních dvou bitů vždy na hodnotě dvou předchozích, pokud je ob jeden bit dozadu nula převrací tyto svoji hodnotu.

Následkem použití jiného kódování pro každé opakování, není i přes identický obsah nikdy vysílaný rámec třikrát. Změny fáze jsou vždy na jiných místech, to může pomoci při příjmu zejména v podmírkách 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 nečinnosti i během vysílání, teplotu a 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 přijímač k synchronizaci hodin aby později korektně identifikoval začátek každého bitu.

Synchronizace je ukončena ve chvíli, kdy se vzor změní z 01 na 00 nebo



**Obrázek 3.10:** Modulace amplitudy při změně fáze [9]

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	-	-

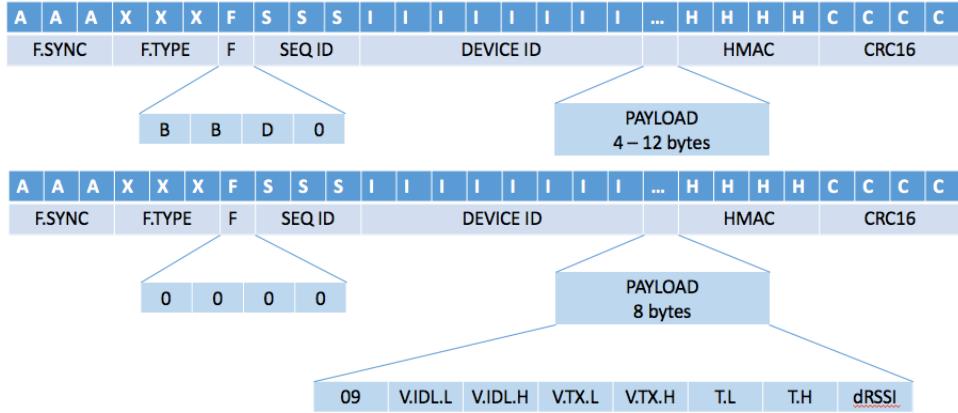
**Tabulka 3.2:** Hodnoty F.TYPE v závislosti na délce zprávy a pořadí opakování [9]

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 rozpoznat 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 [9] vyplývá tabulka 3.2. 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 bajtů, tak bude odeslána v osmi bajtovém rámci a dva bajty tak budou přidány, to je vyjádřeno hodnotou prvních dvou bitů F.FLAGS a jejich hodnotu 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



**Obrázek 3.11:** Struktura rámce Sigfox pro Uplink (nahoře) a konkrétní podoba rámce OOB (dole). Každá buňka reprezentuje půslabiku (4 bity) [9]

downlink odpověďt. 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 znovaodeslání zachycených zpráv. Limitem této ochrany je, že se SEQ.ID každých 2048 zpráv opakuje a tak v určitých situacích nemusí backend znovaodeslatou 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.

**HMAC** HMAC (hash-based message authentication code) je druh MAC proježí 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.

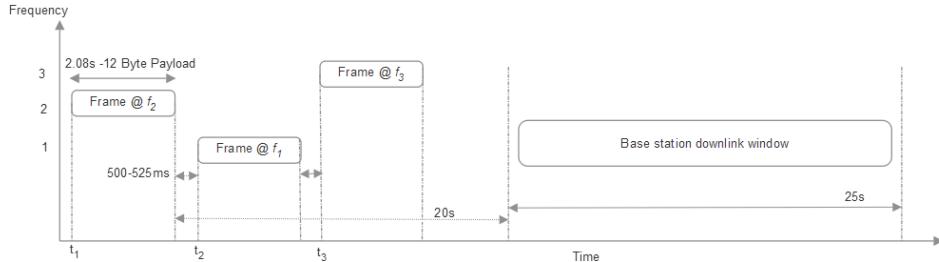
**CRC** Poslední pole je CRC s délkou 16 bitů.

## ■ Downlink

Pro komunikaci směrem ze sítě do zařízení Sigfox používá modulační schéma GFSK, vyšší výkon, širší pásmo a navíc má i větší poměr vysílačního času, viz 3.1. 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 zvlášt. V

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

**Tabulka 3.3:** Pole rámce Downlinku (délka pole v bitech) [50]



**Obrázek 3.12:** Časování jednotlivých opakování uplinku a následující downlink [18]

Evropě po započtení všech omezení vychází pro každé zařízení 140 Uplink zpráv denně a jen 4 Downlink zprávy. Přesný obsah rámce ani další detaily nejsou k dispozici. Pouze lze něco málo vyčíst z tabulky 3.3.

Zprávy dowlinku každé sigfox zařízení přijímá pouze v okně následujícím uplink. Toto okno, kdy je zařízení na příjmu je aktivní pouze pokud měl v poli F.FLAGS třetí bit hodnotu jedna a začíná 20 s po odvysílání prvního rámce, trvá 5 s. Celkové časování je na obrázku 3.12.



## Kapitola 4

### Softwérově definované rádio

#### 4.1 Úvod do SDR

Značná část historie použití rádiových vln využívala rádia, která byla implementována čistě v hardwaru. Pro každou změnu 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 později i rádia SDR, která mohou být použita univerzálně díky digitálnímu zpracování signálu (DSP -Digital signal processing).

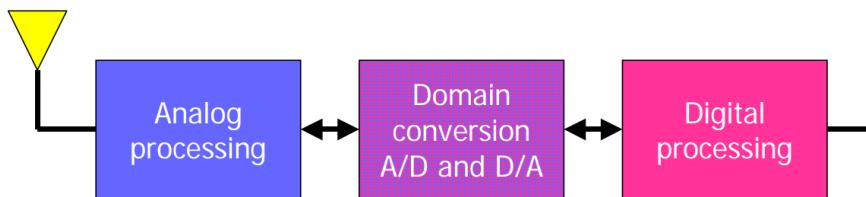
První známe nasazení SDR v praxi bylo v armádě, jednalo se o systém SPE-AKeasy. Systém vyvinula DARPA v roce 1991 a umožňoval díky softwarové implementaci komunikovat prostřednictvím 10 různých vojenských protokolů na frekvencích od 2 MHz po 2 Ghz. SPEAKeasy byl připraven i na přidání dalších modulací a protokolů. O rok později již vyšel první článek o SDR na IEEE, napsal ho Joe Mitola a je tak považován za kmotra SDR. [24]

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 dokáží přizpůsobit aktuálnímu stavu spektra, což vzhledem jeho stále většímu obsazení i z důvodů rozvoje sítí pro internet věcí bude stále důležitější.

#### 4.2 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 rámcem (application framework). Pokud je použity jednotný standard tak tyto části mohou tvořit univerzální a znovupoužitelné komponenty a tak ušetřit prostředky a usnadnit jakékoli 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

**Obrázek 4.1:** Implementace rádia na základě SDR[11]

armádních SDR. [11] Jinou možností je de facto standard pro rádio amatéry - GNU Rádio, kterému je v této práci věnována samostatná sekce 6.1.2. 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 4.1

Front-end se stará o převedení rádiového signálu na frekvenci a šířku pásma, kterou dokáže zpracovat digitální back-end. Tvoří ho analogové zesilovače, směšovače filtry a oscilátory.

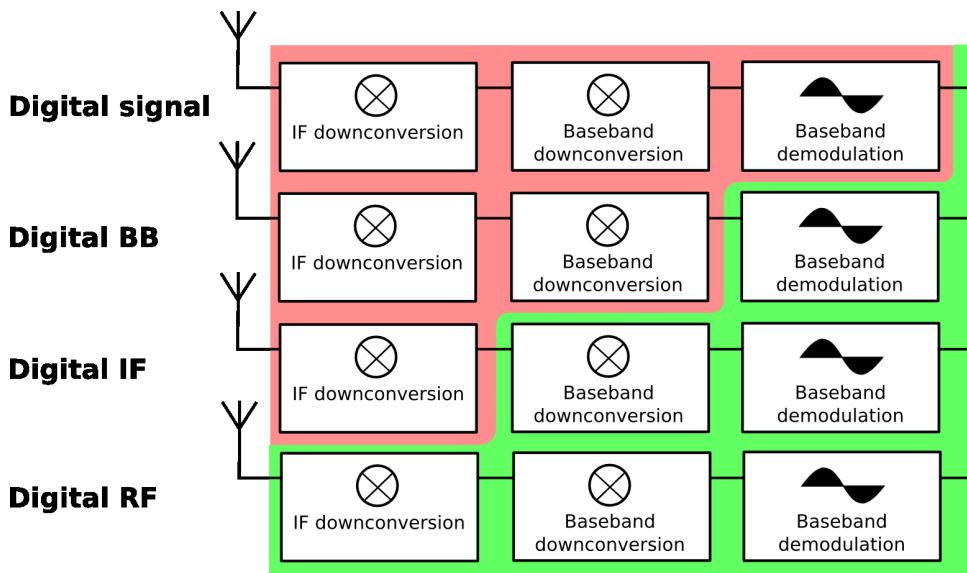
Konvertor domény jak již název napovídá převádí analogový signál na digitální a obráceně. Souži mu k tomu vysokorychlostní širokopásmové A/D a D/A převodníky, jejich vlastnosti zásadně ovlivňují možnosti výsledného SDR.

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 4.2 a podle toho lze systémy kategorizovat.

**Digitální signál (Digital signal)** V toto případě se v podstatě nejedná 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 demodulaci.

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



Obrázek 4.2: Klasifikace SDR dle místa konverze domény[33]

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

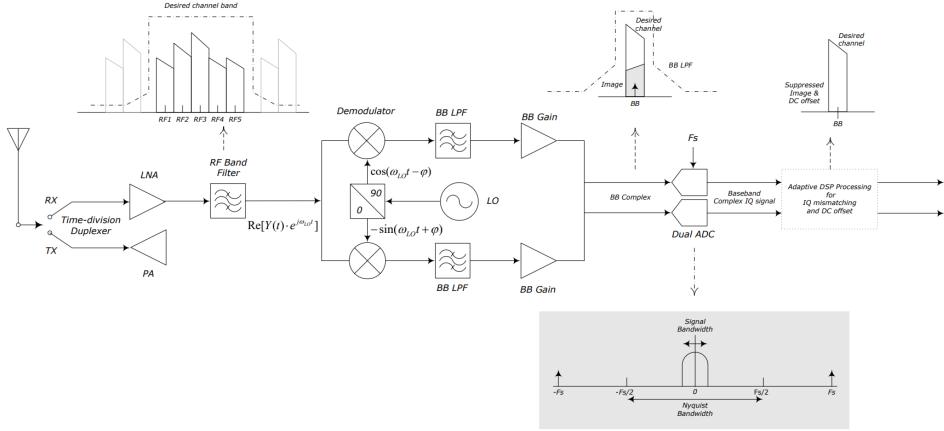
**Digitální rádiová frekvence (Digital RF)** I při vzorkování přímo RF je potřeba signál nejdříve zesílit a filtrovat až poté je možný jeho převod pomocí ADC. Následně probíhají všechny úkony již v SW. Příkladem systému z této kategorie může být HPSDR Mercury [46] .

Toto rozdelení však nepočítá s přímou kvadraturní konverzí RF. V takovém případě se RF signál s reálnými hodnotami po zesílení a filtraci smíchá s výstupem oscilátoru s komplexním výstupem (sínus a kosinus), prožene fitrem dolní propust který odstraní vysoké frekvenční komponenty jako boční pásmo vzniklé smíšením a poté navzrokuje dvěma ADC.[33]

Popsaný postup je výhodný zejména díky své jednoduchosti, není potřeba filtrování IF a cena tak může být nižší. Další výhoda spočívá v tom, že převodníky jsou dva a vzhledem k tomu, že nevzorkují stejný signál může být jejich vzorkovací frekvence poloviční oproti jednomu převodníku.

Mezi nevýhody a také důvody proč se dříve používali slotičejší systémy patří možnost projevení se zrcadlového obrazu signálu a stejnosměrné složky v přijatém signálu. Obě tyto vady jsou způsobeny rozdílem ve fázi a amplitudě mezi kanály I a Q . Tyto vady výrazně degradují přijatý signál avšak je možné je efektivně eliminovat korekcí rovnováhy I/Q 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 4.3:** Blokové schéma SDR s přímou kvadraturní konverzí RF na BB[22]

### 4.3 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 k dispozici velmi výkonný HW na kterém prováděl 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 300-500 kč. Celé to začalo v roce 2008, kdy Realtek představil chipset RTL2832U s DVB-T COFDM demodulátorem, hlavním účelem tohoto čipu a na něm postavených USB donglů byl příjem evropského standardu pro pozemní digitální televizní vysílání DVB-T. 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 ovladače tohoto DVB-T donglu odposlouchával USB pakety během používání FM aplikace pro Windows. Eric zjistil, že narozdíl od příjmu 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 software 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. [36] [8]

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, Odpolech GSM, Sledování letadel pomocí ADSB, Generátor nahodných čísel, Příjem dat meteorologických balónů, Sledování meteoritů, nebo Odpolech vysílaček městské policie

## 4.4 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 příjmu 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 světováho dne již v roce 2008. V současnosti začínají ceny na částkách 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 Radiu a Matlab SimuLinku a zvládá i vysílat. [32]

**bladeRF** Již v roce 2012 v reakci na RTL-SDR začal vývoj bladeRF. Momentálně se nachází již ve své druhé verzi a zajímavostí je, že jedna z variant obsahuje i FPGA pro HW akceleraci digitálního zpracování. Cena začíná na 11 tisících za verzi bez FPGA, verze s FPGA stojí 16 tisíc korun. [23]

**HackRF** Toto SDR vyvinuté Michaelom 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 zařízení pro odpolech 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. [26]

**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. [3]

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

Tabulka 4.1: Přehled dospuňých SDR

[32] [2] [13] [15]

## **Část II**

### **Přehledový příjmač**



## Kapitola 5

### Hardware

V této kapitole shrnu veškerý potřebný hardware pro přehledový přijímač. Jedinou 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 fungovat 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.

#### 5.1 RTL-SDR

I přesto, že konkrétní druh SDR nebyl v zadání určen, byla jeho volba jen formální záležitostí. Jak už totiž vyplývá z kapitoly 4 RTL-SDR je de facto standard pro amatérské využití. Mezi ostatní SDR se vyjímá předeším cenou, která je desetinásobně 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 frekvencí 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 Radia 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. Nejzádanější variantou pro účely SDR je tuner E4000 od firmy Elionics, má totiž nejširší rozsah z používaných tunerů. [20] 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. [37] 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.

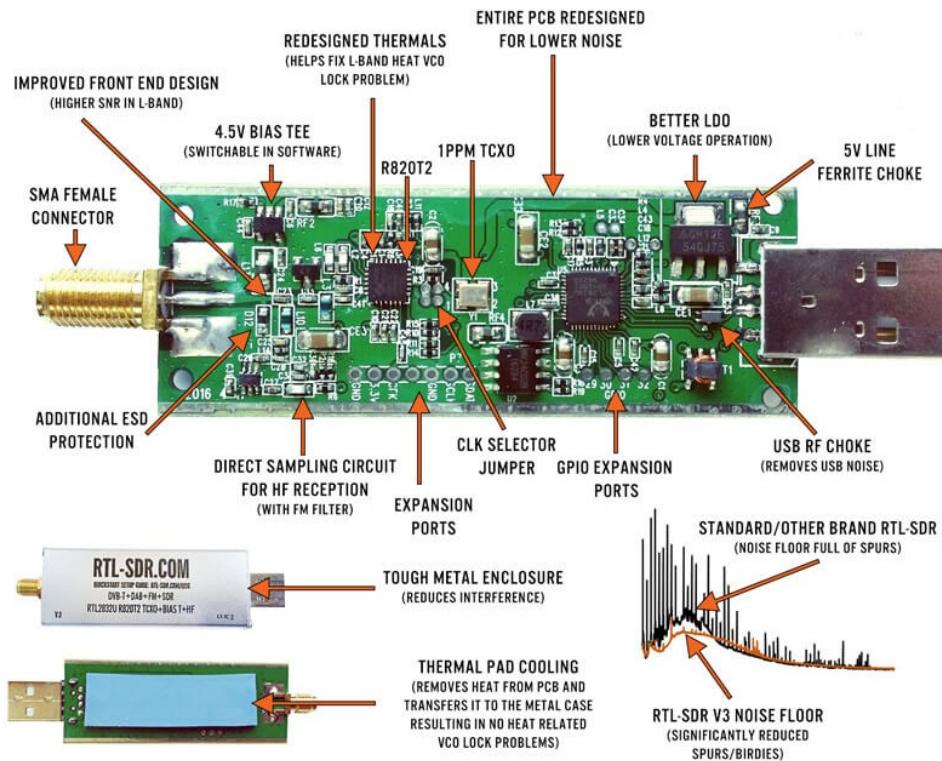
Posledním zásadním parametrem odlišujícím různá RTL-SDR je přítomnost teplotně kompenzovaného oscilátoru (TXCO). Ten je užitečný, protože při použití se RTL-SDR zahřívá a pokud není jeho oscilátor tepelně kompenzován, tak se jeho frekvence mění. Zahřívání na plnou provozní teplotu trvá několik minut, během této doby zůstane nalazená frekvence u SDR-RTL s TXCO stabilní, kdežto u ustátních 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 frekvence 500 kHz - 24 MHz přímo přes SMA konektor) a mnoho dalších malíčkostí, jejich kompletní výčet je v obrázku 5.1

## 5.2 Raspberry Pi

Raspberry Pi je malý jednodeskový počítač, který byl původně určen jako pomůcka při výuce programovacích jazyků scratch a python na základních a středních školách. Díky jeho ceně a faktu, že i přes to, že nedosahuje výkonu dnešních stolních nebo přenosných počítačů, je to plně hodnotný linuxový počítač, byl hned o první generaci mnohem větší zájem než jeho výrobce Raspberry Pi Foundation očekával. Začali ho totiž používat kutilové z celého světa, pro své projekty na které už jednoduché mikrokontrolery jako Arduino nestačily. [25]

Od roku 2012 kdy byl představen první model [19] již byly představeny tři generace a třikrát tolik verzí tohoto počítače. Novější modely přidávaly rychlosť, paměť a konektivitu a přitom si dokázaly udržet stálou cenu. Parametry nejnovější verze Raspberry Pi 3B+, které používám v tomto projektu jsou v



Obrázek 5.1: Výhody RTL-SDR blog V3 oproti běžným RTL DVB-T [37]

tabulce 5.1.

### 5.2.1 Displej

Pro zobrazení uživatelského rozhraní a zároveň pro jeho ovládání jsem použil rezistivní dotykový TFT displej s úhlopříčkou 5 palců a rozlišením 800x480 px. Připojen je pomocí HDMI a dotyková vrstva přes USB. Displej nebylo v systému potřeba nijak instalovat a včetně dotykové vrstvy fungoval tzv. Plug and Play. Jedinou výjimku jsem zaznamenal při testu distribuce ArchLinux, kde nefungoval vůbec. Displej má mít podsvícení s ULP (Ultra Low Power) technologií.

## 5.3 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 nej ostatních komponent), způsob nabíjení baterií, vypínač a propojení těchto komponent.

Jako zdroj energie jsem zvolil články typu 18650 Li-Ion. Je to stejný typ

<b>SOC</b>	Broadcom BCM2837B0
<b>CPU</b>	1.4GHz 64-bit quad-core ARM Cortex-A53 CPU
<b>RAM</b>	1GB LPDDR2 SDRAM
<b>Wireless</b>	Dual-band 802.11ac a Bluetooth 4.2
<b>Ethernet</b>	Gigabit Ethernet over USB 2.0 (max 300 Mbps)
<b>Video</b>	VideoCore IV 3D. Full-size HDMI
<b>USB</b>	4 x 2.0
<b>GPIO</b>	40 pinů
<b>Napájení</b>	5 V

**Tabulka 5.1:** Specifikace Raspberry Pi 3B+ [19]

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 nominálním 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 napěťích dochází k poškození článků) a zároveň při 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 částky) 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 poslední výhodou oproti dalším vyhovujícím je jeho velikost, ostatní kvalitnější moduly byly navrhnutы 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 nabíjet i při vypnutém přijímači. Spotřeba samotného modulu je minimální, je na neustálé připojení k baterii stavěn. Samotné Raspberry Pi 3 má při plné zátěži spotřebu okolo 730 mA [10], RTL-SDR má spotřebu okolo 280 mA [37] a displej má spotřebu menší než 400 mA [1] dohromady tedy méně než 1410 mA což při 5 V voltech dává 7 W a přibližná doba výdrže na baterii tedy bude okolo 4 hodin.

## 5.4 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

### 5.4.1 Autodesk Fusion 360

Fusion 360 of Firmy autodesk je varianta jejich CAD programu zaměřená zejména na kolaboraci a na návrh modelů určených pro výrobu, ať už pomocí 3D tiskáren nebo obrábění, soustružení, řezání vodním paprskem, laserem či plazmou [6]. 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 inventor také podporuje parametrické i přímé modelování [44].

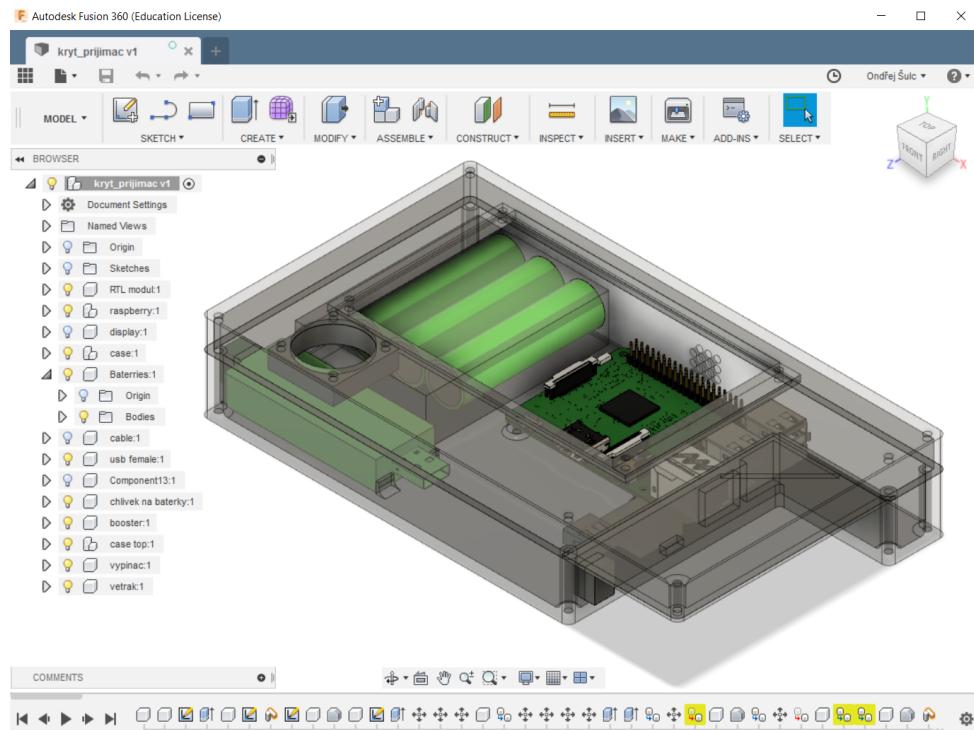
Velkou výhodou, kterou jsem často využil při modelování komponent přijímače je také schopnost otevřít téměř libovolný formát 3D modelu, já jsem tak mohl importovat již hotové modely Raspberry Pi, Li-Ion článků nebo různých konektorů. Poslední velmi užitečnou funkcí je již v úvodu zmíněná podpora kolaborace, díky ní jsem mohl svůj návrh průběžně konzultovat s již zmíněným kolegou a upravovat model dle jeho návrhů, tak aby šel výsledný model bez problému vytisknout.

### 5.4.2 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 držák s Li-Ion články stačilo vymodelovat jako přibližný kvádr, do krytu jsem ho totiž 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 velmi 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

## 5. Hardware



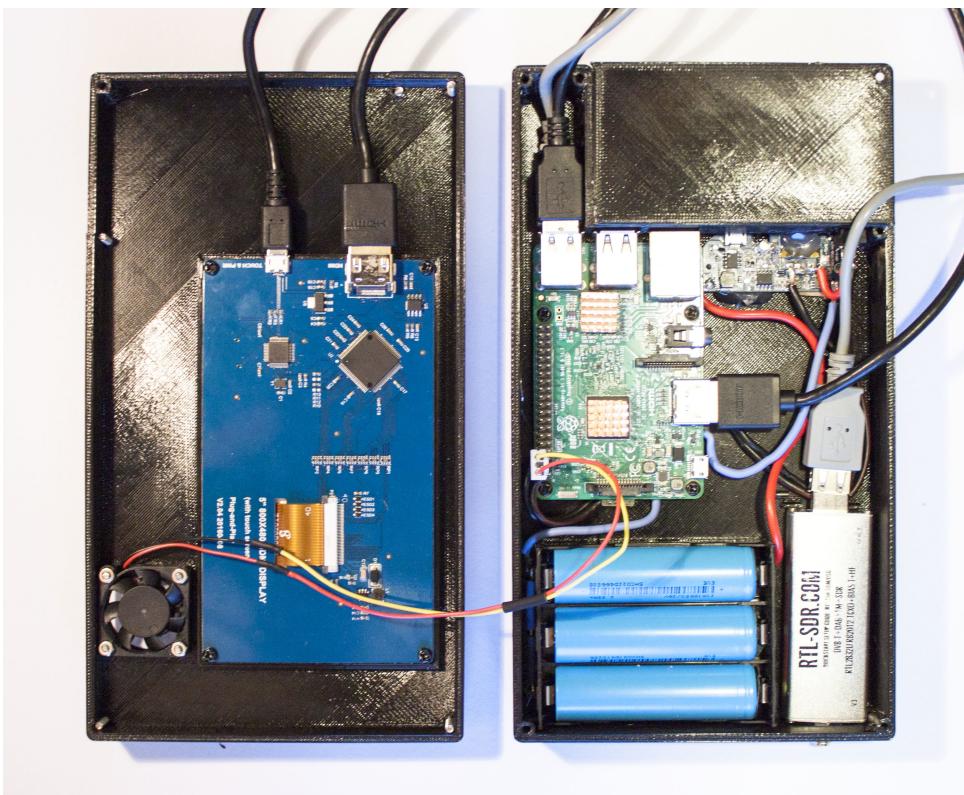
**Obrázek 5.2:** Prostředí Autodesk Fusion 360 a navrhnutý model krytu

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.

### 5.4.3 Tisk

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



**Obrázek 5.3:** Vnitřní uspořádání komponent

## 5.5 Zapojení

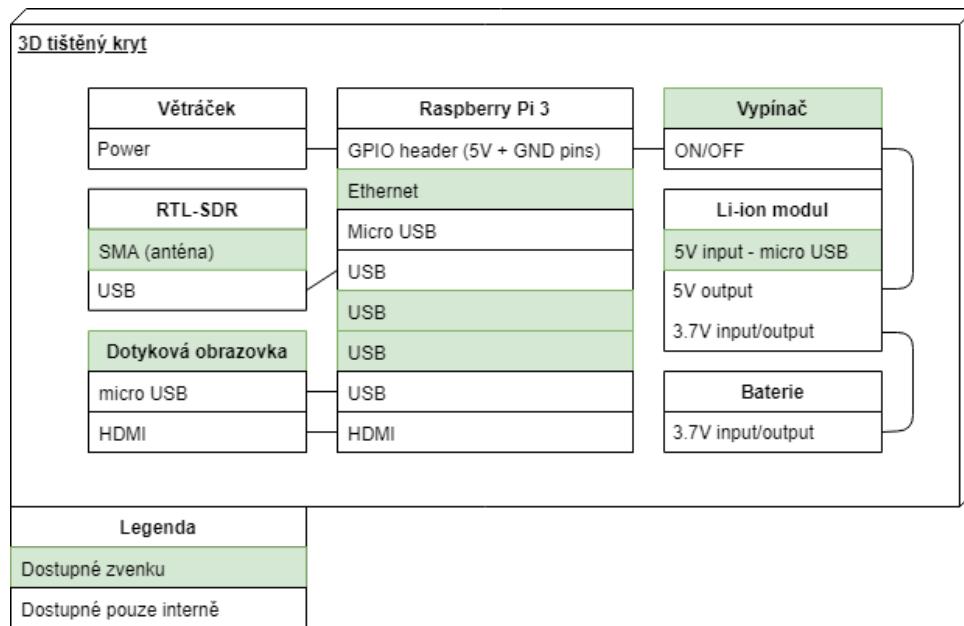
Pro zapojení se centrálním prvkem stejně jako pro výpočty stalo Raspberry Pi a jsou k němu připojeny všechny ostatní komponenty. Konkrétní porty a dostupnost komponent a volných portů zvenku lze vyčíst z diagramu na obrázku 5.4.

## 5.6 Testovací zařízení LoRa a Sigfox

Pro testování funkčnosti přehledového příjimače jsem nakonec používal vlastní testovací zařízení a nemusel se tak spoléhat jen na běžný provoz ve sledovaných sítích.

Pro testování technologie LoRa jsem zakoupil prototypovací desku LoRa32u4 II. Jedná se o mikrokontrolér založený na čipu ATmega32u4 s modulem HPD13 který obsahuje RF čip SX1276 s podporou LoRa na evropských pásmech okolo 868 MHz.

Velkou výhodou tohoto mikrokontroléru je kompatibilita s vývojovým prostředím Arduino a tak jsem neměl žádné potíže s tvorbou testovacích skriptů. Našel jsem dokonce knihovnu, která přidává podporu LoRaWAN a mohl jsem tak otestovat i připojení k síti The Things Network, která má v Praze několik



**Obrázek 5.4:** Diagram zapojení a dostupnosti portů přehledového přijímače

bran.

Pro testování sítě Sigfox mi můj vedoucí zařídil demo tlačítko společnosti T-Mobile která v Čechách síť sigfox provozuje. Jedná se o komerční produkt a ne vývojovou desku, nemohl jsem tedy nijak ovlivnit nastavení, ale tlačítko spolehlivě po každém zmáčknutí odvysílalo tři rámce. To jsem ověřil na spektrogramu i přes neuspěch v demodulaci technologie Sigfox.

# Kapitola 6

## 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 shrnu požadavky, pak představím použitý software z nich vyplývající a nakonec nastíním architekturu celého systému, to jak jsou jednotlivé programy propojeny a jaká mají rozhraní.

### 6.1 Požadavky a z nich vyplývající software

Všechny požadavky vycházejí ze zadání diplomové práce. Hlavním požadavkem je přijmout a dekódovat zprávy vysílané pomocí technologií IoT a následně je zobrazit a to jak lokálně přímo na přehledovém přijímači, tak vzdáleně přes IP protokoly. Tento hlavní požadavek lze rozložit do jednotlivých částí.

První část se týká příjmu zpráv z IoT zařízení z hlediska HW to obstarává RTL-SDR a použitý softwér ho tak musí umět ovládat a připravit přijaté I/Q vzorky pro dekódování, o to se postará knihovna librtlsdr a její utility které jsou součástí projektu rtl-sdr. Druhý požadavek tedy demodulace a dekódování zpráv bude splněn pomocí toolkitu GNU Radio ve kterém bude přijatý signál demudolován a zprávy dekódovány. Požadavek na zobrazení zpráv lokálně a vzdáleně jsem se rozhodl uspokojit zároveň a to pomocí webového uživatelského rozhraní založeného na microframeworku Flask a jeho rozšíření Flas-socketIO zajišťující zpětnou vazbu uživatelského rozhraní a celého systému v reálném čase.

#### 6.1.1 librtlsdr a její utility

Tato knihovna byla vyvinuta v rámci sofwarového projektu rtl-sdr od organizace Osmocom (Open source mobile communications). Knihovna je jeho ústřední částí sloužící k ovládání a přístupu k RTL-SDR a v podstatě všechny uživatelské aplikace využívající RTL-SDR jí potřebují ke svému fungování. Tato knihovna závisí na další knihovně libusb, to je dáno použitím rozhraní USB pro připojení RTL-SDR k počítači [20]. Kromě librtlsdr projekt obsahuje také několik utilit pro použití v terminálu:

**rtl\_fm** Přijímač FM vysílání využívající RTL-SDR. Dokáže RTL-SDR naladit na požadovanou frekvenci a jeho výstupem je zvuková stopa využitelná v terminálových přehrávačích.

**rtl\_sdr** Tato utilita již produkuje přímo I/Q vzorky, dokáže je zapsat do souboru nebo na standartní výstup.

**rtl\_test** Jak již název vypovídá slouží k testu funkčnosti knihovny librtlsdr.

**rtl\_tcp** Tato utilita funguje stejně jako rtl\_sdr, proud I/Q vzorků však neposílá na standartní výstup nebo do souboru ale tvoří z něj TCP/IP stream. Toho využívám v práci pro distribuci vzorků do instancí GNU Radia.

Poslední utilitou, která patří do této sekce i když není přímo součástí rtl-sdr je rtl\_mus. Tento skript vytvořil Simonyi Károly [14]. Jeho účel je vyřešit slabinu utility rtl\_tcp. Ta totiž dokáže posílat vzorky v jednu chvíli pouze jednomu klientu. Rtl\_mus tak zastoupí místo tohoto klienta a dále distribuuje prvky libovolnému množství koncových klientů.

### ■ 6.1.2 GNU Radio

GNU Radio je open source projekt a jedná se o nástroj pro digitální zpracování signálů. Založil ho Eric Blossom a původně běžel jen na platformě Linux. Dnes ho s omezenou funkcionalitou lze provozovat i na počítačích s Windows. Jeho hlavní výhodou je univerzálnost. Poskytuje velkou databázi bloků určených pro zpracovávání signálů jejichž pomocí může být například implementováno softwérové rádio. GNU radio může být použito spolu s HW a tvořit tak SDR nebo i bez něj v použití jako simulační prostředí. Díky tomu je jeho použití rozšířeno jak mezi radioamatéry tak v univerzitní a komerční sféře, kde pomáhá při vývoji a výzkumu.

Většina procesních bloků GNU Radia je implementována v C++, ale jejich vzájemné propojení tedy vytváření flowgrafů probíhá v pythonu. Aby mohl python s procesními bloky pracovat využívá SWIG (Simplified Wrapper and Interface Generator) který bloky v C++ obalí a vytvoří pro ně rozhraní v jazyce python. [30]

Zmíněné flowgrafy jsou podstatou GNU Radia, tvoří řetězec procesních bloků, které zpracovávají přijatá data a pak je posírají pro další zpracování následujícím blokům. Jejich strukturu většinou tvoří počáteční zdrojový blok, který buďto generuje signál nebo ho získává z jiné aplikace nebo například z RTL-SDR, následují bloky pro zpracování signálu jako (de)modulátory, FFT, decimátory a podobně a poslední je koncový blok, který zase funguje jako rozhraní s okolním světem a posílá výsledek zpracování dalším aplikacím, na standardní výstup nebo do nějakého SDR s možností vysílání.

Pro použití s různými SDR má GNU Radio širokou podporu. Nativně podporuje USRP a lze přidat podporu pro mnoho dalších SDR včetně RTL-SDR pomocí knihovny gr-osmosdr tu stejně jako knihovnu librtlsdr vyuvinula organizace Osmocom.

### ■ **GNU Radio companion**

GRC (GNU Radio companion) je rozšíření, které umožňuje v grafickém prostředí skládat jednotlivé bloky a vytvořit tak flowgraf. Oproti vytváření flowgrafa přímo v pythonu má výhodu v přehlednosti, ale hlavně v tom, že kontroluje logiku flowgrafa a upozorní tak na chyby typu spojení bloku s komplexním výstupem s blokem se vstupem pro bajty. Po návrhu flowgrafa v GRC je vygenerován python skript, který tento flowgraf definuje. Není žádný problém tento skript nadále upravovat a přiszpůsobit ho tak použití v rámci větší aplikace jako jsem to udělal já v případě flowgrafů pro zpracování zpráv LoRa a Sigfox.

### ■ **LoRa**

Flowgraf pro technologii LoRa je na první pohled velmi jednoduchý, skládá se jen ze tří bloků: Osmocom zdroje (Osmocom source), LoRa receiveru a koncového bloku pro odesílání zpráv (Message Socket Sink). Tato je jednoduchost je jen zdánlivá, protože LoRa receiver je ve skutečnosti jen obal pro bloky nižší úrovni zastávající skutečné zpracování signálu.

Osmocom zdroj není standartní výbavou GNU Radia. Přidán je pomocí knihovny gr-osmosdr a ve flowgrafe zastává funkci příjmu I/Q vzorků. V rámci jeho nastavení je možné definovat zdroj signálu, ať je to přímo nějaké z podporovaných SDR nebo TCP/IP stream. Dále je možné konfigurovat rádiovou frekvenci, vzorkovací frekvenci, zisk SDR a podobně.

Druhé dva bloky byly do gnu rádia přidány díky knihovně gr-lora[35]. Tato knihovna byla napsána Pieterem Robynsnem jedním z autorů [34] a její zprovoznění na Raspberry Pi se ukázalo oříškem, protože vyžaduje 64-bitovou architekturu, ale o tom se více zmíním v sekci 7.1.3. Knihovnu jsem zároveň musel upravit, ve verzi od Robynsna totiž spolu s dekódovanou zprávou neposílá parametry přijatého rámce jako je SNR, frekvence, faktor rozprostření a podobně. Tato úprava se zprvu také zdála jednoduchou, ale ve výsledku zabrala hodně času, viz sekce 7.1.4

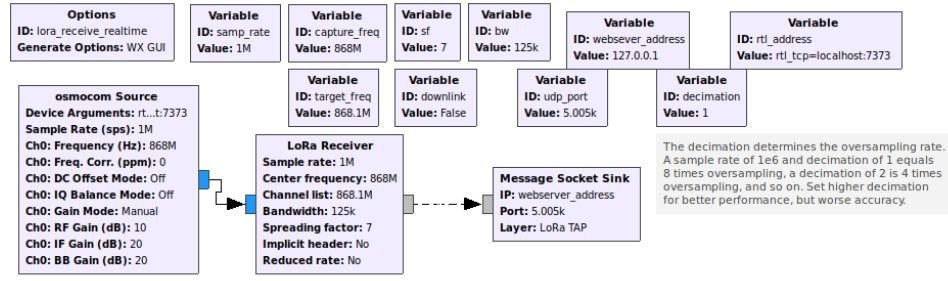
Blok receiver je složen ze třech bloků channelizer, controller a debugger. Tyto bloky implementují postup popsaný v kapitole 3.1.5. V rámci jeho konfigurace je nutné nastavit požadovaný kanál, faktor rozprostření a rozlišovací práh. Jeho výstupem je již dekódovaný rámec, který lze pomocí posledního bloku Message Socket Sink odeslat do dalších aplikací.

Knihovna gr-lora je závislá na dalších knihovnách bez nichž nemůže fungovat, následuje jejich výčet: python2-numpy, python2-scipy, swig, cppunit, fftw, gnuradio, libvolk, log4cpp, cmake, wx, and liquid-dsp. [35]

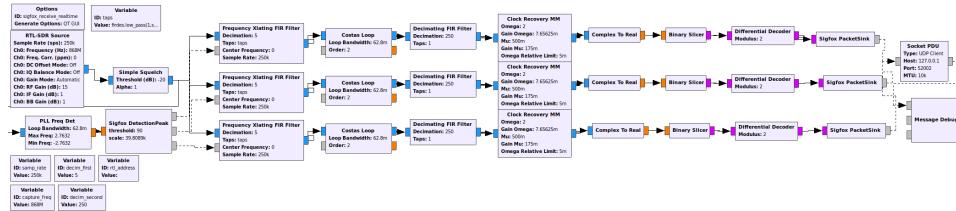
### ■ **Sigfox**

Tento flowgraf vychází z flowgrafa, který byl součástí upravené verze programu Scapy nazvané scapy-radio, která do něj přidávala podporu pro protokoly Bluetooth, Zigbee, Zwave a konečně Sigfox [47]. Obsahuje dva bloky, které nejsou standartní součástí GNU radia, jinak je složen z již připravených bloků.

## 6. Software



Obrázek 6.1: Flowgraf pro příjem LoRa



Obrázek 6.2: Flowgraf pro příjem Sigfox

První nestandardní blok se jmenuje Sigfox DetectionPeak a je implementován v pythonu. Jeho účel je jednotlivé detekované frekvence nosných třech opakování vysílání Sigfox předat do třech samostatných větví flowgrafa, každá demodulující jen jedno vysílání. Blok má dva parametry, paramater threshold udává o kolik se musí lišit následující detekovaná frekvence aby jí bylo možné považovat za následující opakování a parametr scale pomáhá normalizovat frekvenci.

Blok Sigfox PacketSink dekóduje demodulovaný signál. Je implementován v C++ a má dvě fáze. V první fázi detekuje preambuli a jakmile narazí na F.SYNC následovaný F.TYPE přečte ho a zjistí z něho délku zprávy, druhé části a dekóduje následujících několik bajtů, dle délky indikované v preambuli. Tato data přidá k preambuli a vytvoří paket.

I přestože autor flowgrafa, ze kterého tento vychází nepíše nic o tom, že by nebyl plně funkční, nepodařilo se mi ho zprovoznit. Bohužel, jak vyplývá z komentářů dalších uživatelů v sekci issues na stránkách autora, nejsem sám. Kvůli tomuto problému je tak příjem Sigfox nekompletní a vyžaduje další práci.

### 6.1.3 Flask a Flask-socketIO

Flask je framework pro tvorbu webových aplikací. Poskytuje nástroje, knihovny a technologie pro tvorbu stránek, blogů nebo i pro webové rozhraní jednoduších aplikací jako je přehledový přijímač. Sám o sobě nemá mnoho funkcí a proto závisí jen na knihovně Werkzeug která se stará o WSGI protokol pro komunikaci pythonu a webserveru a na knihovně jinja2, která funguje

jako šablonovací systém. Další funkce se do Flasku přidávají pomocí pluginů. Aplikace, které tento framework používají jsou například Pinterest nebo LinkedIn. [48]

Jedním z rozšíření Flasku je Flask-socketIO. Toto rozšíření přidává podporu pro Socket.IO a díky tomu Flask může podporovat obousměrnou komunikaci s klientem s nízkou prodlevou. Toho je využito pro webové rozhraní přehledového přijímače.

## 6.2 Architektura

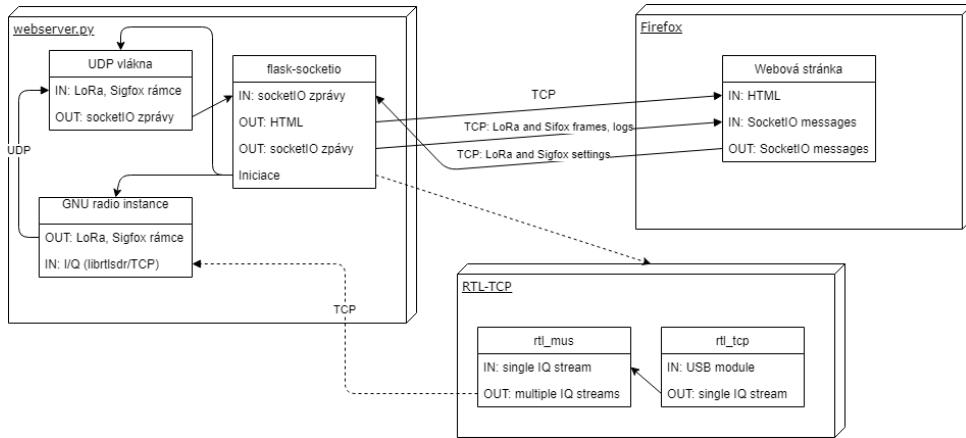
Softwarovou výbavu přehledového přijímače lze rozdělit do několika bloků. Jejich vzájemná spolupráce závisí na prostředí na kterém software běží. Rozlišuji tři prostředí, která se odlyšují zejména přístupem k RTL-SDR.

**SCANNER** Toto prostředí je používané na přehledovém přijímači. Kvůli malému výkonu Raspberry Pi na kterém je přijímač založen, má toto prostředí několik omezení. Prvním je možnost přijímat jen zprávy z jednoho kanálu a jen s jedním činitelem rozprostření v jednu chvíli. Ani to by samo o sobě nestačilo pro plynulý běh a tak jsou ještě vzorky RTL-SDR při demodulaci nadvzorkovány jen dvakrát. Dále je kvůli menším režijním nákladům k I/Q vzorkům přistupováno napřímo z GNU Radia a ne přes TCP a rtl-mus, z čehož vyplývá nemožnost sdílet I/Q vzorky s dalšími aplikacemi nebo po síti.

**PC** Prostředí používané při spuštění software přehledového přijímače na PC. Díky většímu výkonu a lepší optimalizaci knihoven pro x86 architekturu oproti ARM je na PC dostatečný výkon pro demodulaci výše kanálů a činitelů rozprostření současně. Aby mělo více instancí GNU Radia přístup k I/Q vzorkům současně je nutné zpřístupnění RTL-SDR přes TCP a použití rtl\_mus.

**REMOTE** Jedinný rozdíl tohoto prostředí oproti PC je, že místo spuštění rtl\_tcp a rtl\_mus, vyžaduje toto prostředí adresu již spuštěného rtl\_mus serveru. Toto prostředí je vhodné pro výkonější stroj využívající RTL-SDR připojené k přehledovému přijímači. Kombinuje tak výhody výkonu PC a flexibilnějších možností umístění přijímače.

Tato prostředí jsou definována v konfiguračním souboru config.ini, který načítá ústřední python skript webserver.py. Tento skript spouští, propojuje a ovládá ostatní bloky. Druhý blok je přítomen jen v prostředí PC a věnuje se modulu RTL-SDR, zabývá se jeho ovládáním a distribucí digitalizovaných vzorků rádiového spektra. Další blok spouští skripty využívající GNU Radio, ty zpracovávají vzorky z bloku RTL-SDR a po dekódování rámce je předávají kontrolnímu bloku. Posledním blokem je samotný klient, který ve webovém prohlížeči zobrazuje přijaté rámce a umožňuje změnu nastavení. Tyto bloky fungují samostatně a tak lze každý provozovat na jiném stroji, musí však být zajištěna možnost jejich vzájemné komunikace a kompatibilní operační systém.



**Obrázek 6.3:** Diagram vzájemné komunikace jednotlivých bloků

## 6.3 Přehled používaných souborů

V této části popíšu účel a fungování jednotlivých souborů používaných v přehledovém přijímači.

### config.ini

Konfigurační soubor jehož sekce tvoří jednotlivá prostředí. Obsah sekcí určuje vlastnosti jednotlivých prostředí. Tento soubor je načítán při spuštění webserver.py a načtené prostředí je určeno argumenty použitými při jeho spuštění.

### lora\_realtime\_receive.py a sigfox\_realtime\_receive.py

Tyto soubory obsahují třídy, které jsou nainstalovány do webserver.py a fungují jako definice flowgrafu GNU Radia a řídí tak jeho činnost. Jejich obsah v podstatě odpovídá obrázkům 6.1 a 6.2. Co v grafickém prostředí není vidět je upravená inicializace těchto tříd, tak aby bylo možné z webserver.py ovládat parametry jako je frekvence, činitel rozprostření, port pro příjem I/Q vzorků a odesílání dekódovaných rámci a používat tak jednu třídu pro všechny možné konfigurace. Pro přidání podpory dalších IoT technologií bylo nutné přidat další podobnou třídu. Po demodulaci a dekódování rámci IoT technologie posílají tyto přes UDP do vláken k tomu určených v webserver.py.

### static/index.html

Tento soubor definuje možnosti klienta a má dvě hlavní části. První část určuje podobu webového prostředí pro interakci s přehledovým přijímačem a je napsána v jazyce HTML s vnořenými CSS styly. Vytváří tabulkou pro přehledné zobrazení přijatých rámci a také formuláře pro nastavení přijímače.

Druhou o něco zajímavější část tvoří skript v jazyce javaScript. Nejedná se o čistý javaScript, ale je použita knihovna jQuery a Socket.IO. Tato část zajišťuje funkčnost rozhraní. Stará se jednak o komunikaci přes socket-io která probíhá pomocí websocketů a formátu JSON a druhak o vyčítání nastavení z formulářů a o přidávání přijatých rámců ve formě nových řádků do tabulky.

#### **webserver.py**

Centrální část celého přijímače. Tento skript je spouštěn s jedním argumentem, ve kterém mu říkáme v jakém prostředí běží. Ná základě tohoto argumentu načítá ze souboru config.ini správnou konfiguraci a na základě té inicializuje následující prvky: rtl\_tcp a rtl\_mus (jen PC), vlákna pro příjem UDP paketů s rámci LoRa a Sigfox a Flask-SocketIO. Samotné instance GNU Radia vytváří až ve chvíli kdy jsou na základě požadavku z webového rozhraní potřeba a po vypnutí dané funkcionality v prostředí je zase pozastavuje a maže.

Během aktivního příjmu nějaké IoT technologie webserver.py přijímá ve vláknu v pozadí UDP pakety s daty z instancí GNU Radia, ty pak parsuje a vytváří tak slovník s jednotlivými parametry a samotným obsahem rámce. Tento slovník pak pomocí Flask-SocketIO převede na JSON a přes websocket pošle ke klientům. Zároveň je Flask-SocketIO kdykoliv připraven příjmout od klienta JSON s aktualizací nastavení a zpracovat ho.

Více detailů o fungování jednotlivých skriptů, jejich funkcích, struktuře, proměnných a celkovém fungování je k dispozici ve formě komentářů přímo ve zdrojových souborech.



# Kapitola 7

## Průběh práce a výsledky

### 7.1 Postup práce

#### 7.1.1 Počáteční testování

Prvním krokem práce bylo promyslet výběr SDR. Důvody pro výběr RTL-SDR jsem popsal podrobně v sekci 5.1, ale ve zkratce šlo o výrazně nejlepší poměr ceny a vlastností. Poté co SDR přišlo jsem nejdříve testoval jeho schopnosti na různých projektech od příjmu pozic letadel v okolí přes ADB-S a poslechu FM rádiových stanic po sledování spektra v programu SDR#. V předešlém průzkumu provedeném v rámci projektu jsem vyvodil, že budu detektovat sítě LoRa a Sigfox. Důvod byl ten, že jako jediné dva standardy IoT mají sítě s většinovým pokrytím ČR. Nejdříve jsem myslel, že budu schopen přijímač sestrojit a otestovat na základě již existujícího provozu v těchto sítích, ale po neuspěšných pokusech zachytit víc jak jedno vysílání denně pomocí knihovny pro příjem LoRa jsem si uvědomil, že se neobejdou bez testovacího zařízení.

Zakoupil jsem proto modul LoRa32u4 II od společnosti BSFrance a můj vedoucí práce Ing. Pavel Troller mi zařídil reklamní modul s tlačítkem využívající technologii Sigfox. Pomocí těchto modulů jsem se pustil do dalšího testování

Nejprve jsem začal s technologií LoRa. Vzhledem k nepraktičnosti jsem testování neprováděl přímo na Raspberry Pi, které jsem zvolil jako výpočetní jednotku pro přijímač, ale na notebooku, na který jsem nainstaloval operační systém Linux, konkrétně distribuci Ubuntu. Díky tomu, že oficiální systém pro Raspberry Pi Raspbian je také distribucí Linuxu jsem očekával, že ve chvíli kdy budu mít příjem LoRa funkční a odladěný bude stačit vše přenést na Raspberry Pi.

Po instalaci potřebného SW a knihoven jako je GNU Radio nebo librtlsdr jsem testoval dvě různé knihovny pro příjem LoRa pomocí GNU Radia: gr-lora od Matta Knighta [17] a gr-lora od Pietera Robysona [35]. První nejen neměla podporu explicitní hlavičky LoRa (moje testovací zařízení má vždy explicitní hlavičku a tak by ho nešlo k testování požít), ale měla kvůli tomuto nedostatku i větší množství nutně nastavitelných parametrů, protože nemohla vypočítat CR z údajů v hlavičce a tak jsem i přes větší množství závislostí

pokračoval vývoj s [35].

### ■ 7.1.2 Vývoj kontrolní aplikace a webového rozhraní

Náročnost vývoje kontrolní aplikace přibližně odpovídala mým odhadům. Bylo potřeba vymyslet mechanismus spouštění flowgrafů GNU Radia, příjmu od nich dekódovaných zpráv a také rozhraní, které bude dostupné i vzdáleně a půjdou v něm zobrazit přijatá data a měnit nastavení.

Jako výchozí bod jsem zvolil framework Flask protože je stejně jako flowgragy GNU rádia napsán pro python a také jsem s ním již měl zkušenost. Bohužel to byl taky jediný prvek na kterém jsem se na delší dobu zasekl. Nefungovalo mi odesílání zpráv do rozhraní z vlákna přijímajícího UDP pakety z flowgrafů. Nakonec se ukázalo, že za to mohly restarty aplikace Flasku způsobené debugovacím módem, zároveň s Flaskem se totiž nerestartovalo vlákno pro příjem zpráv z flowgrafů a tak se stále odkazovalo na původní instanci Flasku, která již pochopitelně neodpovídala.

Poměrně náročné také bylo zprovoznění interaktivnosti rozhraní na straně klienta. Vzhledem k tomu, že je to webová aplikace, tak její vývoj probíhal v JavaScriptu se kterým jsem neměl žádné předchozí zkušenosti a tak jsem hodně čerpal z ukázek na internetu a snažil se tuto mezeru donutit.

### ■ 7.1.3 Problémy s Raspberry Pi

Problém nastal ve chvíli kdy jsem dosavadní práci chtěl vyzkoušet na Raspberry Pi. Velké množství potřebných knihoven, které jsem v Ubuntu naistaloval pomocí balíčků v repozitáři nebylo v této podobě na Raspberry Pi dostupných a tak jsem je musel komplikovat ze zdroje. To sebou přineslo problémy nekompatibility různých verzí a já tak strávil spoustu času jen přípravou prostředí. Potom co se mi vše podařilo zprovoznit a naistaloval jsem gr-lora jsem zjistil, že nefunguje. Nevěděl jsem proč, protože se neojevila žádná chybová hláška, ale ani testovací flowgraf demodulující nahrané vzorky ze souboru nefungoval. Po věčnosti strávené pročítáním issues na githubu knihovny a spoustě pokusech o zprovoznění jsem se v jedné z issues dozvěděl, že knihovna nejspíše funguje jen na 64-bitových systémech.

Myslel, jsem si že to by nemusel být tak velký problém když Raspberry Pi 3, které jsem používal má 64-bitový procesor. Bohužel však neexistuje oficiální systém který by byl 64-bitový a neoficiálních 64-bitových distribucí je jako šafránu. Po otestování několika z nich jsem se ustálil na distribuci Fedora. Po té co jsem vše nainstaloval a byl opět připraven testovat příjem LoRa na novém systému mi přestalo fungovat samotné Raspberry Pi a při svém odchodu s sebou vzalo i kartu s funkčním systémem. Po koupi nového kusu jsem tedy celé maritorium instalace prošel znovu. Abych zjistil, že příjem LoRa sice funguje, ale je velmi nestabilní a zachytí přibližně každou pátou testovacím zařízením vyslanou zprávu.

Důvodem pro tuto nestabilitu byl malý výkon. Raspberry Pi nestačilo zpracovávat I/Q vzorky z RTL-SDR a tak jich značné množství zahazovalo. Kvůli

tomuto problém jsem musel narušit původní koncepci distribuce vzorků pomocí TCP, které jsem zvolil kvůli modulárnosti a přejít na přímý přístup k RTL-SDR kvůli menším režijním nákladům. Ani to samo o sobě nestačilo a musel jsem tak decimovat přijaté vzorky (RTL-SDR nepodporuje vzorkovací frekvence v rozsahu od 0,3-0,9 Msps) a najít správnou rovnováhu mezi přesností demodulace zprávy a výkonem Raspberry Pi.

### **7.1.4 Úprava knihovny gr-lora**

Abych splnil požadavky v zadání a mohl zobrazit kromě přijatého obsahu paketů i SNR, frekvenci, šířku pásma musel jsem knihovnu gr-lora upravit. Úprava se na první pohled jevila jednoduchá i přesto, že je napsána stejně jako ostatní knihovny pro GNU Radio v C++ se kterým nemám zkušenosti. Jelikož knihovna již měla připravenou strukturu pro přidání těchto parametrů ve formě hlavičky k dekódoavaným datům, myslíl jsem že to zvládnu poměrně snadno. Bohužel však v metodě používané pro výpis dekódovaných dat byl bug, který způsoboval korupci parametrů odesílaných do hlavičky. Tuto metodu není nutné používat pokud je použit blok pro odeslání zpráv a tak byla oprava pro účely této práce jednoduchá, ale přijít na to co chybu způsobuje zabralo mnoho hodin debugování. O nalezené chybě budu informovat autora, protože si myslím, že tento bug byl důvod proč přidání parametrů do hlavičky nepovedl on sám.

### **7.1.5 Podpůrný hardware**

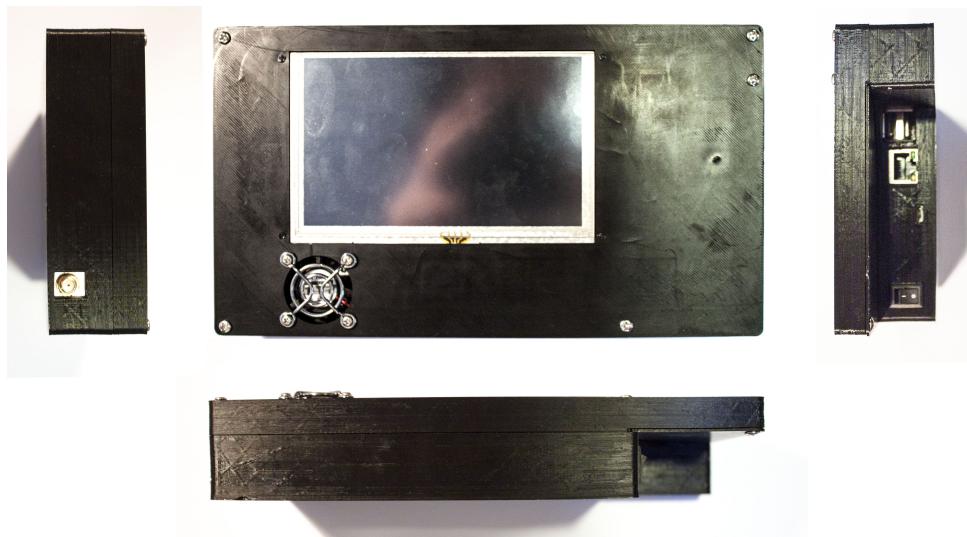
Kvůli tomu že výsledné zařízení mělo být funkčním prototypem jsem musel dodat další podpůrný hardware. Kromě nezbytného RTL-SDR, Raspberry Pi a displeje jsem tak přidal na míru připravené napájení z baterií, chlazení (ventilátor), vypínač a hlavně také kryt vytisknutý na 3D tiskárně.

Co se týče napájení tak nejtěžším bodem bylo najít vhodný DC/DC měnič napětí z 3,7 V baterie na 5 V Raspberry Pi. Většina dostupných modulů nemá dostatečný výkon. Nakonec jsem po důkladném testování použil modul z kvalitní powerbanky. V chlazení jsem se inspiroval a použil stejný větráček jako používá většina aktivně chlazených krabiček na Raspberry Pi.

Nejnáročnější fází však byl návrh krytu. Návrhem správného uspořádání komponent tak, aby byly přístupné volné USB a LAN porty a zároveň z prototypu nevedly ven kabely zapojené do těch vnitřně obsazených, vše zabíralo co nejméně místa a zároveň celý prototyp netrpěl přehříváním se ukázal být komplikovaným a vyžadoval alespoň přibližné vymodelování všech komponent. Limitován jsem také byl tím co je možné vytisknout a proto návrh zabral mnoho hodin práce.

## **7.2 Výsledky**

V rámci této práce se mi podařilo zhотовit funkční a kompaktní prototyp přijímače IoT sítí, který je díky vestavěné baterii mobilní a může tak být



**Obrázek 7.1:** Přehledový přijímač internetu věcí

použit k monitorování i sítí i v odlehlých oblastech mimo dosah elektrické sítě. Přijímač pomocí SDR stabilně detekuje síť typu LoRa přičemž v jednu chvíli dokáže detektovat vysílající zařízení na jednom kanálu a používající jeden činitel rozprostření. Mezi kanály a činitely rozprostření lze plynule přepínat na dotykové obrazovce prototypu, nebo vzdáleně přes klasickou IP síť (Wi-fi nebo Ethernet) ve webovém rozhraní. V případě zachyceného vysílání dokáže přijímač i webové rozhraní zobrazit frekvenci, SNR, činitel rozprostření, šířku pásmu a obsah zprávy. Protože na přijímači běží systém linux, je také možné ovládat ho přes protokol SSH.

Ostatní standardy přijímač sám nedetekuje. Částečně byla implementována podpora technologie Sigfox, ale použitý GNU Radio flowgraf neposkytuje stabilní výsledky. Důvodem pro absenci podpory dalších IoT standardů byla časová náročnost, absence testovacích zařízení absence předešlých projektů, které by se danou technologií zabývaly. Ovládací program přijímače je ale napsán tak, že může být snadno dodána podpora dalších standardů v případě jejich implementace v GNU Radiu.

Další z v zadání předpokládaných, ale neimplementovaných funkcí je měření chybovosti příjmu, to by samozřejmě šlo implementovat při použití testovacího zařízení, které by posílalo předem známá data, avšak při monitorování jako takovém neznáme původní obsah zpráv a chybovost příjmu tak nelze měřit. Pro přehledový přijímač tak podobná funkce není úplně stavěná a tak nebyla vysoko na seznamu priorit a soustředil jsem se proto na jiné funkcionality. Kromě samotného přijímače lze vytvořený software provozovat na dalších linuxových počítačích a to buď s připojeným RTL-SDR nebo s jiným dostupným zdrojem přes TCP. Při podobném použití se předpokládá vyšší výkon stroje a tak je povolena možnost demodulovat a dekódovat více kanálů a činitelů rozprostření zároveň.

## Kapitola 8

### Závěr

Dle požadavků zadání této práce jsem zkonstruoval a naprogramoval funkční přehledový přijímač sítí internetu věcí, který splňuje většinu předpokládaných funkcí. Jeho hlavním nedostatkem je malý počet podporovaných IoT technologií. Jelikož je však přijímač navrhnut způsobem, kdy lze snadno přidat podporu dalších v GNU RADIU implementovaných IoT standardů nabízí se tato možnost pro budoucí práce. Dalším možným budoucím rozšířením jsou nové funkce přehledového přijímače. Například by se pro podobné zařízení hodila funkce zobrazení spektrogramu, kde by bylo možné pozorovat vysílání i nepodporovaných služeb a získat celkový přehled o využití spektra v dané oblasti.



## Bibliografie

- [1] 52-Pi. *5-Inch-800x480-HDMI-TFT-LCD-Touch-Screen*. URL: [https://wiki.52pi.com/index.php?title=5-Inch-800x480-HDMI-TFT-LCD-Touch-Screen%5C\\_SKU:Z-0053](https://wiki.52pi.com/index.php?title=5-Inch-800x480-HDMI-TFT-LCD-Touch-Screen%5C_SKU:Z-0053).
- [2] rtl-sdr.com admin. *REVIEW: AIRSPY VS. SDRPLAY RSP VS. HACKRKF*. 2016. URL: <https://www.rtl-sdr.com/review-airspy-vs-sdrplay-rsp-vs-hackrkf/>.
- [3] Airspy. *What is Airspy?* URL: <https://airspy.com/>.
- [4] LoRa Alliance. *LoRa Alliance technology*. URL: <https://www.lora-alliance.org/technology>.
- [5] LoRa Alliance. *LoRaWAN™ White Papers*. URL: <https://www.lora-alliance.org/lorawan-white-papers>.
- [6] Autodesk. *Revoluční cloudrová 3D CAD aplikace. Vytvářejte cokoliv*. 2018. URL: <https://www.fusion360.cz/>.
- [7] Josh Blum. *LoRa modem with LimeSDR*. Červ. 2016. URL: <https://myriadrf.org/blog/lora-modem-limesdr/>.
- [8] coherent-receiver. *RTL-SDR Introduction*. 2018. URL: <https://coherent-receiver.com/publications>.
- [9] Paul Disk91. *The Sigfox radio protocol*. Lis. 2017. URL: <https://www.disk91.com/2017/technology/sigfox/the-sigfox-radio-protocol/>.
- [10] Raspberry Pi Dramble. *Power Consumption Benchmarks*. URL: <https://www.pidramble.com/wiki/benchmarks/power-consumption>.
- [11] Bertalan Eged a Benjamin Babják. “Universal Software Defined Radio Development Platform”. In: *Dynamic Communications Management*. 2006.
- [12] Claire Goursaud a Jean-Marie Gorce. “Dedicated networks for IoT : PHY / MAC state of the art and challenges”. In: *EAI endorsed transactions on Internet of Things*, (lis. 2015).
- [13] ITEAD. *Airspy R2*. URL: <https://www.itead.cc/airspy.html>.
- [14] Simonyi Károly. *RTL Multi-User Server*. 2014. URL: [https://github.com/simonyiszk/rtl%5C\\_mus](https://github.com/simonyiszk/rtl%5C_mus).

- [15] Taylor Killian. *SDR Showdown: HackRF vs. bladeRF vs. USRP*. 2013. URL: <http://www.taylorkillian.com/2013/08/sdr-showdown-hackrf-vs-bladerf-vs-usrp.html>.
- [16] MATT KNIGHT. “Reversing LoRa: Exploring NextGeneration Wireless”. In: *GRCon*. 2016.
- [17] Matt Knight. *GNU Radio OOT module implementing the LoRa PHY*. URL: <https://github.com/BastilleResearch/gr-lora>.
- [18] Perry Lea. *Essential Technical guide of Sigfox protocol: Network architecture, interfaces, protocol stack*. URL: <https://www.survivingwithandroid.com/2018/07/sigfox-protocol-network-architecture-iot-protocol-stack.html>.
- [19] Conor Lyons. *A History Of The Raspberry Pi*. Břez. 2015. URL: <http://novadigitalmedia.com/history-raspberry-pi/>.
- [20] Steve Markgraf, Dimitri Stolnikov a Hoernchen. *rtl-sdr*. URL: <https://osmocom.org/projects/rtl-sdr/wiki/Rtl-sdr>.
- [21] Jay Miller, Justin Wesley a Matt Frazier. *Differential Binary Phase Shift Keying*. URL: <http://www.cryptofreak.org/projects/mod-chal/dbpsk.php>.
- [22] Kyung-wan Nam. “Direct Down-Conversion System with I/Q Correction”. In: *SLWU085–July 2013* (2013).
- [23] Nuand. *bladeRF 2.0 micro*. URL: <https://www.nuand.com/bladerf-2-0-micro/>.
- [24] Nutaq. *A short history of software-defined radio (SDR) technology*. 2010. URL: <https://www.nutaq.com/blog/short-history-software-defined-radio-sdr-technology>.
- [25] Opensource.com. *What is a Raspberry Pi?* URL: <https://opensource.com/resources/raspberry-pi>.
- [26] Michael Ossmann. *HackRF, an open source SDR platform*. 2013. URL: <https://www.kickstarter.com/projects/mossmann/hackrf-an-open-source-sdr-platform>.
- [27] IoT Portal. *LoRaWAN*. Ún. 2016. URL: <https://www.iot-portal.cz/2016/02/29 lorawan/>.
- [28] IoT Portal. *Mapa pokrytí*. URL: <https://www.iot-portal.cz/mapa-pokryti/>.
- [29] IoT Portal. *Sigfox*. Ún. 2016. URL: <https://www.iot-portal.cz/2016/02/26/sigfox/>.
- [30] Gnu Radio. *Main page*. URL: [https://wiki.gnuradio.org/index.php/Main%5C\\_Page](https://wiki.gnuradio.org/index.php/Main%5C_Page).
- [31] České Radiokomunikace. *Služby IoT*. URL: <https://cra.cz/sluzby-iot>.

- [32] Ettus Research. *USRP B205mini-i*. URL: <https://kb.ettus.com/B200/B210/B200mini/B205mini>.
- [33] András Retzler. "Software Defined Radio Receiver Application with Web-based Interface". Dipl. Budapest University of Technology a Economics, 2014.
- [34] Pieter Robyns et al. "A Multi-Channel Software Decoder for the LoRa Modulation Scheme". In: *Conference: 3rd International Conference on Internet of Things, Big Data and Security*. 2018.
- [35] Pieter Robynson. *gr-lora*. URL: <https://github.com/rpp0/gr-lora>.
- [36] roklobsta. *Welcome to the rtlsdr.org wiki!* 2016. URL: [https://rtlsdr.org/#history%5C\\_and%5C\\_discovery%5C\\_of%5C\\_rtlsdr](https://rtlsdr.org/#history%5C_and%5C_discovery%5C_of%5C_rtlsdr).
- [37] rtl-sdr.com. *BUY RTL-SDR DONGLES (RTL2832U)*. URL: <https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/>.
- [38] T.M. Schmidl a D.C. Cox. "Robust frequency and timing synchronization for OFDM". In: *IEEE Transactions on Communications* 45 (pros. 1997), s. 1613–1621.
- [39] Sigfox. *Make things come alive in a secure way*. Tech. zpr. [https://www.sigfox.com/sites/default/files/1/SIGFOX-White\\_Paper\\_Security.pdf](https://www.sigfox.com/sites/default/files/1/SIGFOX-White_Paper_Security.pdf). Sigfox, ún. 2017.
- [40] Sigfox. *SIGFOX has you covered*. URL: <https://www.sigfox.com/en/coverage>.
- [41] Sigfox. *Sigfox Technical Overview*. Tech. zpr. <https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf>. Sigfox, květ. 2017.
- [42] Simplecell. *Simplecell - connecting things*. URL: <https://simplecell.eu/>.
- [43] Statista. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)*. Ún. 2018. URL: <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>.
- [44] CAD Studio. *Autodesk Fusion 360 - cloudový 3D CAD/CAM/CAE*. URL: <https://www.cadstudio.cz/fusion360>.
- [45] Ondřej Šulc. "Přehled technologií pro IoT". Projekt. 2018.
- [46] tapr. *HPSDR Mercury tech specification*. URL: [https://www.tapr.org/kits%5C\\_merc.html](https://www.tapr.org/kits%5C_merc.html).
- [47] cyber tools. *scapy-radio*. 2016. URL: <https://bitbucket.org/cybertools/scapy-radio/src>.
- [48] Wikipedia. *Flask (web framework)*. URL: [https://en.wikipedia.org/wiki/Flask%5C\\_\(web%5C\\_framework\)](https://en.wikipedia.org/wiki/Flask%5C_(web%5C_framework)).
- [49] Juan Carlos Zuniga a Benoit Ponsard. "SIGFOX System Description". In: *IETF 96*. <https://www.ietf.org/proceedings/96/slides/slides-96-lpwan-10.pdf>. Čvc 2016.

*Bibliografie* .....

- [50] Juan Carlos Zuniga a Benoit Ponsard. *Sigfox System Description*. URL: <https://datatracker.ietf.org/meeting/97/materials/slides-97-lpwan-25-sigfox-system-description-00>.