VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY
DEPARTMENT OF RADIO ELECTRONICS

FRAMEWORK PRO GENEROVÁNÍ PROVOZU V IOT SÍTÍCH
A FRAMEWORK FOR GENERATING TRAFFIC IN IOT NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Ľubomír Švehla**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Ladislav Polák, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Elektronika a sdělovací technika**
Ústav radioelektroniky

Student: Bc. Ľubomír Švehla ***ID:*** 164422
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Framework pro generování provozu v IoT sítích

POKYNY PRO VYPRACOVÁNÍ:

V teoretické části práce prostudujte pokročilé bezdrátové protokoly BLE, Z-Wave a IQRF, které se využívají v IoT sítích. Dále proveďte rešerši dostupného zařízení (hardware) pro generování paketů a vytvoření komunikace v IoT sítích. Navrhněte vhodný systém a postup pro generování provozu (monitorování dát ze sensorů) s cílem mít možnost generovat různé typy útoků v IoT sítích, které využívají protokoly BLE, Z-Wave a IQRF.

V experimentální části práce navržený systém implementujte, zprovozněte a otestujte. Získané výsledky přehledně a detailně vyhodnoťte a diskutujte.

DOPORUČENÁ LITERATURA:

[1] KUCHTA, Radek, Radimir VRBA a Vladimir SULC. Smart Platform for Wireless Communication - Case Study. In: Seventh International Conference on Networking (ICN 2008), IEEE, 2008, s. 117-120. ISBN 978-0-7695-3-06-9.

[2] LIU, Xiruo, Meiyuan ZHAO, Sugang LI, Feixiong ZHANG a Wade TRAPPE. A Security Framework for the Internet of Things in the Future Internet Architecture. Future Internet. 2017, 9(3), s. 1-28. ISSN 1999-5903.

[3] GRAYVER, Eugene. Implementing software defined radio. 2012th ed. New York: Springer, 2012. ISBN 978-1441993311.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: doc. Ing. Ladislav Polák, Ph.D.
Konzultant:

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda oborové rady

ABSTRAKT

Diplomová práca sa zaoberá návrhom frameworku schopného prijímať, spracovávať a zároveň generovať a odosielať pakety technológií Bluetooth Low Energy (BLE), Z-Wave a IQRF. Okrem návrhu samotného softvéru sa zaoberá aj výberom hardvéru. Zvoleným hardvérom je Software Defined Radio (SDR) zariadenie HackRF One. Softvérové riešenie je implementované v jazyku C++ a počíta s využitím nástrojov tretích strán.

KLÍČOVÁ SLOVA

BLE, C++, framework, generovanie paketov, generovanie prevádzky, HackRF One, IoT, IQRF, SDR, Z-Wave

ABSTRACT

The point of this master's thesis is to design a framework capable of receiving and processing and also generating and sending packets of BLE, Z-Wave and IQRF technologies. In addition to designing the software itself, it also deals with choosing suitable hardware. Selected hardware is SDR device HackRF One. The software solution is implemented in C ++ and counts on the use of third-party tools.

KEYWORDS

BLE, C++, framework, HackRF One, IoT, IQRF, packet generation, SDR, traffic generation, Z-Wave

ŠVEHLA, Ľ. Framework pro generování provozu v IoT sítích. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2019. 61 s., 0 s. příloh. Diplomová práce. Vedoucí práce: doc. Ing. Ladislav Polák, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma Framework pro generování provozu v IoT sítích jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Chcel by som sa poďakovať vedúcemu svojej diplomovej práce Ing. Ladislavovi Polákovi, Ph.D a odbornému konzultantovi RNDr. Radkovi Krejčímu za odborné vedenie, trpezlivosť a cenné rady pri spracovaní práce.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Internet of Things (IoT) je oblasť, ktorá sa dnes veľmi rýchlo rozvíja a hovorí sa o nej ako o technológií budúcnosti. Tento koncept zasahuje do rôznych oblastí od infraštruktúry, cez priemysel až po domácnosť. Využíva sa napríklad pre automatizáciu domácností, zber senzorických dát v odľahlých oblastiach, komunikáciu smartfónov s nositeľnými zariadeniami, alebo v automobilovej oblasti pre zvýšenie bezpečnosti a plynulosti premávky. Rôzne IoT technológie sa špecializujú na rôzne oblasti použitia. Niektoré sú cielené na čo najnižšiu spotrebu, iné na čo najväčší dosah a ďalšie na vysoký dátový tok.

S rozvojom využívania IoT rastie aj hrozba útokov na IoT siete. Z toho dôvodu výrobcovia kladú čoraz väčší dôraz na jej zabezpečenie. Napriek tomu sú stále známe viaceré bezpečnostné nedostatky u rôznych protokolov. Tie môžu byť spôsobené nevhodnou implementáciou konkrétneho výrobcu, ale aj chybou samotného štandardu.

Jednou z motivácií tejto práce je práve vytvoriť vhodné prostredie pre testovanie bezpečnosti jednotlivých technológií v IoT oblasti. Zariadenie by však mohlo byť napríklad využité aj ako štandardný prvok siete, buďto v roli koncového, alebo riadiaceho uzlu. V prípade imitácie senzoru by zariadenie bolo schopné generovať štandardné senzorické dáta, ktoré bude následne odosielať centrálnemu uzlu. V prípade imitácie riadiaceho uzlu by zariadenie naopak bolo schopné monitorovať dáta z okolia a spravovať sieť.

Cieľom tejto práce je navrhnúť a vytvoriť systém schopný generovať pakety vybraných IoT protokolov a následne ich odosielať. Pre generovanie komunikácie je v mnohých prípadoch potrebné zistiť údaje o sieti do ktorej sú pakety generované. Z toho dôvodu je potrebné aby systém dokázal v prípade potreby dáta aj prijímať a spracovať tak aby z nich získal užitočné informácie.

Systém je vytvorený pomocou kombinácie vhodného hardvéru so softvérovým riešením. Z toho dôvodu je pred samotným návrhom softvéru potrebné vybrať vhodné zariadenie, na ktoré bude následne softvér vyvíjaný. Navrhnutý systém bude potom implementovaný a otestovaný.

Práca je rozdelená do 5 hlavných kapitol.

V prvej kapitole tejto práce sú preskúmané 3 z technológií využívaných v oblasti IoT. Jedná sa o technológie *Bluetooth Low Energy* (BLE), Z-Wave a IQRF. Všetky tieto technológie sa využívajú okrem iného v oblasti automatizácie domácnosti, či kancelárskych priestorov.

V ďalšej kapitole je na základe požiadavkov na systém vybrané najvhodnejšie spomedzi dostupných zariadení, ktoré sú vhodné pre generovanie komunikácie u zadaných a prípadne ďalších IoT technológií.

Tretia kapitola obsahuje stručný popis existujúcich projektov v tejto oblasti, ktoré sú využité pre realizáciu tejto práce. Táto kapitola takisto obsahuje popis navrhnutého *Application Programming Interface* (API).

V štvrtej kapitole je detailnejší popis implementácie jednotlivých funkcií a štruktúr nástroja.

Piata kapitola obsahuje výsledky testovania implementovaného systému. Sledovaná je úspešnosť príjmu a odosielania a funkčnosť jednotlivých častí nástroja.

1 Rozbor protokolov

Táto kapitola obsahuje súhrn informácií o vybraných komunikačných protokoloch. Konkrétne sa zameriava na tri IoT protokoly, ktoré sú využívané v tejto práci. Sú to protokoly BLE, Z-Wave a IQRF.

1.1 Bluetooth Low Energy (BLE)

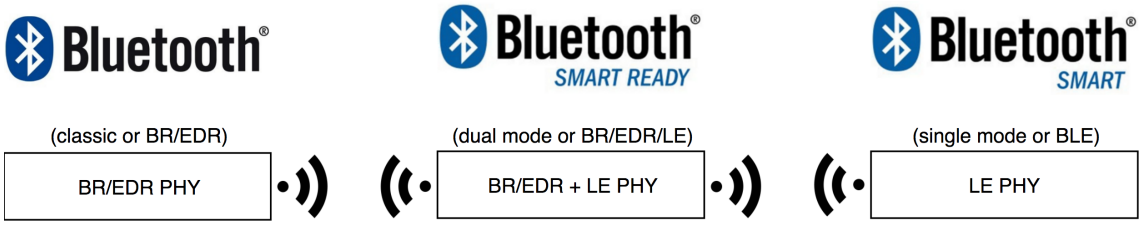
1.1.1 História

Bluetooth je sada protokolov určená pre bezdrôtovú komunikáciu zariadení. Bluetooth začala vyvíjať v roku 1994 švédska spoločnosť Ericsson ako bezdrôtovú náhradu rozhrania RS-232. V roku 1996 sa Ericsson v záujme univerzálnej interoperability spojil vo vývoji so spoločnosťami Nokia a Intel a dohodli sa na vytvorení skupiny známej ako *Bluetooth Special Interest Group* (Bluetooth SIG). Prvá verzia protokolu bola vydaná v roku 1999. Bluetooth SIG následne rástol a rozširoval technológiu Bluetooth o nové protokoly a mechanizmy.

V roku 2007 Bluetooth SIG získal Wibree Alianciu. Projekt vedený spoločnosťou Nokia, ktorého cieľom bolo vyvinúť formu bezdrôtovej komunikácie s veľmi nízkou spotrebou energie.

Následne s vydaním novej verzie 4.0 v roku 2010 vznikol úplne nový a s pôvodným protokolovým zásobníkom nekompatibilný BLE. Ten kladie dôraz na energetickú nenáročnosť komunikácie, čo ho činí ideálnym pre aplikácie v IoT. Vzhľadom na perspektívu BLE, najnovšia verzia Bluetooth štandardu, verzia 5.0, nepriniesla žiadne zmeny do klasického Bluetooth, známeho tiež ako (*Bluetooth Basic Rate/Enhanced Data Rate* (BR/EDR)), ale sústredí sa len na BLE.

Zariadenia BR/EDR a BLE nie sú priamo kompatibilné a Bluetooth zariadenia špecifikácie staršej ako 4.0 so zariadeniami BLE nie sú schopné komunikovať žiadnym spôsobom. Vzhľadom na to začali byť vyvíjané 2 druhy zariadení podporujúce BLE, zariadenia *single mode* a zariadenia *dual mode*. *Single mode* zaradenia, ktoré podporovali čiste BLE sa predávali pod obchodnou značkou *Bluetooth Smart* a *dual mode* zariadenia, ktoré komunikujú okrem BLE aj s BR/EDR sa predávali pod značkou *Bluetooth Smart Ready*. Vzťah medzi ich fyzickými vrstvami je znázornený na Obr. 1.1. *Bluetooth Smart* sa neskôr premenoval na BLE a *Bluetooth Smart Ready* je dostupný pod označením *dual mode* [1].

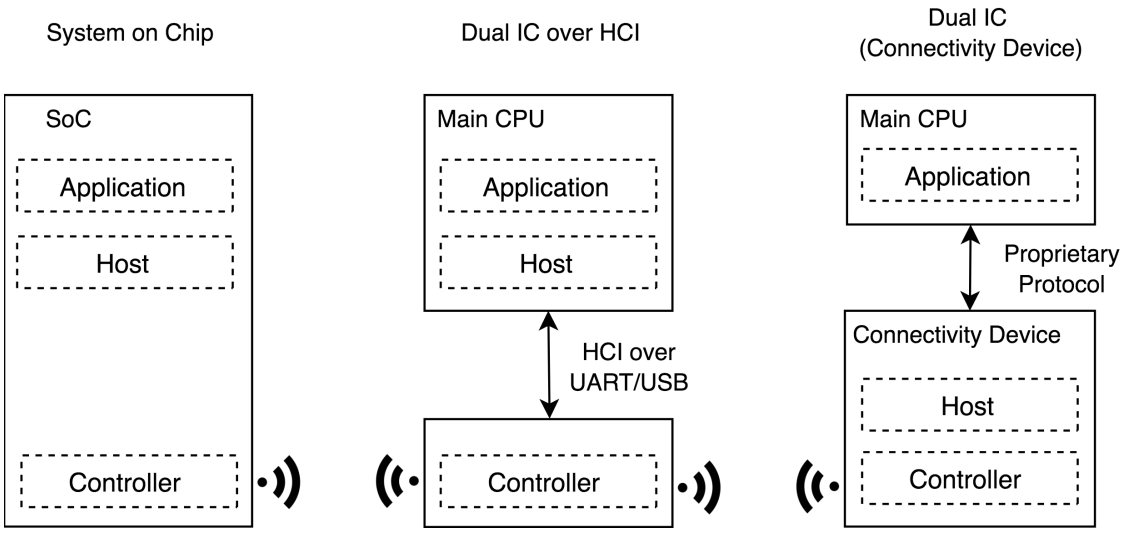


Obr. 1.1: Interoperabilita jednotlivých typov Bluetooth zariadení (na základe [2])

1.1.2 Technická špecifikácia

Sada protokolov Bluetooth obsahuje 3 základné bloky. Patrí sem *application block*, ktorý zahŕňa rozhranie s užívateľskou aplikáciou. Ďalej *host block*, ktorý spravuje vyššie vrstvy a *controller block*, ktorý zahŕňa najnižšie vrstvy vrátane rádia.

Tieto bloky bývajú hardvérovo implementované troma základnými spôsobmi. Tými sú implementácia v podobe *System on Chip*, ďalej implementácia v podobe dvoch integrovaných obvodov s *Host Controller Interface* (HCI), alebo v podobe dvoch integrovaných obvodov s pripojovacím zariadením. Jednotlivé konfigurácie sú zobrazené na Obr. 1.2



je preskúmať požadované technológie z oblasti IoT. Následne

Obr. 1.2: Hardvérové konfigurácie implementácie blokov BLE (prevzaté z [2])

V senzoroch sa z hľadiska čo najnižších nákladov zvykne využívať konfigurácia *System on Chip*. Verzia 2 integrovaných obvodov s HCI je využívaná prevažne u tabletov a smartfónov.

HCI je špeciálny protokol, ktorý je štandardizovaný Bluetooth SIG. Tento protokol zabezpečuje interoperabilitu *host* blokov rôznych zariadení s Bluetooth *controller* blokmi od rôznych výrobcov. Realizovaný je pomocou rozhraní UART, USB, alebo RS-232.

Bluetooth rozlišuje medzi profilmi a protokolmi. Protokoly sú stavebné bloky, ktoré implementujú jednotlivé časti ako je smerovanie, alebo dekódovanie. Profily sú vyššou vrstvou, ktorá určuje, ako by sa mali protokoly pri jednotlivých *use cases* využívať.

Generic Access Profile (GAP) je najvyššou vrstvou BLE, ktorá spravuje riadenie. Je povinný pre všetky zariadenia a spravuje model použitia jednotlivých protokolov na nižších vrstvách.

Generic Attribute Profile (GATT) je najvyššou vrstvou v BLE, ktrorá spravuje dáta. Definuje základný dátový model a spravuje to, ako sa organizujú a vymieňajú dáta na nižších vrstvách.

Security Manager Protocol (SMP) je zároveň protokol aj sada bezpečnostných algoritmov, ktorá slúži na generovanie a odosielanie bezpečnostných kľúčov.

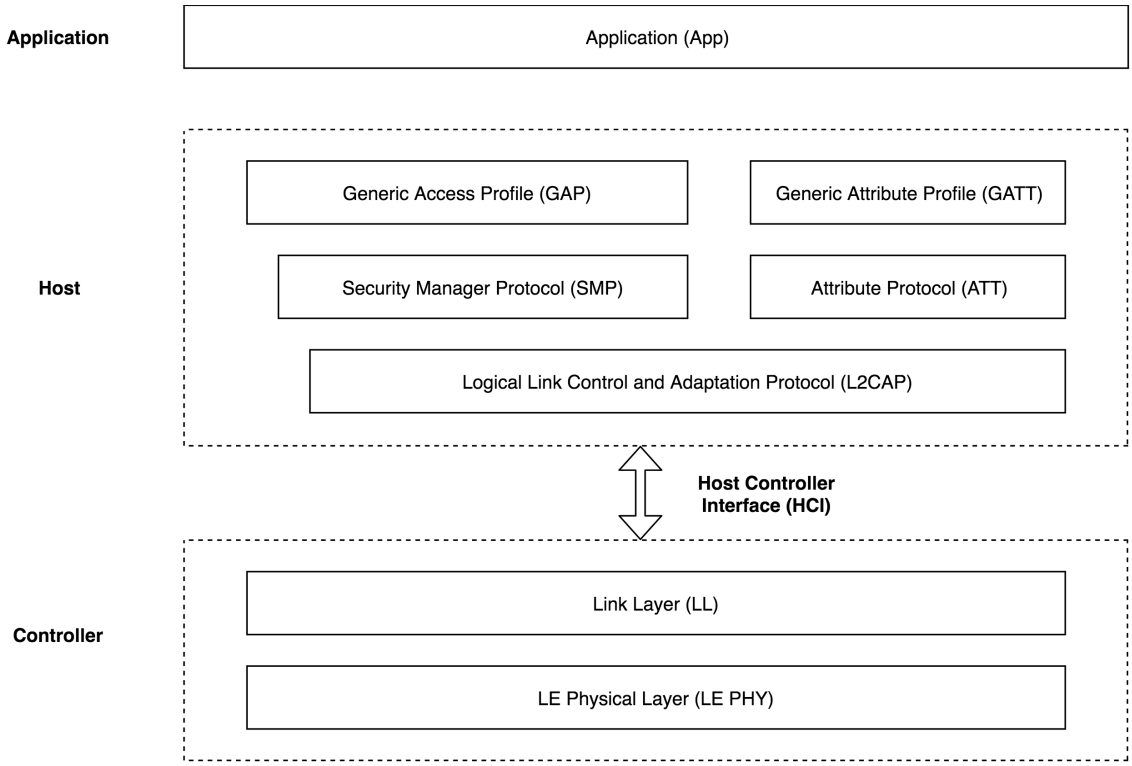
Attribute Protocol (ATT) je protokol typu klient/server, ktorý má na starosti výmenu paketov medzi komunikujúcimi zariadeniami. Klient žiada o dáta od serveru a server následne dáta posiela klientovi. Protokol dohliada na to, aby nebola odoslaná požiadavka na ďalšie dáta pred tým, ako je odpoveď na predošlú požiadavku doručená a spracovaná.

Logical Link Control and Adaptation Protocol (L2CAP) má 2 hlavné funkcie. Prvou z nich je funkcia multiplexeru, ktorá zapúzdruje protokoly vyšších vrstiev do štandardného formátu BLE paketu. Druhou funkciou L2CAP je fragmentácia a rekombinácia. Pomocou fragmentácie zabezpečuje, aby *payload* dáta prenášaných BLE paketov na fyzickej vrstve nepresahovali maximálnu veĺkosť 27 B a pomocou rekombinácie ich po prijatí spätne spojí do jednoho paketu, ktorý odošle vyšším vrstvám.

Link Layer (LL) je vrstva, ktorá priamo komunikuje s fyzickou vrstvou a je jediná vrstva s kritickymi požiadavkami na prácu v skutočnom čase. Vzhľadom na to je zvyčajne od vyšších vrstiev izolovaná pomocou rozhrania HCI, ktoré zapuzdruje jej komplexnosť pri komunikácií s týmito vyššími vrstvami.

LE Physical Layer (LE PHY) má na starosti analógovú ćasť. Je schopná modulovať a demodulovať signály a transformovať ich medzi analógovou a digitálnou podobou [3]. Celá štruktúra jednotlivých vrstiev BLE je zobrazená na Obr. 1.3.

Z dôvodu nízkej energetickej náročnosti je BLE oproti BR/EDR limitované v iných smeroch, ako je napríklad maximálny dátový tok. Porovnanie základných parametrov BLE a BR/EDR je v Tab. 1.1. Maximálna teoretická rýchlosť prenosu je 2 Mb/s.



Obr. 1.3: Vrstvy BLE (prevzaté z [2])

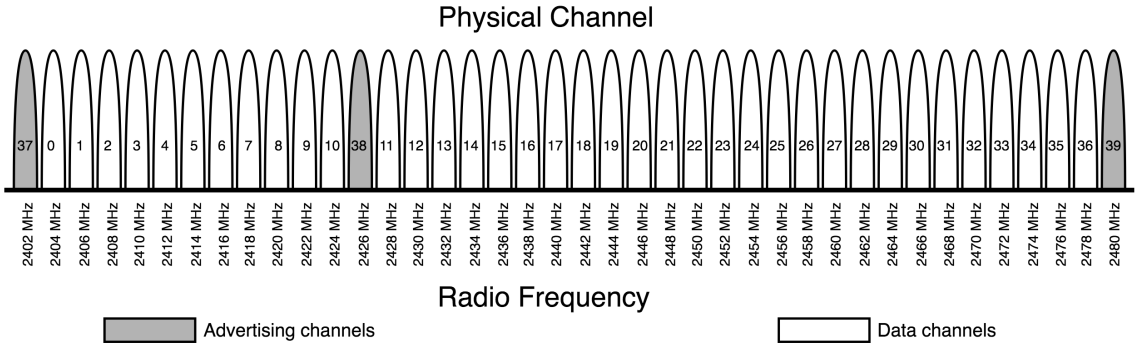
BLE používa rovnaké komunikačné pásmo ako klasický Bluetooth. Je to pásmo medzi 2,402 GHz a 2,48 GHz a je rozdelené na 40 kanálov s šírkou 2 MHz. Medzi týmito 40 kanálmi sú tri *advertising* kanály, na ktorých prebieha nadväzovanie spojenia a *broadcast* komunikácia. Ďalej je tu 37 dátových kanálov pre komunikáciu medzi prepojenými zariadeniami. *Advertising* kanály sú číslované indexami 37 až 39 a ich rozmiestnenie v spektre je zachytené na Obr. 1.4. BLE využíva moduláciu *Gaussian Frequency-Shift keying* (GFSK).

Pre zaistenie simultánneho prenosu viacerých spojení a zabránenie interferencii je použitá technika *frequency hopping*. Spojené zariadenia pri využití *frequency hopping* môžu počas komunikácie meniť kanály na ktorých vysielajú, a to až 1600 krát za sekundu [6]. Skákanie sa riadi vzorcom:

$$channel = (curr\_channel + hop)\ mod\ 37. \tag{1.1}$$

Channel označuje index nasledujúceho kanálu a *curr_channel* označuje index aktuálneho kanálu. Hodnota hop predstavuje počet kanálov o ktoré sa skáče a je definovaná pri vytvorení spojenia a je špecifická pre každé spojenie.

Každé BLE zariadenie je identifikované unikátnou 48 b MAC adresou. Počas komunikácie sa však využíva len spodná polovica tejto adresy. Pri zapnutí rozšírenia



Obr. 1.4: Rozloženie kanálov v spektre

LE Privacy feature sa navyše zariadenie hlási na advertising kanáloch so stochasticky generovanými adresami.

V závislosti od vzdialenosti medzi zariadeniami sú využívané 3 triedy zariadení, sú to *Class 1*, *Class 2* a *Class 3*. Zariadenia *Class 1* sú schopné prenášať dáta na vzdialenosti do 100 metrov. Zariadenia certifikované ako *Class 2* sú schopné vysielať v okruhu maximálne 10 m a u zariadení energeticky najúspornejšej *Class 3* je to len 1 meter. Maximálny vysielaný výkon u triedy 3 je 0 dBm [2, 5, 6, 3].

1.1.3 Topológia

BLE využíva 2 spôsoby komunikácie. Sú to *broadcast* a *point-to-point*. Výhodou *broadcast* komunikácie je možnosť odosielať pakety väčšiemu množstvu zariadení súčasne. Nevýhodou je, že dáta nie sú oproti *point-to-point* komunikácií vôbec zabezpečené a že dáta môžu prúdiť len jedným smerom.

Pri *broadcast* komunikácií vysielajúce *broadcaster* zariadenie odosiela tzv. *non-connectable advertising* pakety všetkým zariadeniam v dosahu. *Observer* zariadenia následne nastavenú frekvenciu skenujú a prijímajú na nej pakety. Pokiaľ *observer* zaznamená prítomnosť *broadcaster* zariadenia, môže mu poslať žiadosť o odoslanie ďalších dát. Tieto dáta mu *broadcaster* odošle prostredníctvom *scan response* paketu. Pre komunikáciu sa u *broadcast* topológie využívajú výhradne *advertising* kanály.

Pri *point-to-point* komunikácií nie sme limitovaní množstvom prenášaných dát, ani smerom komunikácie. Jedno zariadenie je *master*, ktoré skenuje prednastavenú frekvenciu a hľadá *advertising* pakety so žiadosťou o pripojenie. Keď ich získa tak iniciuje spojenie a spravuje časovanie a výmenu dát. Na druhej strane je *slave* zariadenie, ktoré periodicky odosiela *advertising* pakety so žiadosťou o pripojenie a prijíma prichádzajúce spojenie. Následne sa riadi časovaním od *master* zariadenia. *Master* k sebe môže mať pripojených viacero zariadení, ktoré však medzi sebou môžu komunikovať jedine prostredníctvom *mastera*. Vzniká tu teda hviezdicová sieť.

Tab. 1.1: Porovnanie parametrov BLE a BR/EDR

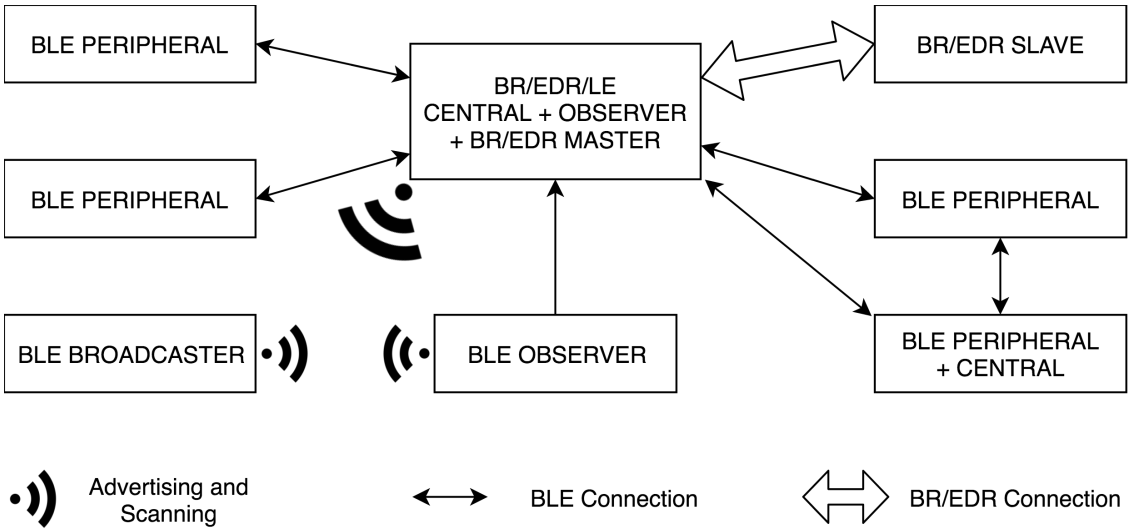
	BLE	BR/EDR
Spôsob vysielania dát	vysielanie v krátkych sledoch impulzov	kontinuálny tok dát
Frekvenčné pásmo	2,402 – 2,480 GHz	2,402 – 2,480 GHz
Kanály	40 kanálov s šírkou 2 MHz	79 kanálov s šírkou 1 MHz
Modulácia	GFSK	GFSK, π/4 DQPSK, 8DPSK
Spotreba energie	0.01x až 0.5x referenčnej hodnoty	1 W (referenčná hodnota)
Maximálny vysielaný výkon	Class 1: +20 dBm, Class 1.5: +10 dBm, Class 2:+4 dBm, Class 3: 0 dBm	Class 1: +20 dBm, Class 2:+4 dBm, Class 3: 0 dBm
Maximálny dátový tok	LE 2M PHY: 2 Mb/s LE 1M PHY: 1 Mb/s LE Coded PHY (S=2): 500 Kb/s LE Coded PHY (S=8): 125 Kb/s	EDR PHY (8DPSK): 3 Mb/s EDR PHY (π/4 DQPSK): 2 Mb/s BR PHY (GFSK):1 Mb/s
Sieťové topológie	Point-to-Point, Broadcast, Mesh	Point-to-Point

Od verzie 5 je možná pre komunikáciu v BLE aj Mesh topológia. Tá umožňuje vytvorenie veľkej siete, v rámci ktorej môžu zariadenia voľne komunikovať. Takisto je možné kombinovať v zariadení funkčnosť *broadcast* a *point-to-point* komunikácie. Príklad zmiešanej topológie je na Obr. 1.5 [2, 6, 3].

1.1.4 Zabezpečenie

V BLE sú definované dva bezpečnostné módy s výlučným použitím. Mód 1 definuje šifrovanie dát a mód 2 zabezpečuje integritu dát pomocou podpisovania. Mód 1 zahŕňa štyri bezpečnostné úrovne, kde úroveň 1 znamená žiadne zabezpčenie a úroveň 4 maximálne. U tej je vyžadované šifrovanie a použitie bezpečnostného párovania pomocou *Secure Connections*.

K výmene kľúčov pre šifrovanie dochádza procesom párovania, pri ktorom sa vytvorí *Long Term Key* (LTK), ktorý je využívaný pri šifrovaní výmeny linkového kľúča pre šifrovanie komunikácie. LTK sa na zariadenie uloží pri prvom párovaní. Pri nasledujúcom pripojení sa pomocou procesu bonding len vymenia linkové kľúče. Pre prípad zabudnutia LTK, napríklad pri vybití batérie senzoru existuje v protokole



Obr. 1.5: Zmiešaná topológia BLE siete (prevzaté z [2])

príkaz LL_REJECT_IND, ktorý vynúti jeho pregenerovanie.

BLE umožnuje štyri metódy párovania. sú to metódy *Out of Band* (OOB), *Passkey Entry*, *Just Works* a *Numeric Comparison*.

OOB je metóda, pri ktorej k výmene *Temporary Key* (TK) pre generovanie LTK dochádza pomocou iného komunikáčného kanálu, ako je napríklad *Near-field communication* (NFC).

Passkey Entry je metóda pri ktorej užívateľ zadá na jednom zariadení šesťmiestne číslo, ktoré je zobrazené na druhom zariadení. Z tohoto čísla sa potom generuje TK.

Tretia metóda s názvom *Just Works* nevyžaduje prístup užívateľa a TK je nastavené na hodnotu 0.

V poslednej z metód, *Numeric Comparison*, užívateľ potvrdzuje zhodu šesťmiestnych čísel zobrazených na oboch zariadeniach. Na rozdiel od *Passkey Entry* sa TK generuje nezávisle na zobrazenom čísle.

Párovanie vo verziách BLE starších ako 4.2 je označované *LE Legacy Pairing*. Pri tomto párovaní dochádza ku generovaniu kľúčov na jednom zariadení a následnej distribúcii týchto kľúčov. Počas tohoto procesu nie je spojenie nijako chránené pred odpočúvaním.

Od verzie 4.2 však bolo zavedené bezpečné párovanie nazývané *Secure Connections* (SC), ktoré používa odvodenie kľúčov pomocou eliptických kriviek (ECDS) na oboch zariadeniach. Vďaka tomu, že nedochádza k posielaniu kľúčov, je SC odolné proti odpočúvaniu.

Pre šifrovanie je v BLE použitá bloková šifra AES s blokmi o veľkosti 128 bitov, a to v *counter* (CTR) móde, alebo v *counter with CBC* (CCM) móde pre zaiste-

nie integrity. U zariadení, ktoré podporujú SC je možné navyše zapnúť *SC Only* mód, ktorý vyžaduje SC a odmieta spojenie so zariadeniami, ktoré ich nepodporujú. Tým je zaistené najvyššie možné zabezpečenie BLE spojenia na úkor spätnej kompatibility [2, 6].

1.2 Z-Wave

1.2.1 História

Z-Wave technológia začala vznikať v roku 2001 ako systém pre kontrolu osvetlenia. Technológia bola vyvíjaná dánskou spoločnosťou Zensys.

V roku 2005 sa spojilo viacero výrobcov v oblasti automatizácie domácností za účelom zjednotiť dovtedy roztrieštený trh v tejto sfére a zaviesť jednotnú technológiu pre komunikáciu zariadení od rôznych výrobcov. To by im umožnilo jednoduchšiu štandardizáciu a zároveň zníženie nákladov. Pre realizáciu sa rozhodli využiť technológiu Z-Wave a to viedlo k vzniku Z-Wave aliancie.

Technológia sa stretla s úspechom a podporila rozvoj IoT v oblasti automatizácie domácnosti. Dnes je v tejto oblasti stále najrozšírenejšou technológiou [7].

1.2.2 Technická špecifikácia

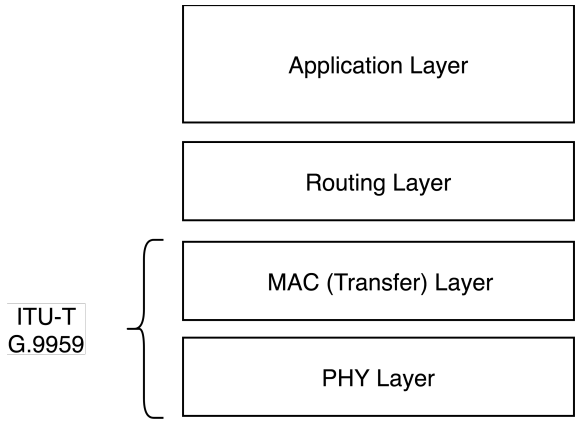
Protokol Z-Wave komunikuje v pásme ISM (868 MHz v Európe a 908 MHz v USA). V Európe sú konkrétne využité frekvencie 868.4 MHz, 868.42 MHz a 869.85 MHz. Využíva moduláciu GFSK a v závislosti od typu transceiveru sa využíva prenosová rýchlosť 9,6 Kb/s, 40 Kb/s, alebo 100 Kb/s. pri prenosovej rýchlosti 9,6 Kb/s je využité kódovanie typu Manchester, pri zvyšných dvoch *Non return to zero* (NRZ).

Open source implementácia časti protokolu Z-Wave je dostupná pod názvom *open-zwave*. Tá je však zameraná najmä na tvorbu aplikácií pre kontroléry pripojiteľné k PC. Z-Wave protokol sa skladá zo 4 vrstiev: fyickej, transportnej, smerovacej a aplikačnej vrstvy. Štruktúra protokolu je zobrazená na Obr. 1.6. Štruktúra paketov Z-Wave je zachytená na Obr. 1.7.

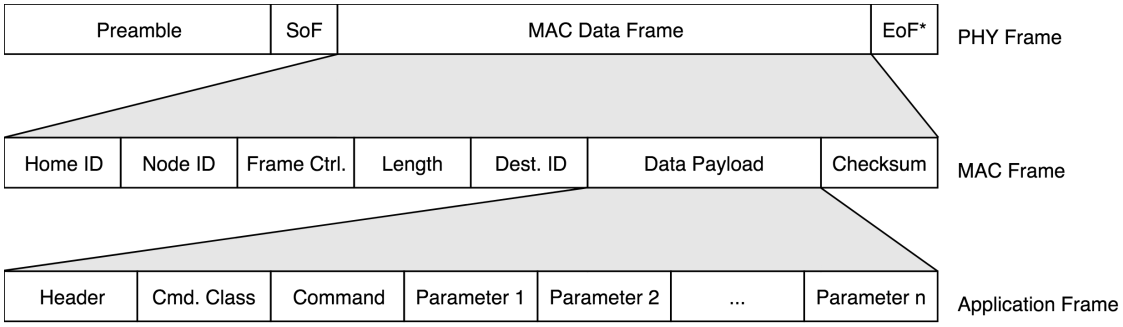
Fyzická a transportná vrstva protokolu je založená na ITU-T G.9959 špecifikácií, čo zabezpečuje kompatibilitu zariadení od rôznych výrobcov.

Rámce fyzickej vrstvy zabezpečujú synchronizáciu. Štandardne sa skladajú zo synchronizačnej preambule, štartovacej sekvencie (SoF) a *payload* dát, v ktorých je obsiahnutý rámec transportnej vrstvy. V prípade využitia prenosovej rýchlosti 9,6 Kb/s obsahujú naviac jedno špeciálne pole, ktoré označuje koniec rámca (EoF).

Transportná vrstva je zodpovedná za preposielanie rámcov, validáciu dát, potvrdzovanie prijatia dát a prebúdzanie zaradení s nízkou spotrebou. Obsahuje údaje



Obr. 1.6: Štruktúra protokolu Z-Wave (na základe [7])



Obr. 1.7: Štruktúra paketov Z-Wave (na základe [8])

potrebné pre identifikáciu siete (*Home ID*) a zariadení (*Source ID*, *Dest ID*). Ďalej obsahuje pole *Frame Ctrl*, ktoré obsahuje informácie o type rámca a ďalšie kontrolné príznaky. V ďašom poli s názvom *Lenght* je informácia o dĺžke rámca, ktorá je dôležitá vzhľadom na variabilnú dĺžku *payload* dát. Na konci paketu je obsiahnutý *checksum*, ktorý zabezpečuje validitu dát.

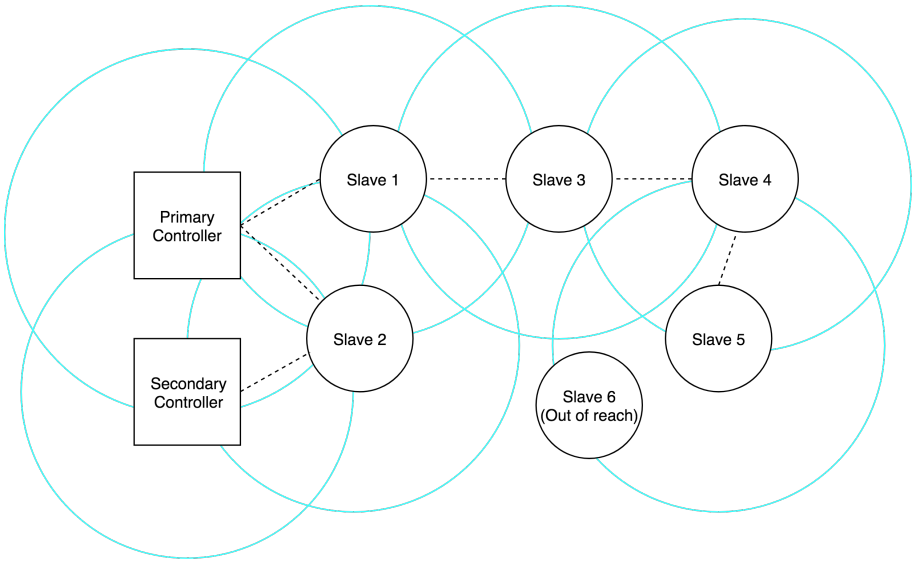
Existujú 3 typy rámcov na transportnej vrstve: *singlecast*, *multicast* a *acknowledgement*. *Singlecast* rámec je odosielaný jednomu zariadeniu, ktoré odpovedá *acknowledgement* rámcom, pričom ten má veĺmi podobnú štruktúru. Ak *acknowledgement* nedorazí, prenos sa opakuje. *Multicast* rámce sú odosielané viacerým zariadeniam, ktoré príjem nepotvrdzujú. Smerovacia vrstva spravuje smerovaciu tabuľku. Tá je v Z-Wave automaticky udržovaná, aby sa predišlo kolíziám v prípade premiestňovania, pridávania, či odstraňovania uzlov.

Aplikačná vrstva spracúva príkazy. Tie sú charakterizované triedou (*Cmd Class*), ktorá zastrešuje nejakú funkcionalitu, alebo zariadenie. Každá z tried zahŕňa potom jeden, alebo viac príkazov. Z-Wave protokol taktiež podporuje zabezpečenie

Advanced Encryption Standard (AES) šifrovaním s 128-bitovým kľúčom. Takto zabezpečený payload je potom nasledovaný autentizačným poľom o veľkosti 8 bajtov [8, 9].

1.2.3 Topológia

V sieti sú 2 základné typy prvkov. Riadiace uzly, ktoré iniciujú príkazy a koncové uzly, ktoré na tieto príkazy reagujú. Vďaka aplikácií *mesh* topológie je možné, aby koncový uzol preposielal správy ďalším uzlom, ktoré nie sú v priamom dosahu riadiaceho uzlu. Maximálne množstvo uzlov v sieti je 232 a maximálny počet skokov medzi uzlami je 4.



Obr. 1.8: Ukážka Z-Wave siete

Každá Z-Wave sieť má unikátne 32 bitové *Home ID*, ktoré je naprogramované pri výrobe do každého riadiaceho uzlu. Vďaka rozlíšeniu pomocou *Home ID* je zabezpečená kooperácia viacerých sietí s vzájomným prekrytím. V sieti môže byť viac riadiacich uzlov, len jeden je však primárny a z neho je odvodená *Home ID*. Ďalšie riadiace uzly majú rovnakú funkcionalitu ako primárny riadiaci uzol, ale nemôžu pridávať a vyraďovať zariadenia zo siete.

Zariadenia v rámci siete sú identifikované pomocou *Node ID*, ktoré im je pridelené pri pripojení do siete a je v rámci siete unikátne [9].

1.2.4 Zabezpečenie

Zabezpečenie protokolu je zaistené podporou AES s 128-bitovými kľúčmi na aplikačnej vrstve. Toto šifrovanie je však voliteľne nastaviteľné a veľa výrobcov ho vo

východzej konfigurácií nenastavuje.

Pri šifrovanej komunikácií sa taktiež u každej správy overuje zdroj vypočítaním overovacieho kódu správy.

Kľúče pre šifrovanie a kontrolu zdroja správy sú odvodené z náhodne vygenerovaného sieťového kľúča. Sieťový kľúč sa prenáša z kontroléru šifrovanou komunikáciou pomocou dočasného kľúča, ktorý je uložený vo *firmware* zariadenia. Vytvorenenie kľúčov prebieha počas procesu pridania nového zariadenia do siete.

Pre prevenciu tzv. *replay* útoku je pri výpočte overovacieho kódu každej správy použitá náhodná hodnota (nonce), ktorá zamedzuje opätovnému použitiu zachyteného paketu [8].

1.3 IQRF

1.3.1 História

IQRF je bezdrôtová platforma pre IoT, ktorú v roku 2004 začala vyvíjať česká spoločnosť MICRORISC. V roku 2017 sa odčlenila dcérska spoločnosť IQRF Tech s.r.o., ktorá stojí za vznikom IQRF Aliancie. Tá združuje spoločnosti a inštitúcie, ktoré s IQRF pracujú a podieľajú sa na jeho vývoji.

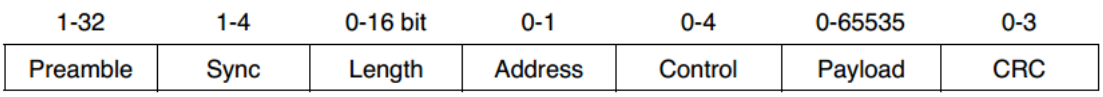
1.3.2 Technická špecifikácia

Dosah IQRF je v ráde desiatok až stoviek metrov (v špeciálnych prípadoch niekoľko kilometrov). IQRF je schopné pracovať v ISM pásmach 433 MHz, 868 MHz a 916 MHz. Komunikácia v pásme 868 MHz, ktoré sa v Európe štandardne využíva, prebieha na 67 kanáloch s prenosovou rýchlosťou 20 kb/s. Využitá je modulácia GFSK.

Transceiver, ktorý má vbudovaný *Operačný systém* (OS) je plne užívateľsky programovateľný v jazyku C pomocou funkcií OS a to bezdrôtovo, alebo pomocou *Serial Peripheral Interface* (SPI) komunikácie. Všetky IQRF transceivere majú povolený upgrade na vyššiu verziu OS. Downgrade je povolený len u niektorých verzií, u väčšiny verzií však môže byť vykonaný len u výrobcu.

Aktuálne Transceivery pre komunikáciu využívajú integrovaný obvod SPIRIT1 od spoločnosti STMicroelectronics. Ten sa stará aj o nižšie vrstvy protokolu [13]. Štruktúra fyzickej vrstvy paketu je znázornená na Obr. 1.9.

Pakety vyšších vrstiev majú podobu ako na Obr. 1.10. Protokol IQMESH rozlišuje 2 typy paketov, ktoré však majú rovnakú kostru. Tú možno vidieť na Obr. 1.10. Kostra sa skladá zo štyroch hlavných blokov, ktoré obsahujú ďalšie podbloky.



Obr. 1.9: Štruktúra IQRF fyzickej vrstvy paketu (prevzaté z [13])

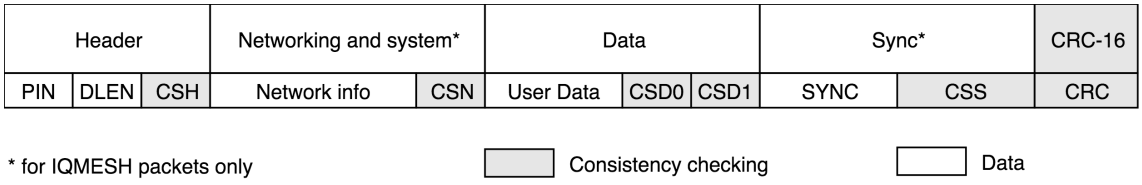
V bloku *Header* je v podbloku *DLEN* informácia o dĺžke paketu. V jeho druhom podbloku *PIN* sú obsiahnuté informácie o príznakoch, ktoré určujú vlastnosti a zároveň, ktoré podbloky sú obsiahnuté v bloku *Networking and system*.

Blok *Networking and system* je špecifický pre IQMESH pakety a jeho dĺžka je variabilná v závislosti od príznakov bloku *Header*. Obsahuje detailnejšie informácie, ktoré sú potrebné namiesto samotných príznakov v prípade, že sa nejedná o jednoduchý peer-to-peer paket.

Blok *Data* obsahuje užívateľské dáta a jeho dĺžka je variabilná v rozmedzí 0 B až 255 B.

Blok *Sync*, ktorý je opäť špecifický pre IQMESH pakety slúži pre synchronizáciu pri smerovaní.

To, či sa jedná o Peer-to-peer paket, alebo IQMESH paket, je dané najvyšším bitom v bloku PIN. Dĺžka IQMESH paketu je variabilná. Protokol IQMESH je proprietárny a tak sú informácie o štruktúre paketov mimo najvyššie vrstvy obmedzené [10, 14, 13].



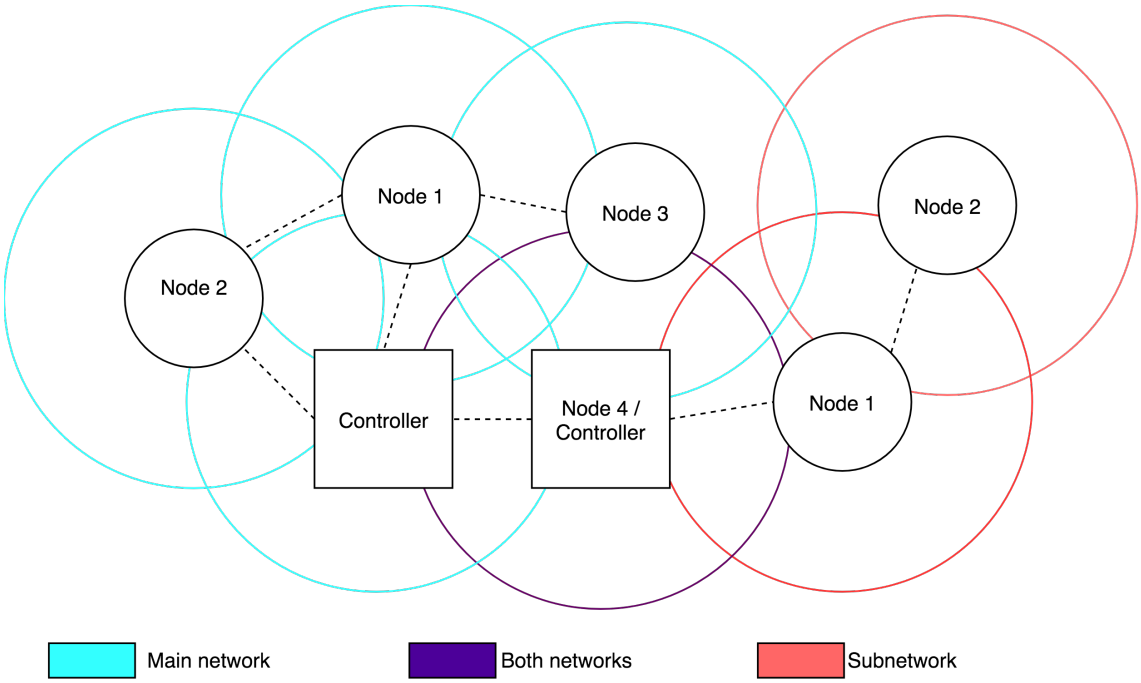
Obr. 1.10: Štruktúra IQRF paketu na vyšších vrstvách (na základe [10])

1.3.3 Topológia

V IQRF sú 2 základné topológie: *Peer-to-peer* a *MESH* (ktorá je implementovaná proprietárnym protokolom IQMESH).

Peer-to-peer mód je prednastavený a zariadenia v ňom komunikujú bez koordinačného prvku s tým že vysielané správy sú určené všetkým ostatným modulom. Sieť v *peer-to-peer* móde nie je obmedzená maximálnym počtom užívateľov.

MESH je topológia s jedným koordinátorom, ktorý spravuje komunikáciu s ostatnými zariadeniami (uzlami), ktorých môže byť až 239. Každý uzol môže fungovať



Obr. 1.11: Štruktúra IQMESH siete.

ako smerovač pre komunikáciu s uzlami mimo priameho dosahu koordinátora. Pokiaľ by bol počet zariadení v sieti nedostačujúci, je možné škálovať sieť pomocou nastavenia cieľových zariadení ako subkoordinátorov, pre sub-sieť. Každá sub-sieť môže obsahovať takisto 239 uzlov. Štruktúra *IQMESH* siete je znázornená na Obr. 1.11

Zariadenia sú do siete pridávané/odoberané pomocou *bonding/unbonding* procesu. Po úspešnom *bondingu* získa zariadenie svoj *Identifikátor* (ID) v rámci siete, ID siete a *networking password*. Následne je pomocou *discovery* procesu vytvorená štruktúra pre smerovanie správ v rámci siete. Uzly môžu komunikovať kedykoľvek, no v typických aplikáciách komunikáciu vždy inicializuje koordinátor [10, 14].

1.3.4 Zabezpečenie

Všetka *IQMESH* komunikácia je šifrovaná. Mimo užívateľskej komunikácie môžu byť touto komunikáciou posielané aj systémové správy.

Na zabezpečenie sú v IQRF využité 3 rôzne prvky založené na AES-128. Prvým je *networking* šifrovanie, ktoré šifruje veškerú komunikáciu v sieti. Druhým prvkom je *access* šifrovanie, ktoré je využívané pri párovaní nových zariadení, pri bezdrôtovej údržbe siete a aktualizácií a pri zálohovaní a obnove nastavení. Posledným prvkom je user šifrovanie, ktoré slúži pre prípadné dodatočné zašifrovanie *payload* dát.

Network key a *Access key* nie sú uložené priamo v zariadení, ale sú dynamicky generované pomocou hashovacích funkcií z hesiel, ktoré sú zadané výrobcom (*Ne-*

twork password) respektíve užívateľom (*Access password*). *Access password*, rovnako ako *User key* využívajú 16 B dlhé kľúče s využitím algoritmu *Cipher Block Chaining* (CBC). Sú zadávané užívateľom a majú variabilnú dĺžku od 0 do 16 znakov s prednastavenou hodnotou 16 krát znak 0x00. *Network password* má takisto 16 B dlhé kľúče no využíva proprietárny algoritmus *Cipher Data Chaining* (CDC). Každý transceiver má unikátne 192 b heslo, z ktorého je generovaný 128 b kľúč [10].

2 Výber zariadenia

2.1 Kritériá výberu

Pre realizáciu diplomovej práce je potrebné, aby použité zariadenie dokázalo vysielať na frekvenciách v pásmach 433 MHZ, 868 MHz a 2,4 GHz. Ďalším dôležitým faktorom bolo, aby všetky tieto frekvencie dokázali byť obslúžené jedným zariadením.

V prípade budúceho rozšírenia frameworku je taktiež vhodné, aby podporovalo aj ostatné frekvencie využívané v IoT. Jedná sa o frekvencie 100 MHz až 5,8 GHz. Takisto by malo byť zariadenie schopné spracovať signály s rôznymi druhmi modulácií.

Na základe požadovaných vlastností a parametrov zariadenia je zvolená technológia *Softvérovo definované rádio* (SDR).

Pri výbere konkrétneho SDR zariadenia je kladený dôraz na hardvérové parametre, ale aj na ďalšie faktory. Medzi tieto faktory patrí dostupnosť zariadení v Európe, cena týchto zariadení, softvérová podpora od výrobcu a veľkosť komunity okolo jednotlivých zariadení. Takisto je braný ohľad na skúsenosti s danými zariadeniami pri iných výskumných činnostiach.

2.2 Porovnanie vybraných dostupných zariadení

Z dostupných SDR boli napokon pre konečné porovnanie vybrané zariadenia HackRF One, Ettus B210, BladeRF x40 a LimeSDR. Prehľad základných parametrov jednotlivých SDR je v Tab. 2.1

Najlepšie SDR, ktoré sme po hardvérovej stránke našli je zariadenie Ettus B210. Jedná sa o *full-duplex* 2x2 *Multiple-input and multiple-output* (MIMO) zariadenie. Pracuje v pásme 70 MHz až 6 GHz, teda v celej časti rádiofrekvenčného pásma, kde sú IoT technológie využívané. Jeho vzorkovacia frekvencia, ktorá je limitovaná použitím *Universal serial bus* (USB) 3 je 61,44 MHz, čo je najviac spomedzi všetkých porovnávaných zariadení. Táto hodnota však stále nepostačuje pre súčasné pokrytie celého pásma v ktorom vysiela BLE. Vysielaný výkon, ktorý je v celom pásme vyšší ako 10 dBm zaručuje veľký dosah. Cena zariadenia je podstatne vyššia ako u ostatných porovnávaných zariadení. Potenciál zariadenia Ettus B210 by navyše v tejto práci nebol naplno využitý. Cena takisto limituje rozšírenosť tohoto zariadenia, z čoho vyplýva menšia odborná komunita okolo tohoto zariadenia.

Ďalším zariadením je BladeRF x40. Na rozdiel od Ettus B210 má len jeden kanál pre príjem a jeden pre vysielanie. Frekvenčný rozsah tohoto SDR pokrýva rádiofrekvenčné pásmo využívané technológiami, ktoré sú využité v tejto práci, no nepokrýva celé rádiofrekvenčné pásmo využívané v IoT. Ďalšie parametre sú takisto

v porovnaní s predošlým zariadením slabšie. Cenovo je však zariadenie výrazne dostupnejšie ako Ettus B210. Takisto je vďaka svojej cene a dlhšej dobe dostupnosti na trhu populárne medzi verejnosťou a využívané.

Tretím SDR, ktoré sme porovnávali je LimeSDR. Podobne ako u zariadenia Ettus B210 sa jedná o *full-duplex* zariadenie s 2x2 MIMO a s rovnakou vzorkovacou frekvenciou. Frekvenčný rozsah tohoto zariadenia je podobne ako u BladeRF x40 dostatočný pre pokrytie rádiofrekvenčného pásma využívaného u technológií v tejto práci, no nie pre pokrytie celého rádiofrekvenčného pásma využívaného v IoT. Vysielaný výkon je väčší ako 0 dBm. Zariadenie sa však začalo predávať relatívne nedávno prostredníctvom webovej stránky, ktorá sa zaoberá skupinovým financovaním projektov. V Európe tým pádom dostupné nie je a takisto nie je zatiaľ tak rozšírené medzi komunitou.

Posledným SDR, ktoré sme porovnávali je zariadenie HackRF One. Zariadenie je na rozdiel od ostantých zariadení *half-duplex*. Frekvenčný rozsah podobne ako Ettus B210 pokryje celé rádiofrekvenčné pásmo využívané IoT technológiami. Vzorkovacia frekvencia zariadenia je limitovaná použitím USB 2 na 20 MHz a je spomedzi všetkých porovnávaných zariadení najmenšia. Vysielaný výkon pri vysokých frekvenciách dosahuje len -15 dBm, no v nami využívaných pásmach sa pohybuje na úrovni 15 dBm. Zariadenie v istej miere funguje od roku 2014 a bolo jedným z prvých cenovo dostupných SDR zariadení tejto kategórie. Vďaka tomu sa stalo veľmi obľúbené medzi komunitou a existuje viacero projektov, ktoré súvisia so zachytávaním a generovaním komunikácie v IoT pomocou tohoto zariadenia.

Spomedzi vybraných zariadení sme napokon vybrali HackRF One. Jeho hlavnými výhodami oproti ostatným zariadeniam sú jeho dostupnosť, frekvenčný rozsah, cena a jeho rozšírenosť.

2.3 HackRF One

HackRF One je *open hardware* zariadenie vyvinuté Michaelom Ossmannom a jeho tímom Great Scott Gadgets. Toto zariadenie je znázornené na Obr. 2.1. Jedná sa o druhú hardvérovú platformu v rámci projektu HackRF.

Prvé HackRF zariadenie sa volalo Jawbreaker. Jednalo sa o beta platformu, ktorá bola na trhu dostupná pred doladením a uvedením HackRF One.

Zdrojové kódy HackRF, slúžiace pre ovládanie hardvéru sú voľne dostupné na stránkach projektu [15] .

Tab. 2.1: Prehľad parametrov vybraných SDR

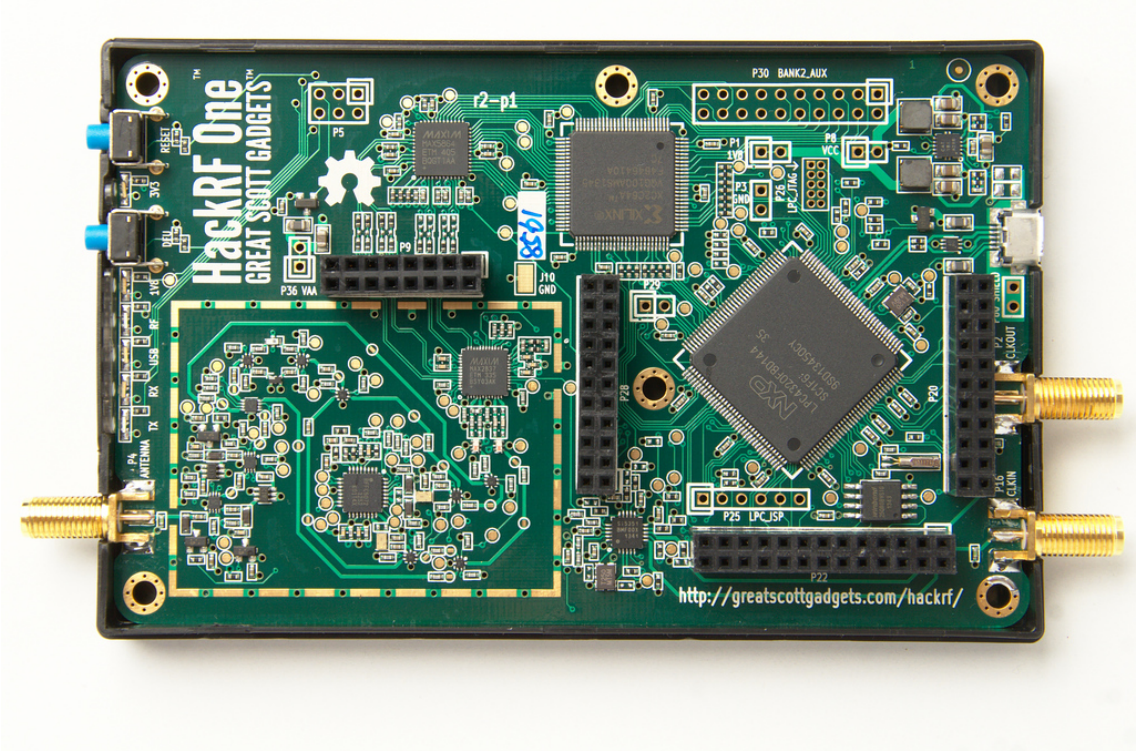
	HackRF One	Ettus B210	BladeRF x40	LimeSDR
Frekvenčný rozsah	1MHz-6GHz	70MHz-6GHz	300MHz-3.8GHz	100kHz-3.8GHz
Bitová hĺbka	8 bits	12 bits	12 bits	12 bits
Vzorkovacia frekvencia	20MHz	61.44MHz	40MHz	61.44MHz
Vysielacie kanály	1	2	1	2
Prijímacie kanály	1	2	1	2
Duplex	Half	Full	Full	Full
Interface	USB 2.0	USB 3.0	USB 3.0	USB 3.0
Chipset	MAX5864, MAX2837 , RFFC5072	AD936	LMS6002M	LMS7002M
Open Source	Full	Schematic, Firmware	Schematic, Firmware	Full
Presnosť oscilátoru	+/-20ppm	+/-2ppm	+/-1ppm	+/-4ppm
Vysielaný výkon	-10dBm (15dBm @ 2.4GHz)	10dBm	6dBm	0dBm
Cena (September 2018)	$299	$1119	$420	$299

Obr. 2.1: Zariadenie HackRF One

2.3.1 Hardvérové vlastnosti

Zariadenie HackRF One je SDR ktorého základné parametre sú opísané v predchádzajúcej časti kapitoly.

Vzorkovacia frekvencia je nastaviteľná v rozmedzí 2 MHz až 20 MHz. Jedná sa o *half-duplex* zariadenie. Zariadenie je schopné pracovať v režime prijímania, vysie-



Obr. 2.2: Obvod HackRF One (prevzaté z [16])

lania, alebo v režime signálového zdroja. Maximálny prijímateľný výkon je -5 dBm. Pri prekročení tohoto výkonu môže dôjsť k trvalému poškodeniu zariadenia.

Maximálny vysielací výkon závisí na frekvencií, na ktorej zariadenie komunikuje. V pásme do 2150 MHz je to 5 dBm až 15 dBm. V rozmedzí 2750 MHz až 4000 MHz je to 0 dBm až 5 dBm a v pásme do 6000 MHz je to -10 dBm až 0 dBm, pričom platí, že s rastúcou frekvenciou maximálny výkon klesá. Pásmo 2150 MHz až 2750 MHz je optimálne a maximálny vysielací výkon tu dosahuje 13 dBm až 15 dBm.

HackRF One je štandardne dodávané s anténou ANT500, ktorá pracuje v pásme 75 MHz až 1 GHz. Vzhľadom na plánované využitie zariadenia s BLE bola zakúpená anténa Eightwood, ktorá pracuje mimo iného v pásmach 433 MHz, 868 MHz aj 2,4 GHz, ktoré sú využívané technológiami v tejto práci.

Hardvérové rozhranie obsahuje *High Speed* USB s konektorom USB Micro-B. Napájanie je zabezpečené cez USB. Ďalej sú na zariadení obsiahnuté 3 *SubMiniature version A* (SMA) female konektory. Jeden slúži pre pripojenie antény, zvyšné dva z týchto konektorov slúžia ako hodinový vstup a výstup pre synchronizáciu v prípade použitia viacerých zariadení. V prípade prepojeného signálu do CLKIN, tento signál nahradí signál generovaný z kryštálu. Zariadenie takisto obsahuje rozširovacie rozhranie v podobe štyroch pin líšt. Pomocou nich má zariadenie prístup k GPIO, RTC,

napájaniu, I2S, SPI, I2C, UART, hodinám, SDIO, CPLD a vysokorýchlostným dual ADC a dual DAC. Obvod použitý v HackRF One je znázornený na Obr. 2.2.

Na zariadení sa nachádzajú takisto 2 tlačidlá. Jedno z nich slúži na reset mikrokontroléru. Druhé tlačítko slúži na vyvolanie USB DFU bootloaderu. Ten umožňuje prípadné odblokovanie zariadenia s poškodeným firmware. Táto funkcionalita sa uplatňuje len pri resete a tak je možné tlačidlo nakonfigurovať tak, aby pri normálnom chode zariadenia plnilo funkciu požadovanú užívateľom.

Zariadenie môže byť použité ako USB periféria, ale môže byť taktiež naprogramované a fungovať samostatne [15].

3 Návrh systému

Táto kapitola je zameraná na predstavenie riešenia práce. V práci sú pri tvorbe softvéru využité aj nástroje tretích strán.

Vzhľadom na to kapitola mimo návrhu výsledného systému obsahuje aj popis existujúcich nástrojov, ktoré sú použité pri realizácií tejto práce. Jedná sa o nástroje *Waving-Z* a *BTLE Radio packet sniffer/scanner and sender*, ktoré sú bližšie popísané v kapitolách 3.1.2 a 3.1.3. Mimo týchto nástrojov sa takisto počíta s využitím zdrojových kódov slúžiacich pre ovládanie zariadenia *HackRF One*.

Technológia IQRF je spomedzi troch technológií, ktoré sú predmetom tejto práce, najmenej rozšírená a takisto je proprietárna. Z toho dôvodu sa v prípade protokolu IQRF nepodarilo nájsť podobný nástroj ako *Waving-Z*, či *BTLE Radio packet sniffer/scanner and sender*. Implementácia IQRF vzhľadom na určité podobnosti protokolov, ako je napríklad rovnaká použitá modulácia, môže byť založená na týchto nástrojoch. Aby bolo možné túto technologiu do systému implementovať, je o nej potrebné zistiť viac informácií. Dodatočnej analýze IQRF komunikácií sa v tejto kapitole venuje samostatná sekcia.

3.1 Použité nástroje

3.1.1 Knižnica libhackrf

Všetky softvérové aj hardvérové materiály súvisiace so zariadením HackRF sú voľne dostupné v rámci repozitára na portáli GitHub [16]. Tu sú okrem základného firmvéru dostupné aj všetky materiály potrebné pre zhotovenie vlastného HackRF zariadenia po hardvérovej stránke. Takisto sú tu dostupné zdrojové kódy obslužného softvéru. Zdrojové kódy pre HackRF sú napísané v jazyku C a na ich vývoji sa podieľali Jared Boone, Benjamin Vernoux a Michael Ossmann.

Pri realizácií výsledného systému sú použité práve zdrojové kódy obslužného softvéru. Ich základ tvorí hlavičkový súbor `hackrf.h`. V súbore sú deklarované základné štruktúry, ktoré sú programom využívané. Ďalej sú tu deklarované všetky funkcie, ktoré zabezpečujú komunikáciu s firmware zariadenia a zároveň funkcie obslužného systému. Tie slúžia napríklad pre získanie informácií o zariadení, pre nastavovanie parametrov zariadenia, alebo pre odosielanie, či príjem dát.

Zároveň sú v tomto repozitári dostupné zdrojové kódy jednotlivých nástrojov, ktoré sú štandardne spúšťané užívateľmi z príkazového riadku. Pre realizáciu tejto práce sú zaujímavé hlavne 2 súbory. Prvým z nich je `hackrf_info.c`, ktorý slúži pre získanie zoznamu pripojených HackRF zariadení a získanie základných informácií o týchto zariadeniach, ako je napríklad výrobné číslo zariadenia, či verzia

nainštalovaného firmvéru. Druhým zaujímavým súborom je `hackrf_transfer.c`, ktorý využíva funkcie z knižnice `libhackrf` pre nastavenie parametrov komunikácie a následné odosielanie alebo príjem dát.

3.1.2 Waving-Z

Nástroj *Waving-Z* je nástroj slúžiaci pre moduláciu a demoduláciu ITU G.9959 rámcov, ktoré sú využívané u technológie Z-Wave. *Waving-Z* je napísaný v jazyku C++ a jeho tvorcom je Mirko Maischberger. Repozitár obsahujúci zdrojové kódy je voľne dostupný pod licenciou `GNU General Public License` na potáli GitHub [17]. Nástroj vznikol na základe programu rtl-zwave. Ten slúži pre demoduláciu ITU G.9959 rámcov pomocou RTL-SDR a jeho repozitár je rovnako ako v prípade *Waving-Z* dostupný na portáli GitHub.

Waving-Z je navrhnutý pre použitie so zariadeniami RTL-SDR a HackRF One, mal by však byť schopný pracovať aj s inými SDR. Nástroj dokáže na vstupe pracovať so zachytenými kvadratúrnymi dátami, ktoré sú uložené v súbore, alebo mu prichádzajú priamo z SDR. Podobne je to aj v prípade modulácie a následného odosielania.

Jeho základ je tvorený hlavičkovým súborom. Ten má názov `wavingz.h` a sú v ňom okrem iného definované štruktúry, pre ukladanie jednotlivých polí rámcov. Ďalej sú tu deklarované funkcie, ktoré slúžia pre moduláciu a demoduláciu, získavanie dát z navzorkovanej informácie a skladanie, či parsovanie rámcov na jednotlivé polia. Časť funkcií je tu však takisto aj implementovaná, čo pri práci s týmto nástrojom môže spôsobiť problémy. Zvyšné funkcie sú implementované v ďalšom súbore `wavingz.cpp`.

Samotná procedúra príjmu a demodulácie je implementovaná v súbore `wave-in.cpp`, ktorý využíva funkcie implementované v `wavingz.h` a `wavingz.cpp`. Implementáciu modulácie a odosielania je možné nájsť v súbore `wave-out.cpp`.

3.1.3 BTLE Radio packet sniffer/scanner and sender

BTLE Radio packet sniffer/scanner and sender je nástroj slúžiaci pre príjem a odosielanie paketov BLE verzie 4.0. Nástroj vytvoril Xianjun Jiao a je voľne dostupný pod licenciou `GNU General Public License` repozitára na portáli GitHub [18]. Primárne je tento nástroj určený pre použitie so zariadením HackRF One a zariadením BladeRF. Tento nástroj je napísaný v jazyku C.

Nástroj dokáže za pomoci využitia funkcií obslužného softvéru HackRF prijímať a odosielať dáta, ktoré sú uložené v kvadratúrnej podobe. Tieto dáta moduluje, respektíve demoduluje a ďalej spracováva až do podoby čitateľných paketov. Nástroj

je primárne určený na počúvanie jedného kanálu, na ktorý sa užívateľ naladí. Použiť sa dá teda bez problémov pre príjem paketov na *advertising* kanáloch.

Podľa špecifikácie je schopný na *advertising* kanáloch zachytávať *ADV_CONNECT_REQ* pakety a spracovať informácie v nich obsiahnuté. Tieto informácie obsahujú okrem iného počiatočný kanál a hodnotu skoku, ktorá sa používa v priebehu celého dátového spojenia. Pomocou týchto informácií je teda nástroj schopný naladiť sa na dátový prenos a skákať po jednotlivých kanáloch spolu s prenosom. Experimentálne sa mi však funkčnosť tejto možnosti zatiaľ overiť nepodarilo, nakoľko sa spojenie vždy po maximálne dvoch prenesených paketoch prerušilo. V prípade odosielania na rôznych kanáloch je však nástroj limitovaný len rýchlosťou prepínania kanálov u použitého hardvéru.

Zdrojový kód je implementovaný odlišným spôsobom ako u predošlých dvoch nástrojov. Všetky štruktúry a funkcie potrebné pre príjem dát sú obsiahnuté v súbore `btle_rx.c`. V prípade odosielania je to zasa súbor `btle_tx.c`.

3.2 Analýza IQRF komunikácie

V prípade protokolu IQRF nebol nájdený žiadny funkčný nástroj, ktorý by dokázal prijímať, spracovávať a odosielať pakety.

Dokumentácia k tomuto protokolu je takisto značne obmedzená a neobsahuje detailnejšie informácie o štruktúre paketov.

Za účelom lepšieho pochopenia štruktúry IQRF paketov bol pre SDR vytvorený systém schopný prijímať IQRF dáta a následne bola nad týmito dátami vykonaná analýza.

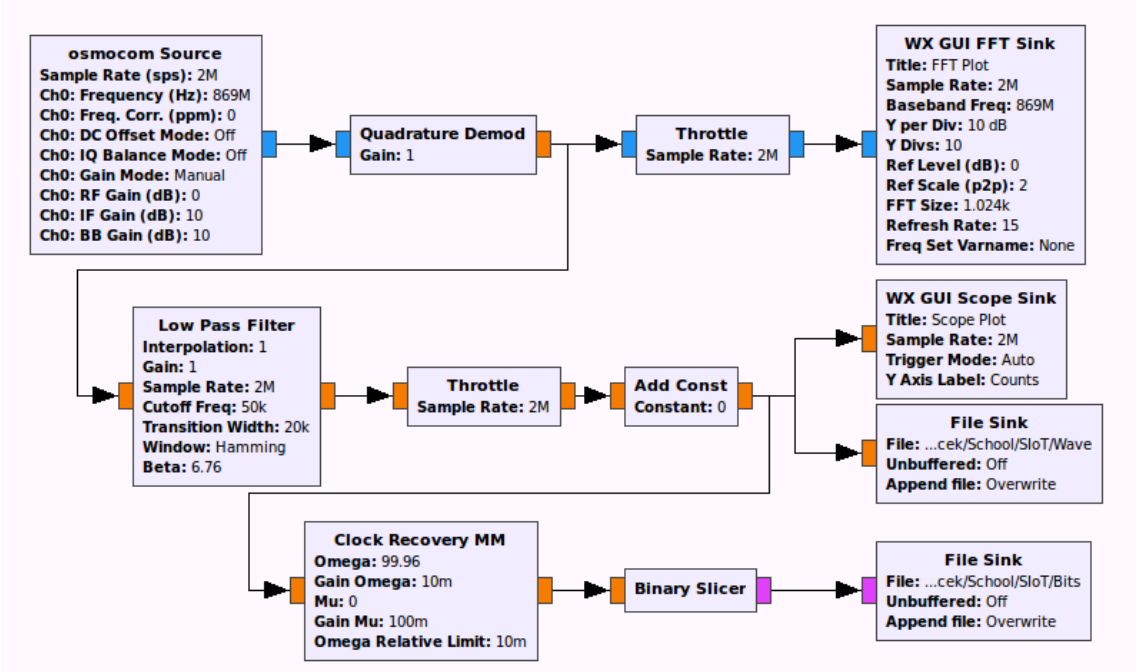
Sledovaná je tu komunikácia medzi zariadeniami DK-EVAL-04A a CK-USB-04A, ktoré obsahujú RF moduly TR-72D . Zariadenie CK-USB-04A je pomocou USB pripojené k počítaču, z ktorého sú pomocou nástroja IQRF IDE naprogramované postupne oba používané moduly pre jednotlivé pozorované scenáre.

3.2.1 Príjem komunikácie cez GNU Radio

Pre príjem signálov IQRF je vytvorená schéma prijímacej aparatúry v nástroji *GNU Radio*. Prijímacia aparatúra sa skladá výhradne z originálnych *GNU Radio* blokov. Schéma použitých blokov je znázornená na obrázku Obr. 3.1.

Blok *osmocom Source* reprezentuje samotné SDR, teda v tomto prípade zariadenie HackRF One. Pomocou bloku je možné nastaviť parametre na ktorých SDR prijíma signály.

Ďalej nasleduje demodulácia signálu pomocou bloku *Quadrature Demod* a filtrovanie dolnou prepusťou, ktorá je reprezentovaná blokom *Low Pass Filter*.



Obr. 3.1: Schéma prijímača IQRF dát v GNU Radio.

Blok *Throttle* slúži nástroju *GNU Radio* pre limitovanie výstupného dátového toku na nastavenú hodnotu, čím sa zmiernia nároky na potrebný výpočtový výkon procesoru. Štandardne totiž nástroj *GNU Radio* v snahe pracovať v reálnom čase množstvo spracovávaných dát nijako nelimituje a jediným obmedzením je potom použitý hardvér.

Pomocou ďalšieho bloku *Add Const* je kompenzované prípadné posunutie nulovej hladiny signálu.

Pre monitorovanie prijímaných dát je po demodulácií vykresľované spektrum pomocou bloku *WX GUI FFT Sink* a po vyfiltrovaní a kompenzácií odchýlky je takisto vykresľovaný aj prijímaný signál pomocou bloku *WX GUI Scope Sink*. Tento signál je zároveň pomocou bloku *File Sink* ukladaný do súboru a ďalej vykreslený a analyzovaný pomocou nástroja *Audacity*, čo je popísané v časti 3.2.2.

Signál je ďalej takisto prevedený do binárnej podoby. Najskôr je pomocou bloku *Clock Recovery MM* signál prevedený na požadovanú symbolovú rýchlosť. Blokom *Binary Slicer* sú následne zo signálu generované jednotlivé bity a tie sú potom pomocou bloku *File Sink* uložené do súboru pre ďalšie spracovanie inými nástrojmi.

3.2.2 Analýza prijatých IQRF dát

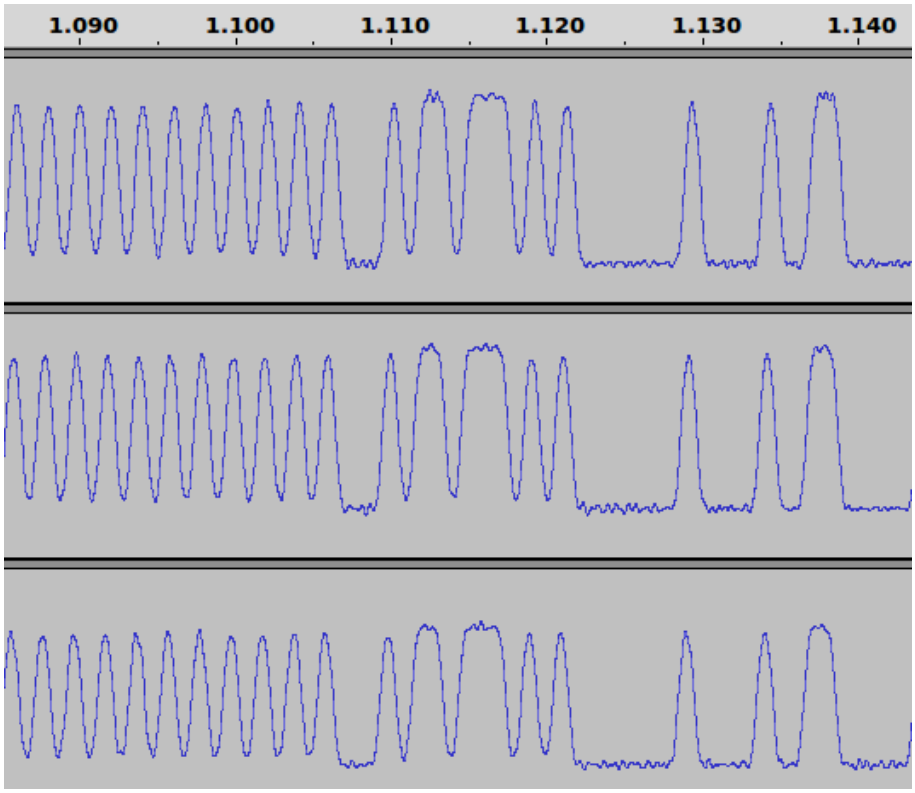
S cieľom lepšie porozumieť štruktúre IQRF paketov sú skúmané IQMESH aj Peer-to-peer pakety. U IQMESH sa jedná o pakety pre zber informácií z teplotných sen-

zorov. U Peer-to-peer sa jedná o pakety nesúce systémovú komunikáciu, napríklad pri pripájaní nových zariadení.

U týchto paketov je možné v dátach detegovať preambulu a synchronizáciu, samotné dáta však už sú šifrované a tak z nich nie je možné bez hlbšej znalosti protokolu získať užitočné informácie. Vzhľadom na to sú následne tak isto sledované aj pakety v móde *RF Programming* (RFPGM), kde dáta nie sú šifrované.

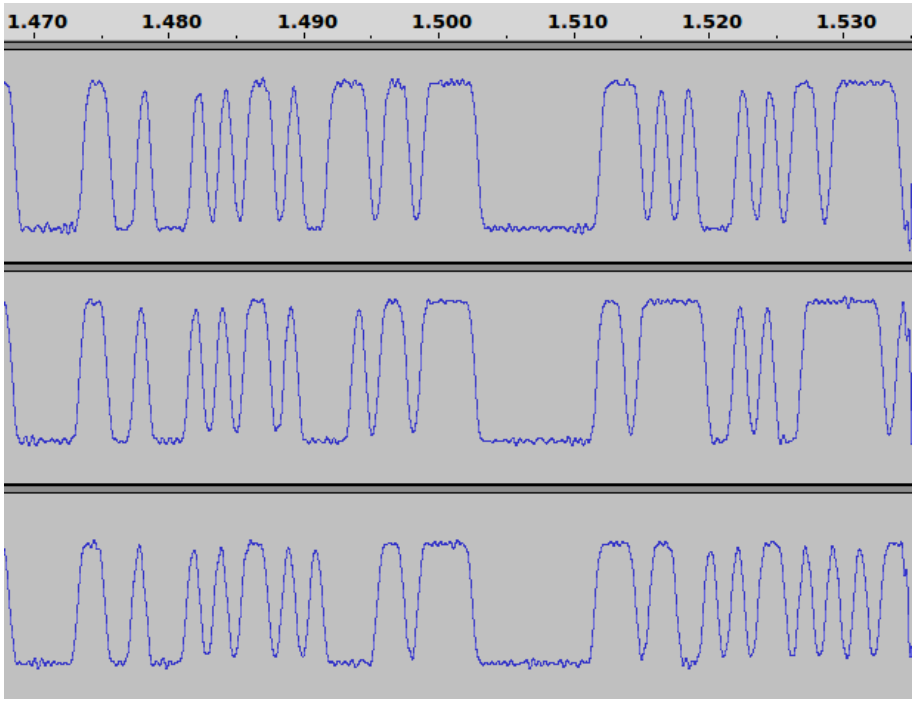
Na začiatku sú zaznamenané tri rovnaké sekvencie príkazov. V prípade RFPGM ide o príkaz na nastavenie prístupového hesla na hodnotu 1234 a o príkaz na zrušenie tohoto hesla. Tieto tri zachytené signály sú následne pomocou nástroja *Audacity* orezané a zarovnané pre účel porovnania častí nesúcich informáciu.

V tomto prípade sa už okrem preambule a synchronizácie zhodujú aj ďalšie časti paketov. Napriek tomu je u každého z paketov rozličné CRC a koniec *payload* dát najnižšej vrstvy. Z dostupnej dokumentácie sa však nepodarilo s určitosťou zistiť, čo tieto rozdiely spôsobuje. Náhľad na začiatok a koniec troch zaznamenaných paketov možno vidieť na obrázkoch Obr. 3.2 a Obr. 3.3



Obr. 3.2: Časť preambule a synchronizačného bloku paketu pre nastavenie hesla u troch zaznamenaných paketov.

Tento prijímací obvod by sa teda dal využiť u jednoduchých detekcií, na základe dĺžky jednotlivých blokov informácií a rozostupov medzi nimi. Bez hlbšej znalosti



Obr. 3.3: Časť *payload* dát a CRC bloku paketu pre nastavenie hesla s viditeľnými rozdielmi u troch zaznamenaných paketov.

protokolu však nie je však použiteľný na účel získavania a odosielania dát. Ďalšie možnosti práce s protokolom IQRF v rámci tejto práce sú teda obmedzené až do zverejnenia viacerých informácií od IQRF Aliancie.

3.3 Návrh API

S ohľadom na použitie nástrojov tretích strán, ktoré sú implementované v jazykoch C a C++ je výsledný systém implementovaný v týchto dvoch jazykoch a API je navrhnuté v jazyku C. API v jazyku C umožní väčšiu flexibilitu vzhľadom na to, že užívateľovi dovoľuje tento framework používať tak v jazyku C, ako aj v jazyku C++. Ako pracovný názov bol pre framework zvolený názov *IoT Traffic Generating Tool*, z čoho vyplýva skratka ITGT využitá pre pomenovanie jednotlivých štruktúr a funkcií. Táto časť práce obsahuje stručný popis vybraných štruktúr a funkcií.

3.3.1 Prehľad štruktúr

`itgt_device_list`

Štruktúra `itgt_device_list` obsahuje lineárne viazaný zoznam zariadení. Ten obsahuje obsahuje 33 znakov dlhý reťazec pre uloženie sériového čísla zariadenia. Ďalej

je tu použitá položka `flags`, ktorá obsahuje príznaky slúžiace pre aktualizovanie zoznamu, či výber voľných zariadení. Nakoniec je tu ukazovateľ na ďalšiu položku zoznamu. Užívateľ so štruktúrou pracuje pomocou užívateľských funkcií, ktoré sú popísané v sekcií 3.3.2

```
typedef struct devlist_device_t
{
    char serial_number[33];
    itgt_flags_t flags;
    struct devlist_device_t *next;

} *itgt_devlist_device_t;
```

itgt_handle

Základná dátová štruktúra navrhnutého API, `itgt_handle`, slúži pre komunikáciu jednotlivých funkcií nástroja a odovzdávanie dát použitým nástrojom tretích strán.

Obsahuje informáciu o použitom zariadení, jeho nastavených parametroch a ukazovateľ na štruktúru využívanú knižnicou `libhackrf`. Ďalej táto štruktúra nesie dáta a metadáta posledného prečítaného paketu a informáciu o používanom komunikačnom protokole. Štruktúra nesie aj informáciu o súbore pre zápis, či čítanie v prípade, že s ním užívateľ chce pracovať. Nakoniec je tu obsiahnutá štruktúra slúžiaca pre filtrovanie prijatých správ. Tá obsahuje funkciu definovanú užívateľom a užívateľské dáta, ktoré jej užívateľ chce dať na vstup.

K jednotlivým častiam tejto štruktúry sa pristupuje rôznym spôsobom. Obsah tejto štruktúry je znázornený v sekcií 3.3.1.

```
typedef struct handle_t
{
    itgt_device_t *device;
    itgt_packet_t packet;
    protocol_t protocol;
    itgt_savefile_t *file;
    itgt_filter_t *clb;
} itgt_handle_t;
```

itgt_device

`itgt_device` nesie informácie o používanom zariadení. Konkrétne ide o jeho sériové číslo, pomocou ktorého je možné zariadenia rozlíšiť, zoznam nastavených paramet-

rov a zoznam príznakov, ktoré symbolizujú stav v ktorom sa zariadenie nachádza. Jednotlivé položky tejto štruktúry sa štandardne nastavujú pomocou užívateľských funkcií, ktoré sú popísané v sekcií 3.3.1.

`itgt_packet`

`itgt_packet` obsahuje *buffer* v ktorom je posledný prečítaný paket. Paket je tu uložený spolu s metadátami, ktoré popisujú čas prijatia, veľkosť paketu a jeho typ. Dáta sa sem ukladajú štandardne automaticky, pomocou funkcie pre zápis, ktorá je vnorená vo funkciách pre zachytávanie paketov. Užívateľ má možnosť tieto dáta upravovať aj priamo a tak si obsah *bufferu* a metadát prispôsobiť pred odoslaním paketov.

```
typedef struct packet_t
{
    struct timeval time;
    uint8_t length;
    uint8_t packet_type;
    uint8_t channel;
    btle_recv_status_t receiver_status;
    char data[64];
} itgt_packet_t;
```

`itgt_savefile`

`itgt_savefile` obsahuje informácie o súbore s ktorým systém pracuje. Konkrétne je v tejto štruktúre informácia o tom, či sa súbor používa, ďalej je tu deskriptor použitého súboru a reťazec nesúci názov súboru. Užívateľ s touto štruktúrou priamo nemanipuluje.

`itgt_filter`

`itgt_filter` nesie informácie o filtri prijímaných paketov. Súčasťou tejto štruktúry je ukazovateľ na užívateľom definovanú filtrovaciu funkciu a položka `user_data` nesúca vstupné parametre pre túto funkciu. Užívateľ k tejto štruktúre pristupuje priamo.

```
typedef struct filter_t
{
    itgt_filter_clb_t function;
    void* user_data;
} itgt_filter_t;
```

3.3.2 Prehľad funkcií a obsluhy

Prvá vec, ktorú musí užívateľ vykonať, ak chce s týmto frameworkom pracovať, je získať prehľad o pripojených zariadeniach. To možno docieliť pomocou funkcie `itgt_deviceListLookup()`, ktorá k tomu využíva funkcie z knižnice `libhackrf`. Tie zároveň umožňujú získať informácie o zariadení, ktoré sú neskôr využité pre rozlíšenie jednotlivých zariadení. Zoznam zariadení je následne uložený do štruktúry `itgt_device_list`.

```
itgt_error_t itgt_deviceListLookup(itgt_device_list_t *device_list);
```

Užívateľ má možnosť vytvorený zoznam zariadení priebežne aktualizovať pomocou funkcie `itgt_deviceListUpdate()`.

```
itgt_error_t itgt_deviceListUpdate(itgt_device_list_t *device_list,
    hackrf_device_list_t *list);
```

Pre korektné ukončenie programu je k dispozícií funkcia `itgt_deviceListFree()`, ktorá uvoľní pamäť alokovanú pre jednotlivé zariadenia v zozname.

```
itgt_error_t itgt_deviceListFree(itgt_device_list_t *device_list);
```

Užívateľ do parametrov jednotlivých zariadení zasahuje prostredníctvom štruktúry *handle*. Pre počiatočné nastavenie handle a inicializáciu parametrov zariadenia slúži funkcia `itgt_create()`. Tá má tri povinné parametere. Prvým z nich je samotný *handle*, ktorý sa tu inicializuje. Druhým je `device_list`, z ktorého je priradené zariadenie. Tretím parametrom je protokol, s ktorým chce užívateľ pracovať. Okrem týchto povinných parametrov má aj voliteľné parametre. Jedným z nich je parameter `serial_number`, ktorý obsahuje sériové číslo zariadenia v `device_list`, ktoré chce užívateľ použiť. V prípade, že ho užívateľ nezadá, je automaticky vybrané prvé voľné zariadenie v zozname. Takisto sa pred samotným výberom kontroluje, či je zariadenie stále pripojené. Ostatné parametre `itgt_create()` môže užívateľ použiť v prípade, že chce inicializovať parametre prenosu na želané hodnoty.

```
itgt_error_t itgt_create(itgt_handle_t *handle, itgt_device_list_t *L,
    char* serial_number, protocol_t protocol, int channel, int BW, int
    gain);
```

Užívateľ môže parametre meniť aj dodatočne, v prípade, že zariadenie práve nevysiela alebo neprijíma. Za týmto účelom má užívateľ k dispozícií funkciu `itgt_setParams()`. V prípade frekvencie som sa rozhodol pre zavedenie vyššej ab-

strakcie a užívateľ miesto presnej frekvencie štandardne zadáva číslo kanálu, na základe neho sa následne v kombinácií s informáciou o požadovanom protokole vyberie príslušná frekvencia.

```
\vspace{3mm}
itgt_error_t itgt_setParams(itgt_handle_t *handle, int channel, int BW,
    int gain);
```

Pokiaľ však užívateľ chce nastaviť presnú frekvenciu, môže to docieliť použitím funkcie `itgt_setFreq()`.

```
itgt_error_t itgt_setFreq(itgt_handle_t *handle, int64_t freq);
```

Po vytvorení a nastavení *handle* môže užívateľ realizovať samotné prijímanie, či vysielanie. Prijímanie je realizovateľné funkciou `itgt_receive()`, ktorá načíta ďalší paket a uloží ho do *bufferu*. Metadáta prijatých paketov, ako dĺžka a čas, sú ukladané do štruktúry `itgt_packet` spolu s dátami. Pre vysielanie slúži funkcia `itgt_send()`, ktorá odošle aktuálny obsah *bufferu* v štruktúre `itgt_packet`. Veľkosť odoslaných dát je špecifikovaná na základe metadát prislúchajúcich danému paketu.

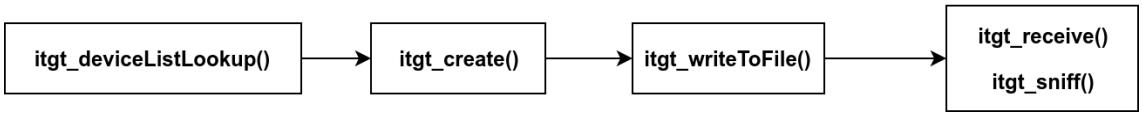
```
itgt_error_t itgt_receive(itgt_handle_t *handle);

itgt_error_t itgt_send(itgt_handle_t *handle);
```

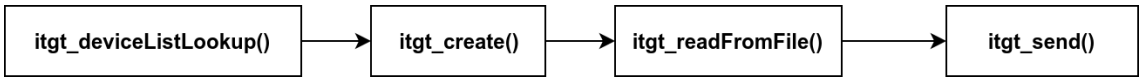
V prípade, že si užívateľ želá miesto prijatia jedného paketu sledovať komunikáciu dlhodobo, môže využiť funkciu `itgt_sniff()`. Tá skenuje zadanú frekvenciu a prijíma pakety až do prerušenia programu. Ak zároveň užívateľ využíva zápis do súboru, tak môže byť celá komunikácia zaznamenaná a ďalej spracovaná, alebo reprodukovaná.

```
itgt_error_t itgt_sniff(itgt_handle_t *handle);
```

Framework umožňuje aj ukladanie prečítaných dát do súboru pomocou funkcie `itgt_writeToFile()`. Funkcia nastaví príznak používania súboru pre ukladanie dát. Následne je pomocou ďalších funkcií do súboru najskôr zapísaná hlavička, ktorá obsahuje verziu nástroja a používaný protokol a potom dáta a metadáta jednotlivých paketov. Pokiaľ nie je `file` prázdny súbor, je tento súbor prepísaný. Pre čítanie dát zo súborov slúži funkcia `itgt_readFromFile()`. Tá dokáže jednotlivé pakety parsovať vďaka informácií o dĺžke dát, ktorá je súčasťou metadát každého uloženého paketu.



Obr. 3.4: Schéma pre príjem dát do súboru



Obr. 3.5: Schéma pre odosielanie dát zo súboru

Funkcie `itgt_writeToFile()` a `itgt_readFromFile()` neslúžia len pre záznam a odosielanie uložených dát. Pomocou funkcie `itgt_readFromFile()` môže užívateľ načítať paket zo súboru do *bufferu* a následne ho ďalej upravovať. Upravené, alebo novo vytvorené pakety potom užívateľ môže pomocou funkcie `itgt_writeToFile()` ukladať do súboru. Schéma znázorňujúca použitie funkcií potrebných pre príjem a odosielanie dát s využitím súboru je zobrazená na obrázkoch Obr. 3.4 a Obr. 3.5.

```
itgt_error_t itgt_readFromFile(itgt_handle_t *handle, const char
    *filename);

itgt_error_t itgt_writeToFile(itgt_handle_t *handle, const char
    *filename);
```

Pomocou funkcií `itgt_readPacket()` a `itgt_writePacket()` je vykonávané čítanie, respektíve zápis jednotlivých paketov po tom, čo užívateľ funkciami `itgt_readFromFile()` a `itgt_writeToFile()` nastaví využívanie súboru pre čítanie, respektíve zápis.

```
int itgt_readPacket(itgt_handle_t *handle);

itgt_error_t itgt_writePacket(itgt_handle_t *handle);
```

V prípade úpravy zaznamenaných dát môže užívateľ pomocou funkcie `itgt_copyFileHeader()` skopírovať hlavičku súboru s pôvodnými dátami do súboru s novými dátami, pred tým ako budú zapisované jednotlivé pakety.

```
itgt_error_t itgt_copyFileHeader(itgt_handle_t *input, itgt_handle_t
    *output);
```

Framework takisto umožňuje užívateľovi prijímané pakety filtrovať. To je rea-

lizované pomocou *callback* funkcie `itgt_filter_clbk()`. Tá obsahuje na vstupe ukazovateľ na filtrovaciu funkciu, ktorú definuje užívateľ. Pokiaľ užívateľ funkciu nedefinuje, ostáva ukazovateľ prázdny a program filtrovanie ignoruje.

```
bool itgt_filter_clbk(char *string, void *user_data, itgt_filter_clb_t
    functionPtr);
```

4 Implementácia systému

Táto kapitola nadväzuje na návrh API z predchádzajúcej kapitoly. Kapitola je rozdelená na jednotlivé bloky podľa ktorých je systém postupne implementovaný a opisuje niektoré dôležité detaily z implementácie. Zdrojové kódy vytvoreného systému sú dostupné v repozitári na portáli github[1].

4.1 Definícia štruktúr

Štruktúry s ktorými systém pracuje sú definované na základe navrhnutého API. Všetky štruktúry sú implementované v rámci samostatného hlavičkového súboru `structures.h`. Okrem základných štruktúr sú tu definované aj ďalšie pomocné štruktúry a výčtové typy. Základné štruktúry pre prácu s nástrojom sú štruktúry `itgt_device_list` a `itgt_handle`, ktoré sú spolu s ďalšími štruktúrami bližšie popísané v sekcií 3.3.1.

4.2 Implementácia BLE prenosov

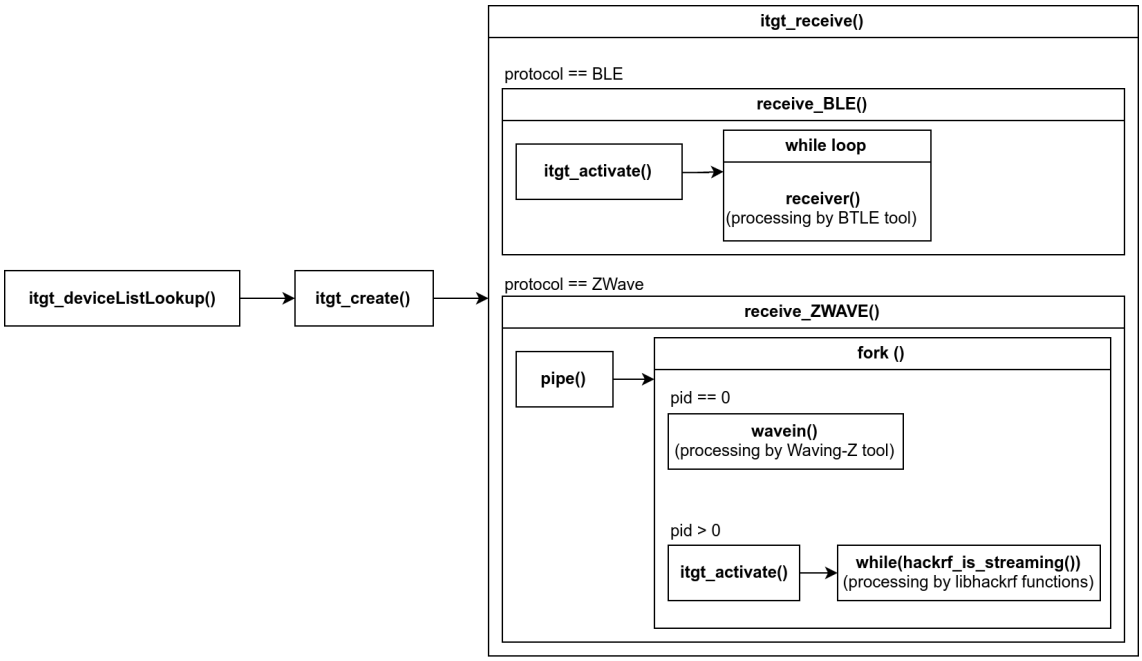
Pri prijímaní a odosielaní BLE dát je využitý nástroj *BTLE Radio packet sniffer/scanner and sender*. Samotný nástroj pracuje priamo s knižnicou `libhackrf`. Pomocou callback funkcie postupne načítava prichádzajúce vzorky, ktoré znak po znaku ukladá do cyklického *bufferu*. Z týchto uložených vzorkov sú následne vytvárané jednotlivé symboly, ktoré sú ďalej spracovávané.

Na začiatku každého paketu je informácia o hodnote prístupovej adresy, ktorú využíva sledované BLE zariadenie. V bajtoch nasledujúcich po nej možno určiť o aký typ paketu sa jedná. Na základe toho dokáže nástroj určiť aký typ paketu prijal. Podľa typu prijatého paketu potom na štandardný výstup vytlačí rozparsované dáta daného druhu paketu. Zároveň sú v prípade paketov, ktoré signalizujú zmenu niektorých parametrov prenosu, tieto hodnoty získané a príslušné parametre sú upravené.

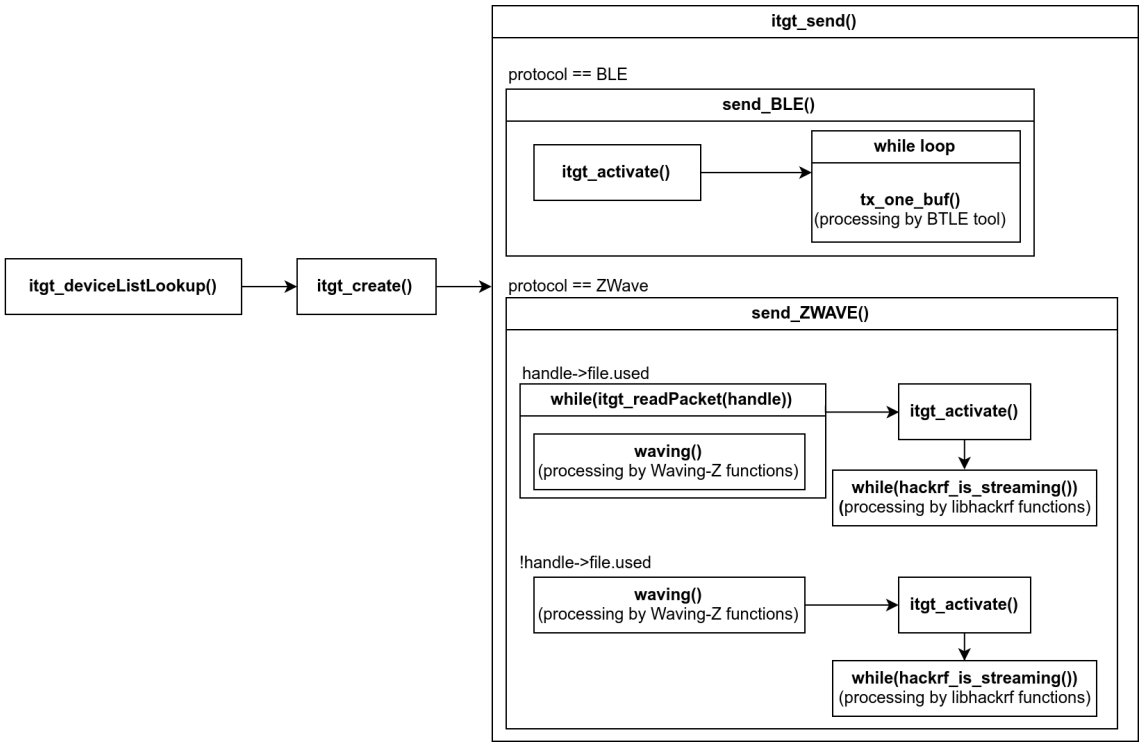
Pre využitie v tejto práci je potrebné upraviť výstup na požadovanú podobu a zabezpečiť prenos jednotlivých parametrov medzi funkciami nástroja *BTLE Radio packet sniffer/scanner and sender* a zvyškom programu.

Štruktúra znázorňujúca prepojenie funkcií pre príjem BLE a Z-Wave dát je zobrazená na Obr. 4.1. Štruktúra znázorňujúca prepojenie funkcií pre odosielanie dát je zobrazená na Obr. 4.2 . V oboch prípadoch sa dáta medzi jednotlivými funkciami prenášajú pomocou štruktúry `handle`.

[1] https://github.com/iljacek/DP-frameworkIoT



Obr. 4.1: Štruktúra znázorňujúca prepojenie funkcií pre príjem dát



Obr. 4.2: Štruktúra znázorňujúca prepojenie funkcií pre odosielanie dát

4.2.1 Prepojenie nástrojov

Prepojenie je v prípade BLE realizované v porovnaní s nástrojom *Waving-Z* jednoducho. Hlavné telo použitého nástroja je implementované v rámci funkcie `send_BLE()` pre odosielanie, respektíve `receive_BLE()` pre príjem.

Vstupným parametrom týchto funkcií je ukazovateľ na štruktúru *handle*, ktorá obsahuje všetky dôležité údaje o prenose. *Handle* je následne odovzdávaný prostredníctvom upraveného kódu aj do jednotlivých funkcií nástroja *BTLE Radio packet sniffer/scanner and sender*.

```
/* Process of calling BLTE tool functions receiver and
   receiver_controller in while loop. Receiver will get next packet to
   buffer and receiver_controller handles the hopping. All data are
   passed through handle structure */
itgt_error_t receive_BLE(itgt_handle_t *handle)
{
  ...
  while(do_exit == false) { //while hackrf is streaming
    ...
    if (run_flag) {
      receiver(rxp,..., handle);

      if (hop_flag){
        if ( receiver_controller(rf_dev, ..., handle) != 0 )
      goto program_quit;
      }
    ...
}
```

4.3 Implementácia Z-Wave prenosov

Na rozdiel od nástroja pracujúceho s techonológiou BLE, nástroj *Waving-Z* nepracuje priamo s knižnicou `libhackrf`. Nástroj *Waving-Z* pracuje s dátami zo štandardného vstupu.

V prípade tejto práce sa predpokladá, že na štandardnom vstupe bude čítať dáta zo zariadenia HackRF One. A tak je potrebné zabezpečiť prepojenie týchto dvoch nástrojov, ktoré sú bližšie popísané v sekciách 3.1.1 a 3.1.2.

Pomocou využitia knižnice `libhackrf` je teda vykonávaný samotný príjem, respektíve vysielanie dát. Spracovanie dát prebieha pomocou funkcií nástroja *Waving-Z*.

Štruktúry znázorňujúce prepojenie funkcií pre príjem a odosielanie dát sú na obrázkoch Obr. 4.1 a Obr. 4.2.

4.3.1 Prepojenie nástrojov

Na začiatku funkcií pre príjem Z-Wave dát je vytvorená rúra a pomocou funkcie `fork()` je vytvorený nový proces. V prípade príjmu potom otcovský proces vykonáva samotný príjem dát pomocou funkcií z knižnice `libhackrf` a potomok vykonáva spracovanie dát pomocou funkcií nástroja *Waving-Z*.

```
/* Create the pipe. */
if (pipe (mypipe)) {
  fprintf (stderr, "Pipe failed.\n");
  return EXIT_FAILURE;
}

/* Create the child process. */
pid = fork ();
```

V prípade príjmu sú dáta podobne ako pri BLE prijímané callback funkciou, ktorá jednotlivé vzorky ukladá tentokrát nie do bufferu, ale do rúry. Následne sú na druhej strane *pipeline* spracované ďalšou callback funkciou.

Nástroj *Waving-Z* podobne ako nástroj využitý u BLE sa štandardne používa volaním z príkazového riadku. Tento nástroj po prijatí dát a ich spracovaní vytlačí rozparsované dáta na štandardný výstup.

V prípade využitia jeho jednotlivých funkcií v rámci tejto práce je však potrebné zabezpečiť ukladanie dát do *bufferu* a takisto zabezpečiť prístup pre ďalšiu funkcionalitu ako je ukladanie, či filtrovanie dát.

Za týmto účelom je do funkcií nástroja *Waving-Z* pridaná položka `user_data` typu void*, ktorá umožňuje využitie vlastných štruktúr pri práci s týmto nástrojom. Vzhľadom na to, že tento nástroj sa štandardne využíva volaním z príkazového riadku, táto úprava nie je navrhnutá do centrálneho repozitára na portáli Github, ale je realizovaná len lokálne. Prostredníctvom položky `user_data` je v prípade tejto práce prenášaná štruktúra `itgt_handle`.

```
/* process of reading the raw data streamed from hackrf and passing it to
   Waving-Z for decoding */
if (pid == (pid_t) 0)
{
  /* This is the child process. Close other end first. */
  close (mypipe[1]);
```

```
  fprintf(stderr, "fclose(fd) %d\n", __LINE__);

  fd = fdopen (mypipe[0], "r");

  struct process_wavingz {
      void operator()(uint8_t* begin, uint8_t*end, void *user_data)
      {
        wavingz::zwave_print(std::cout, begin, end,
            (itgt_handle_t*)handle) << std::endl;
        }
  } wave_callback;

  wavingz::demod::demod_nrz wavein(sample_rate, wave_callback,
      (itgt_handle_t*)handle);
  ...
}
```

V prípade odosielania dát je využité sekvenčné spracovanie. Dáta sú teda najskôr spracované nástrojom *Waving-Z* a spracované dáta sú uložené. Po spracovaní dát sú tieto dáta odoslané prostredníctvom funkcií knižnice `libhackrf`. Dôvodom použitia sekvenčného spracovania bola vyššia úspešnosť odosielania paketov pri testovaní.

4.4 Výber výsledného formátu spracovávaných dát

Pre efektívne a jednoduché využitie systému je dôležité, aby prijímané a odosielané dáta boli držané v jednotnom formáte.

Vzhľadom na to, že táto práca by mala byť využitá na vytváranie komunikácie pri penetračnom testovaní v rámci projektu SIoT[2], bol za formát dát zvolený formát UniRec. Tento formát je v projekte SIoT značne využívaný, jeho použitie teda povedie k lepšej interoperabilite s ďalšími nástrojmi v budúcnosti.

4.4.1 Formát UniRec

Formát UniRec slúži pre prácu s jednoduchými neštrukturovanými záznamami. Záznam je podobný klasickej štruktúre v jazyku C, avšak polia môžu byť definované aj počas behu programu. Štruktúry teda môžu byť vytvárané dynamicky. Detailnejší popis formátu je dostupný na [19].

[2]https://github.com/SecureGatewayIoT

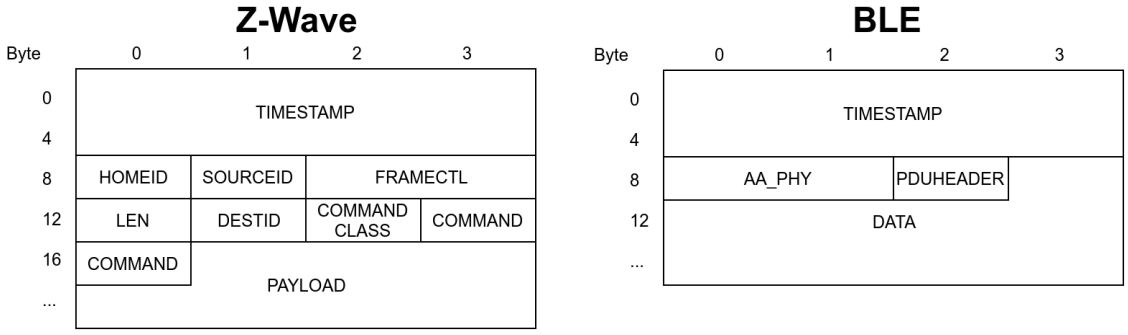
Základnou výhodou formátu je, že prístup k jednotlivým položkám je veľmi rýchly. Pre vytvorenie záznamu musí užívateľ najskôr špecifikovať množinu použitých polí a ich typov. Následne je alokovaná pamäť pre záznam a hodnoty polí sú nastavené pomocou jednoduchých makier.

4.4.2 Návrh štruktúry dát pre UniRec

Pre efektívnu prácu s dátami bolo potrebné zaručiť vhodné rozdelenie dát. Prvé pole je u všetkých štruktúr čas, kedy bol daný paket prijatý

V prípade technológie Z-Wave sa pracuje len s jedným typom paketov, všetky dáta teda využívajú jednu spoločnú štruktúru. Táto štruktúra je založená na štruktúre paketu, ktorá je popísaná v sekcií 1.2.

V prípade technológie BLE majú jednotlivé druhy paketov značne odlišnú štruktúru. Na začiatku každého paketu je prístupová adresa, ktorá má dĺžku 4 bajty. Po nej nasleduje hlavička paketu o dĺžke 1 bajtu. Hlavička má však odlišnú štruktúru u *Advertising* paketov a u *LL* paketov. Vzhľadom na to, že typ paketu je ukladaný v rámci metadát, tak je možné použiť jednotné pole pre oba druhy paketov. Zvyšné dáta sú v *raw* podobe pridané ako posledné pole. Formát štruktúry dát je zobrazený na Obr. 4.3.



Obr. 4.3: Použitý formát UniRec dát pre Z-Wave a pre BLE

V rámci tejto práce z dôvodu umožnenia viacerých simultánnych procesov je záznam ukladaný do preddefinovaného *buffera*, ktorý je súčasťou štruktúry *handle*. Popri samotných UniRec dátach je tu ukladaná aj informácia o dĺžke dát, druhu paketu, kanáli na ktorom boli dáta prijaté a hodnota `crc_init`, ktorá slúži pre výpočet kontrolného CRC.

4.5 Zápis a čítanie dát zo súboru

Aktuálny paket je vždy uchovávaný v bufferi. V prípade že užívateľ chce pracovať s viacerými paketmi, je mu umožnené ukladať prijímané pakety do súboru, respektíve z neho pakety postupne čítať.

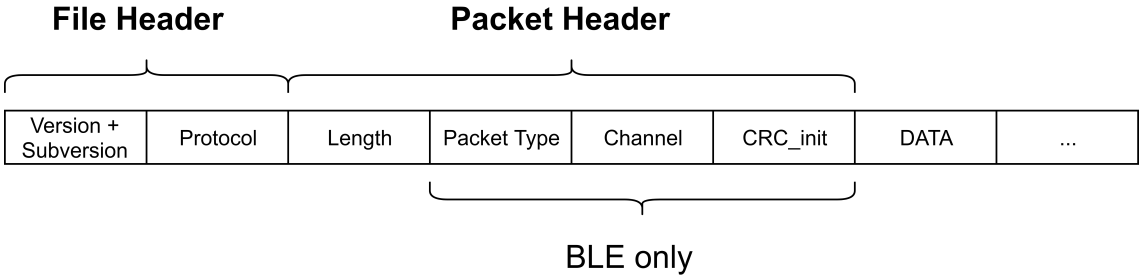
Aktuálne sa využíva vlastný formát súboru, no do budúcnosti je zvažované použitie štandardného UniRec formátu dát v súbore, aby bolo možné priame prepojenie zaznamenaných dát s ďalšími nástrojmi.

4.5.1 Formát ukladaných dát

Súbor obsahuje hlavičku súboru, ktorá je nasledovaná jednotlivými paketmi. Tie obsahujú popri samotných dátach aj hlavičku, ktorá obsahuje metadáta jednotlivých paketov.

V hlavičke súboru je obsiahnutá verzia a subverzia nástroja a pri čítaní súboru sa kontroluje kompatibilita s používanou verziou. Ďalej obsahuje hlavička súboru informáciu o protokole, ktorého sa záznam týka.

V hlavičke paketu je potom obsiahnutá v prípade protokolu Z-Wave dĺžka dát, ktoré sú vo formáte UniRec. V prípade BLE je súčasťou hlavičky aj informácia o type paketu, hodnota pre inicializáciu CRC a informácia o kanáli, na ktorom bol paket prijatý. Formát dát v súbore je zobrazený na Obr. 4.4.



Obr. 4.4: Formát dát zapísaných v súbore

4.5.2 Prepojenie s implementovaným systémom

Čítanie využíva dve funkcie. Sú to funkcie `itgt_readFromFile()`a `itgt_readPacket()`. Zápis využíva ekvivalenté funkcie `itgt_writeToFile()` a `itgt_writePacket()`. Okrem týchto dvoch funkcií zápis využíva aj tretiu funkciu `itgt_writeHeader()`, ktorá je súčasťou funkcií pre zápis dát. Samostatná funkcia pre zapisovanie hlavičky je implementovaná pre prípad, že by sa užívateľ rozhodol zmeniť protokol medzi tým, ako vytvorí súbor pre zápis a ako začne zachytávať

pakety. Prvá z funkcií slúži v oboch prípadoch pre vytvorenie súboru s príslušným menom a nastavenie príznaku, ktorý informuje nástroj o používaní súboru.

Funkcia `itgt_writeHeader()` v prípade zápisu dát vytvorí počas aktivácie zariadenia hlavičku súboru. V prípade čítania dát túto hlavičku prečíta a na základe nej vyhodnotí zhodu s aktuálnou verziou a nastaví hodnotu protokolu v *handle* u využívaného zariadenia.

Funkcie `itgt_readPacket()` a `itgt_writePacket()` slúžia na čítanie, respektíve zápis jednotlivých paketov s ich metadátami a je realizovaná vo vnútri používaných nástrojov tretích strán. Štruktúra s informáciami o použitom súbore je do nich prenesená ako užívateľské dáta prostredníctvom štruktúry *handle*.

4.6 Implementácia ďalších užívateľských funkcií

Pre efektívnejšiu prácu s nástrojom je potrebné implementovanie ďalších funkcií. Jedná sa predovšetkým o získavanie informácií o jednotlivých poliach zachytených paketov, respektíve upravovanie jednotlivých polí týchto paketov. Ďalej je implementovaná takisto filtrovacia funkcia, pomocou ktorej je možné zachytávať len pakety požadované užívateľom.

4.6.1 Úprava dát v bufferi

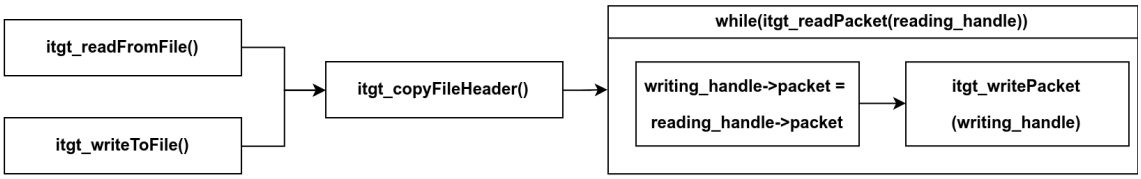
Vzhľadom na použitie formátu UniRec nie sú implementované ďalšie funkcie pre získavanie, respektíve upravovanie dát jednotlivých paketov. Predpokladá sa, že užívateľ ovláda prácu s formátom UniRec a takisto štruktúru paketov, s ktorými pracuje.

Na základe toho potom užívateľ môže pomocou vytvorenia dvoch objektov typu *handle* realizovať čítanie, úpravu a opätovné ukladanie paketov.

Užívateľ najskôr pomocou funkcie `itgt_writeToFile()` pre jeden *handle* nastaví súbor pre zápis a pomocou funkcie `itgt_readFromFile()` pre druhý *handle* nastaví súbor pre čítanie. V ďalšom kroku pomocou funkcie `itgt_copyFileHeader()` skopíruje hlavičku súboru.

Následne môže užívateľ pomocou funkcie `itgt_readPacket()` prechádzať jednotlivé pakety zo súboru. Spolu s dátami sú do *handle* nahrané aj metadáta, tie potom môže užívateľ jednoducho skopírovať do druhého handle, z ktorého sa zapisuje, alebo ich prepísať na požadované hodnoty. V prípade zmeny dát, ktoré sú obsiahnuté v položke `data`, je potrebná aj úprava parametru `len`, ktorý reprezentuje dĺžku dát. Aktuálnu dĺžku dát je možné nanovo získať použitím UniRec funkcie `ur_rec_size()`. Tým sa zaručí, že dĺžka zodpovedá dĺžke dát paketu po prípadnej úprave.

Proces kopírovania dát je zobrazený na Obr. 4.5. Pokiaľ chce užívateľ dáta zmeniť, je potrebné, aby pred použitím funkcie `itgt_writePacket()` upravil požadované položky štruktúry `packet`, ktorá je súčasťou štruktúry `handle`.



Obr. 4.5: Proces pre kopírovanie dát medzi dvoma súbormi

4.6.2 Filtrovanie prijímaných správ

Pri zachytávaní dát, je veľmi praktické mať možnosť zachytávať len dáta o ktoré má užívateľ záujem. Napríklad dáta z jedného konkrétneho zariadenia, alebo len určitý typ paketov. Užívateľ môže mať v rôznych prípadoch použitia na tento filter značne odlišné nároky, z toho dôvodu bolo kľúčové toto filtrovanie vykonávať pomocou *callback* funkcie, ktorú si definuje užívateľ sám.

Callback funkcia `itgt_filter_clbk` berie na vstupe reťazec, ktorý reprezentuje prijatý paket v *raw* formáte. Tento reťazec je získaný v rámci funkcií nástrojov *BTLE Radio packet sniffer/scanner and sender* a *Waving-Z*. Reťazec je v hexadecimálnej podobe a každý bajt je oddelený medzerou. Ďalšími parametrami sú užívateľské dáta a ukazovateľ na samotnú filtrovaciu funkciu definovanú užívateľom. Návratová hodnota funkcie je typu `bool`, pričom hodnota `true` reprezentuje, že paket filtrom prešiel a hodnota `false` značí, že paket bol vyfiltrovaný.

V prípade, že užívateľ funkciu nedefinoval, tak je vracaná automaticky hodnota `true`. V prípade, že užívateľ filtrovaciu funkciu definoval, dáta filtrovaného reťazca a užívateľské dáta sú odovzdané na vstup užívateľom definovanej funkcie, ktorá vyhodnotí výsledok filtrovania a jej očakávaným výstupom je hodnota `true`, alebo `false`.

5 Testovanie

Implementovaný systém bol otestovaný z viacerých hľadísk. Prvým z týchto hľadísk je testovanie na úniky pamäti. Ďalšími hľadiskami sú testovanie úspešnosti príjmu a odosielania.

5.1 Testovanie na úniky pamäti

Testovanie na úniky pamäti bolo vykonané pomocou nástroja Valgrind. Ten bol spustený s nasledovnými parametrami:
`/usr/bin/valgrind --tool=memcheck --xml=yes --xml-file=/tmp/valgrind --gen-suppressions=all --leak-check=full --leak-resolution=med --track-origins=yes --vgdb=n \path\to\file`.

Jednotlivé bloky programu boli kontrolované samostatne. Úniky boli nájdené len pri prijímaní a odosielaní dát technológie Z-Wave. Jednalo sa o úniky typu `Invalid write`. K únikom však dochádzalo v časti kódu, ktorý je súčasťou nástroja *Waving-Z*. V rámci ďalšej optimalizácie tohoto nástroja budú tieto úniky odstránené.

5.1.1 Úspešnosť príjmu

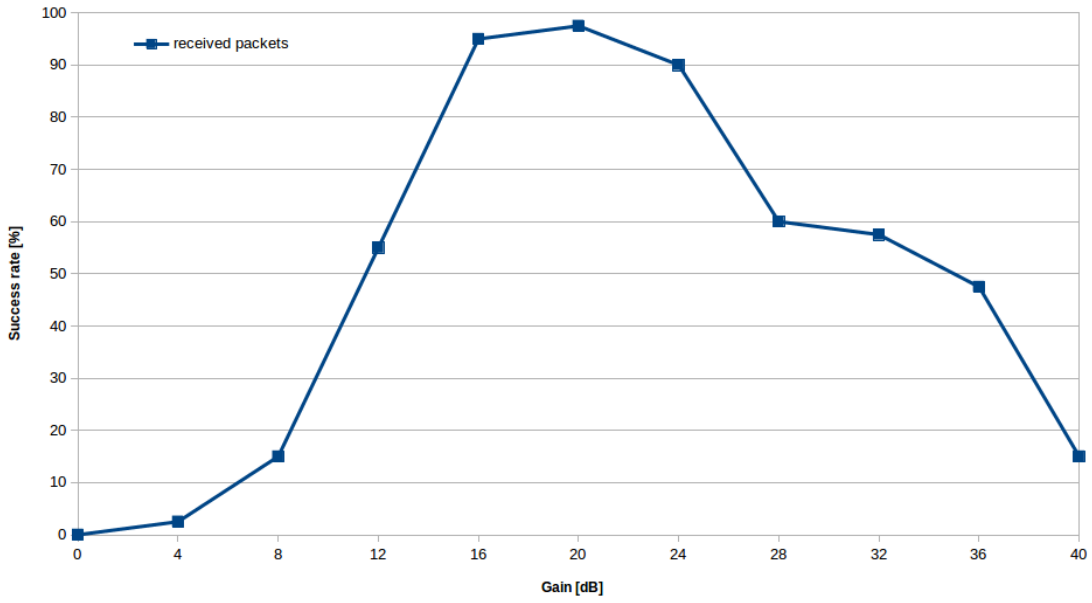
Pri testovaní úspešnosti príjmu bolo sledovaných viacero zariadení pre každú z technológií. V prípade Z-Wave šlo o zásuvku Fibaro FIB-FGWPE-102[1], senzor otvorenia dverí DC-23ZW[2] a senzor svetla, pohybu a teploty AEOTEC ZW100-C[3]. V prípade BLE šlo o žiarovku BeeWi BBL227-A10EU[4] a o Android aplikáciu nRF Connect, ktorá je schopná generovať a odosielať požadované pakety.

Úspešnosť príjmu paketov bola závislá tak od nastavených parametrov, ako aj od vzájomnej polohy zariadení. Meranie bolo uskutočnené pri umiestnení jednotlivých zariadení vo vzdialenosti 1 meter od antény zariadenia HackRF One. Zisk antény bol nastavený na hodnotu 20 dB a šírka pásma na hodnotu 2 MHz. Hodnota zisku antény a šírky pásma bola takto zvolená z dôvodu, že tieto hodnoty sa používajú ako prednastavené, v prípade, že užívateľ hodnoty parametrov nenastavuje.

Pri tomto nastavení Z-Wave dosiahnutá pre obe používané frekvencie (868,42 MHz a 869,85 MHz) priemerná úspešnosť zachytávania paketov na úrovni 90%. Pakety sa však podarilo úspešne dekódovať len v prípade použitia štandardnej bitovej rýchlosti 40 kb/s. Ďalšou optimalizáciou nástroja by však malo byť možné dekódovať aj pakety zvyšných bitových rýchlostí.

[1] https://manuals.fibaro.com/wall-plug/
[2] http://www.climax.com.tw/dc23zw.php
[3] https://aeotec.com/z-wave-sensor
[4] http://www.qioto.com/product/beewi-bbl227-led-color-bulb/

Pre zásuvku Fibaro FIB-FGWPE-102 bola sledovaná aj závislosť úspešnosti príjmu na nastavenom zisku antény. V tomto prípade boli zariadenia umiestnené 3 metre od seba a nemali na seba priamy výhľad. Optimálny zisk antény pri tomto rozmiestnení bol na úrovni 20 dB. Pri zisku antény väčšom ako 24 dB sa začali prijímať okrem paketov aj falošné pakety, ktoré vznikli dekódovaním šumu. Pri zisku antény väčšom ako 28 dB sa začali niektoré prijaté pakety deformovať. Graf zobrazujúci závislosť úspešnosti príjmu na nastavenom zisku antény je zobrazený na Obr. 5.1.

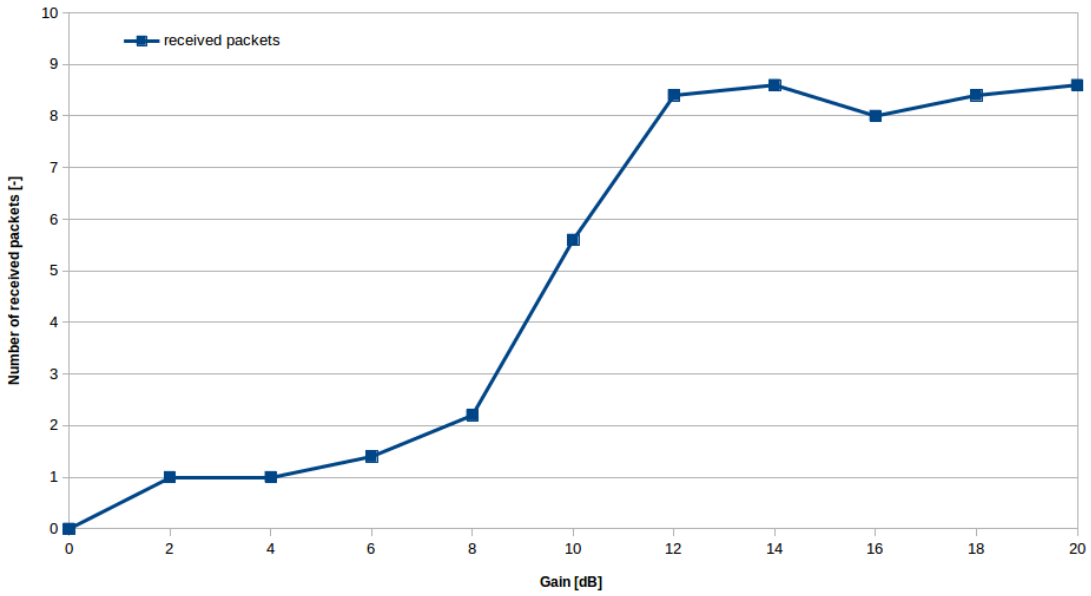


Obr. 5.1: Graf zobrazujúci závislosť úspešnosti príjmu Z-Wave paketov na zisku antény

V prípade BLE boli pri pôvodnom meraní opäť použité prednastavené parametre. Prednastavený zisk antény má v prípade BLE hodnotu 20 dB a šírka pásma má hodnotu 4 MHz. Vzdialenosť meraného zariadenia od antény HackRF One bola opäť 1 meter.

Okrem zachytávania paketov na *advertising* kanáloch sa podarilo úspešne zachytiť aj proces párovania mobilného telefónu so žiarovkou BeeWi BBL227-A10EU vrátane skákania po jednotlivých dátových kanáloch. Vzhľadom na to, že žiadosť o spárovanie sa posiela na náhodnom z troch *advertising* kanálov a zariadenie je schopné v danom momente sledovať len jednu z nich, tak úspešnosť zachytenia procesu párovania žiarovky bola značne limitovaná. Ak sa však podarilo zachytiť jej začiatok, bol zachytený celý proces.

Pre pakety generované aplikáciou nRF Connect bolo sledované množstvo prijatých paketov v závislosti na nastavenom zisku antény. V tomto prípade boli ostatné parametre a rozmiestnenie zariadení bez zmeny. Prijímala sa sekvencia trvajúca 3 sekundy, počas ktorej boli aplikáciou opakovane odosielané *advertising* pakety. Od hodnoty zisku antény 12 dB množstvo prijatých paketov výrazne neklesalo, ani nerástlo. Informácia o celkovom množstve odoslaných paketov nebola k dispozícií a tak bol v tomto prípade zaznamenaný počet zachytených paketov, miesto percentuálnej úspešnosti. Počet zachytených paketov pri súčasnom zachytávaní pomocou zariadenia Ubertooth One bol približne rovnaký. Graf zobrazujúci závislosť úspešnosti príjmu pomocou HackRF One na nastavenom zisku antény je zobrazený na Obr. 5.2.



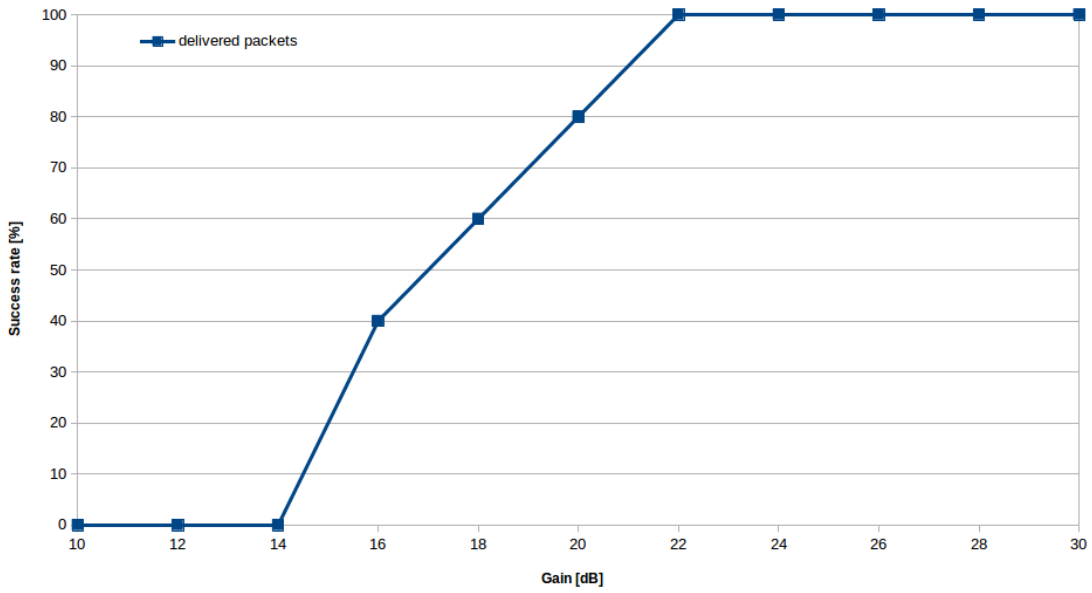
Obr. 5.2: Graf zobrazujúci závislosť úspešnosti príjmu BLE paketov na zisku antény

5.1.2 Úspešnosť odosielania

Pri testovaní odosielania bol u Z-Wave využitý RTL-SDR RTL2832U dongle, ktorý je schopný zachytávať signály v RF pásme 24 MHz až 1.7 GHz. Okrem neho bola využitá podobne ako v príapde príjmu aj zásuvka Fibaro FIB-FGWPE-102. Pri testovaní u BLE bolo využité zariadenie Ubertooth One[5], ktoré pracuje na frekvencií 2,4 GHz a je určené špeciálne pre príjem paketov Bluetooth a BLE. Okrem neho bola opäť využitá aj žiarovka BeeWi BBL227-A10EU.

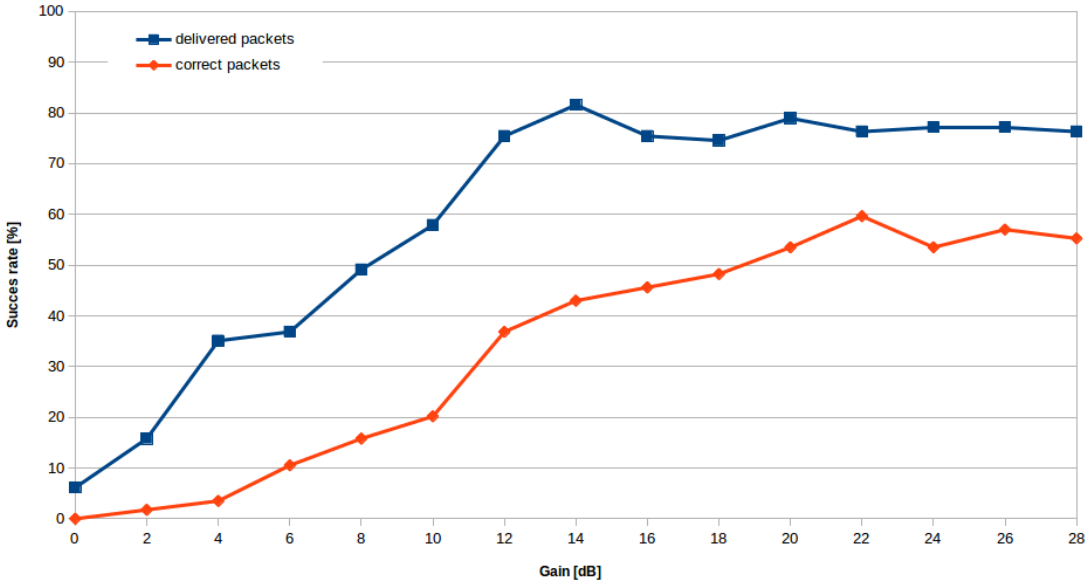
[5]https://greatscottgadgets.com/ubertoothone/

V prípade testovania odosielania paketov boli u Z-Wave nastavené rovnaké parametre ako pri testovaní príjmu. Vzdialenosť zariadení bola 3 metre a zariadenia na seba nemali priamy výhľad. Pri testovaní sa kontrolovali zachytené pakety pomocou zariadenia RTL-SDR RTL2832U a zároveň bola prakticky testovaná funkčnosť príkazov pre vypnutie a zapnutie zásuvky Fibaro FIB-FGWPE-102. Úspešnosti 100% bolo pri pokuse o vypínanie a zapínanie zásuvky dosiahnuté pri ziskoch antény 22 dB a viac. Graf zobrazujúci závislosť úspešnosti odosielania na nastavenom zisku antény je zobrazený na Obr. 5.3.



Obr. 5.3: Graf zobrazujúci závislosť úspešnosti odosielania Z-Wave paketov na zisku antény

U BLE boli v prípade testovania odosielania použité rovnaké parametre ako pri prijímaní. Pakety odoslané pomocou HackRF One boli zachytávané zariadením Ubertooth One. Testované bolo odosielanie sekvencie 40 *advertising* paketov na kanáli 37. Meranie bolo opäť testované v závislosti na nastavenom zisku antény. Najvyššia dosiahnutá priemerná úspešnosť príjmu dosahovala 81,5% pri použitom zisku antény 14 dB. Najvyššia dosiahnutá priemerná úspešnosť prijatých validných paketov so správnym CRC dosahovala 60% pri zisku antény 22 dB. Pri týchto výsledkoch však treba brať v úvahu aj fakt, že zariadenie Ubertooth One nemá dokonalú úspešnosť príjmu a tak mohli ostať niektoré validne odoslané pakety nedetekované. Graf zobrazujúci závislosť úspešnosti odosielania na nastavenom zisku antény je zobrazený na Obr. 5.4.



Obr. 5.4: Graf zobrazujúci závislosť úspešnosti odosielania BLE paketov na zisku antény

6 Záver

Cieľom tejto diplomovej práce bolo na základe prieskumu informácií o vybraných IoT protokoloch vybrať vhodné zariadenie a navrhnúť vhodné softvérové riešenie za účelom generovania paketov a tvorby komunikácie v IoT sieťach. Následne bolo potrebné toto riešenie implementovať a výsledný systém otestovať.

Na základe preskúmania zadaných technológií bola zvolená pre hardvérové riešenie technológia SDR, ktorej využitie je veľmi flexibilné. Pomocou SDR je možné zabezpečiť rozsah zariadenia vo všetkých frekvenčných pásmach, v ktorých jednotlivé IoT technológie pracujú. Tieto technológie síce všetky využívajú moduláciu GFSK, no v prípade budúceho rozširovania tohoto systému o iné technológie môže byť využitie SDR výhodou napríklad aj z tohoto hľadiska.

Medzi dostupnými SDR zariadeniami bol následne vykonaný prieskum, ktorý porovnával ich vlastnosti, cenu, dostupnosť a rozšírenosť. Spomedzi dostupných zariadení bolo napokon vybrané HackRF One. To spĺňalo hardvérové požiadavky, bolo dostupné priamo v Českej republike a spomedzi skúmaných zariadení bolo najviac rozšírené a využívané v projektoch spojených s tematikou IoT.

K zariadeniu HackRF One som vďaka jeho rozšírenosti našiel funkčné nástroje tretích strán, ktoré sú schopné generovať a odosielať pakety pre technológie Z-Wave a BLE. Vzhľadom na proprietárnosť a menšiu rozšírenosť technológie IQRF sa mi u nej nepodarilo nájsť funkčný nástroj pre generovanie paketov. Ten by však mohol v budúcnosti pri svojej implementácií čerpať zo zvyšných dvoch nástrojov, vďaka podobnosti technológií v niektorých smeroch. Zdrojové kódy firmvéru aj softvéru obslužného programu k HackRF One sú takisto ako vyššie spomínané nástroje voľne dostupné na Internete, čo takisto pomôže uľahčiť budúcu prácu.

Na základe predchádzajúcich zistení bolo v jazyku C navrhnuté API výsledného systému, na základe ktorého bol nástroj následne implementovaný. API bude užívateľovi umožňovať prácu s viacerými zariadeniami, ku ktorým bude môcť pristupovať pomocou štruktúry *handle*. Užívateľovi bude API umožňovať, okrem samotného odosielania a prijímania paketov, aj ďalšie úkony. Medzi ne patrí napríklad možnosť meniť parametre prenosu, filtrovať príchodzie pakety, ukladať dáta do súborov a čítať ich, alebo čítať a upravovať jednotlivé polia prijatých paketov.

V rámci implementácie bolo kľúčové najskôr prepojiť použité nástroje navzájom medzi sebou a s výsledným systémom. To bolo docielené zmenou výsledného formátu dát, pridaním parametrov u funkcií nástrojov tretích strán a v niektorých prípadoch tvorbou nových procesov a ich prepojením pomocou *pipeline*. Dodatočné informácie pre funkcie jednotlivých nástrojov boli posielané pridaním parametra *handle*, ktorý obsahoval štruktúru s informáciami o zariadení, štruktúru obsahujúcu dáta a metadáta aktuálneho paketu, štruktúru obsahujúcu súbor pre čítanie/zápis a ďalšie

potrebné informácie. Dáta paketov sú v tomto nástroji ukladané vo formáte UniRec. Pri využití nástroja pre jeho primárny účel na penetračné testovanie v rámci projektu SIoT, zabezpečí použitie tohoto formátu lepšiu interoperabilitu nástroja s ďalšími modulmi. Pomocou UniRec funkcií je potom možné aj jednoduché získavanie a upravovanie dát paketu. V prípade zapisovania a čítania zo súboru je aktuálne použitý vlastný formát, ktorý okrem dát paketu vo formáte UniRec obsahuje aj ďalšie metadáta. V budúcnosti však v prípade potreby môže byť formát dát v súbore upravený na tradičný spôsob využívaný u UniRec dát. Užívateľovi je takisto umožnené filtrovanie prijímaných správ a zaznamenávanie len tých, ktoré splňujú užívateľom definované kritériá.

Pri testovaní sa ukázalo, že pomocou frameworku je možné zachytávať aj odosielať pakety technológií Z-Wave a BLE. Úspešnosť nie je bezchybná a pre spoľahlivé využitie frameworku je nutná ďalšia optimalizácia. Napriek tomu sa pomocou frameworku podarilo zachytiť a úspešne zreprodukovať príkaz pre vypnutie a zapnutie testovanej zásuvky Fibaro FIB-FGWPE-102.

V budúcnosti je možné tento nástroj rozširovať o ďalšie protokoly a takisto ho ďalej optimalizovať a testovať.

Literatúra

[1] *History of Bluetooth Special Interest Group* [online]. [cit. 20. 10. 2018] , dostupné z URL: <https://www.bluetooth.com/about-us/our-history>

[2] TOWNSEND, K.; DAVIDSON, R.; AKIBA; CUFI, C. *Getting started with Bluetooth low energy: tools and techniques for low-power networking.* Revised First Edition. Sebastopol, CA: O'Reilly, 2014. ISBN 978-149-1949-511.

[3] GOMEZ, C.; OLLER, J.; PARADELLS, J. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 2012, 12.9: 11734-11753.

[4] *Radio versions* [online]. [cit. 20. 10. 2018] , dostupné z URL: <https://www.bluetooth.com/bluetooth-technology/radio-versions>

[5] *Bluetooth 5 speed: How to achieve maximum throughput for your BLE application* , posledná aktualizácia 7. 8. 2018. [cit. 20. 10. 2018] , dostupné z URL: <https://www.novelbits.io/bluetooth-5-speed-maximum-throughput/>

[6] *Bluetooth Core Specification, Rev. 5.0* , posledná aktualizácia 6. 12. 2016. [cit. 20. 10. 2018] , dostupné z URL: <https://www.bluetooth.com/specifications/bluetooth-core-specification>

[7] WITHANAGE, C.; ASHOK, R.; YUEN, C.; OTTO K. A comparison of the popular home automation technologies. *IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)*, 2014, ISNB 978-1-4799-1300-8

[8] FOULADI, B.; GHANOUN, S. *Security Evaluation of the Z-Wave Wireless Protocol*

[9] RICE, M.; BADENHOP, C.; FULLER, J.; HALL, J.; RAMSEY, B. Evaluating ITU-T G.9959 based wireless systems used in critical infrastructure assets. *Critical Infrastructure Protection IX.*, Springer,2015. ISBN 978-3-319-26566-7

[10] *IQRF OS User's guide, Ver. 4.02 for TR-7xD* [online] , posledná aktualizácia 23. 8. 2017 [cit. 27. 11. 2018] , dostupné z URL: <http://www.iqrf.org/weben/downloads.php?id=155>

[11] *IQRF OS Ref. guide, Ver. 4.02 for TR-7xD* [online] , posledná aktualizácia 23. 8. 2017 [cit. 27. 11. 2018] , dostupné z URL: <https://www.iqrf.org/IQRF-OS-Reference-guide/>

[12] *IQRF DPA Framework – Technical Guide, Ver. 3.02* [online] , posledná aktualizácia 23. 8. 2017 [cit. 27. 11. 2018] , dostupné z URL: <https://www.iqrf.org/DpaTechGuide/>

[13] *SPIRIT1 - Low data rate, low power sub-1GHz transceiver* [online] , 10/2016, [cit. 27. 11. 2018] , dostupné z URL: <https://www.st.com/en/wireless-connectivity/spirit1.html>

[14] KUCHTA, R.; VRBA, R.; SULC, V.; Smart Wireless Communication Platform IQRF. *Mobile and Wireless Communications Network Layer and Circuit Level Design*, Salma Ait Fares and Fumiyuki Adachi (Ed.), 2010, ISBN: 978-953-307-042-1, InTech

[15] *HackRF Wiki* [online] , [cit. 27. 11. 2018] , dostupné z URL: <https://github.com/mossmann/hackrf/wiki>

[16] *hackrf* [online]. [cit. 1. 12. 2018] , dostupné z URL: <https://github.com/mossmann/hackrf>

[17] *Waving-Z* [online]. [cit. 1. 12. 2018] , dostupné z URL: <https://github.com/baol/waving-z>

[18] *A BTLE radio packet sniffer/scanner and sender* [online]. [cit. 1. 12. 2018] , dostupné z URL: <https://github.com/JiaoXianjun/BTLE>

[19] *UniRec* [online]. [cit. 21. 4. 2019] , dostupné z URL: <https://github.com/CESNET/Nemea-Framework/tree/master/unirec>

Zoznam symbolov, veličín a skratiek

AES	Advanced Encryption Standard
API	Application Programming Interface
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
Bluetooth SIG	Bluetooth Special Interest Group
BR/EDR	Bluetooth Basic Rate/Enhanced Data Rate
CRC	Cyclic Redundancy Check
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GFSK	Gaussian Frequency-Shift keying
HCI	Host Controller Interface
ID	Identifikátor
IoT	Internet of Things
L2CAP	Logical Link Control and Adaptation Protocol
LE PHY	LE Physical Layer
LL	Link Layer
LTK	Long Term Key
MIMO	Multiple-input and multiple-output
NFC	Near-field communication
NRZ	Non return to zero
OOB	Out of Band
OS	Operačný systém
RFPGM	RF Programming
SC	Secure Connections
SDR	Softvérovo definované rádio
SMA	SubMiniature version A
SMP	Security Manager Protocol
SPI	Serial Peripheral Interface
TK	Temporary Key
USB	Universal serial bus