VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

LOKALIZACE ZAŘÍZENÍ V BEZDRÁTOVÉM SYSTÉMU NA ZÁKLADĚ ÚROVNĚ PŘIJÍMANÉHO SIGNÁLU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

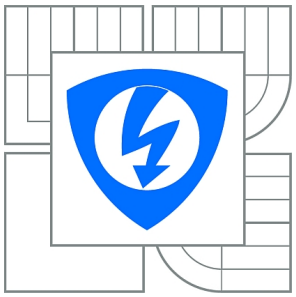
AUTOR PRÁCE
AUTHOR

Bc. JURAJ POPOVEC

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

LOKALIZACE ZAŘÍZENÍ V BEZDRÁTOVÉM SYSTÉMU NA ZÁKLADĚ ÚROVNĚ PŘIJÍMANÉHO SIGNÁLU

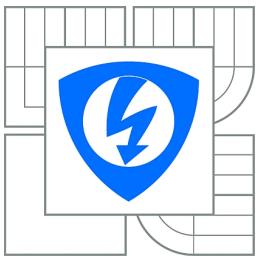
RSSI BASED LOCALIZATION OF SENSOR UNITS IN WIRELESS NETWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JURAJ POPOVEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. MIROSLAV BOTTA
SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Juraj Popovec ***ID:*** 115260
Ročník: 2 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Lokalizace zařízení v bezdrátovém systému na základě úrovně přijímaného signálu

POKYNY PRO VYPRACOVÁNÍ:

Úlohou studenta v rámci řešení diplomové práce bude nastudování problematiky určování vzdálenosti mezi bezdrátovými jednotkami v senzorové sítí. Student provede analýzu radiového modelu a implementuje, nebo simuluje metodu umožňující lokalizování bezdrátové jednotky v reálném čase. Vytvořený systém porovná se známými technikami pro odhad vzdálenosti mezi bezdrátovými uzlami.

DOPORUČENÁ LITERATURA:

[1] Ajay Malik. 2009. RTLS for Dummies. For Dummies. URL: http://cias.rit.edu/~nmtp/nmtp/teamkittycat/RTLSforDummies.pdf
[2] FARAHANI, Shahin. Zigbee Wireless Networks and Transceivers. [s.l.] : Elsevier, 2008. 329 s. ISBN 978-0-7506-8393-7
[3] Awad, A.; Frunzke, T.; Dressler, F., Adaptive Distance Estimation and Localization in WSN using RSSI Measures, Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on , vol., no., pp.471,478, 29-31 Aug. 2007, doi: 10.1109/DSD.2007.4341511, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4341511&isnumber=4341433

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 30.5.2014

Vedoucí práce: Ing. Miroslav Botta
Konzultanti diplomové práce:

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

ABSTRAKT

Táto práca popisuje spracovanie parametra RSSI a jeho následné použitie k výpočtu vzdialenosti medzi bezdrôtovými uzlami. Taktiež sa zaoberá analýzou rádiového modelu prostredia a kalibráciou jednotlivých veličín potrebných pre lokalizáciu. Následne je realizovaný systém pre lokalizáciu bezdrôtových uzlov v senzorovej sieti. Pre výpočty využíva dynamicky kalibrované veličiny popisujúce rádiový model.

KĽÚČOVÉ SLOVÁ

RSSI, bezdrôtové senzorové siete, rádiový propagačný model, lokalizácia

ABSTRACT

This thesis describes processing of RSSI parameter and its subsequently use for calculating distance between wireless nodes. This thesis also describes analysis of radio model environment and calibration of key variables needed for localization. There is also system realized for localization of wireless nodes in sensor network. It uses dynamically calibrated variables for calculations, which describes radio model.

KEYWORDS

RSSI, wireless sensor networks, radio propagation model, localization

POPOVEC, Juraj *Lokalizace zařízení v bezdrátovém systému na základě úrovně přijímaného signálu*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 56 s. Vedúci práce bol Ing. Miroslav Botta

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Lokalizace zařízení v bezdrátovém systému na základě úrovně přijímaného signálu“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce Ing. Miroslavovi Bottovi za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výskum popísaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených z projektu SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pre inovácie.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007-13
OP Výzkum a vývoj pro inovace

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Táto práca sa venuje problematike určovania vzdialeností medzi zariadeniami v bezdrôtových sietiach. K tejto úlohe boli využité senzorové bezdrôtové siete, tiež označované ako WSN (Wireless Sensor Networks). K určovaniu vzdialeností je využitý parameter RSSI (Received Signal Strength Indication). V sietiach WSN je to jedna z najednoduchšie implemetovateľných metód, keďže nevyžaduje žiaden neštandardný hardvér.

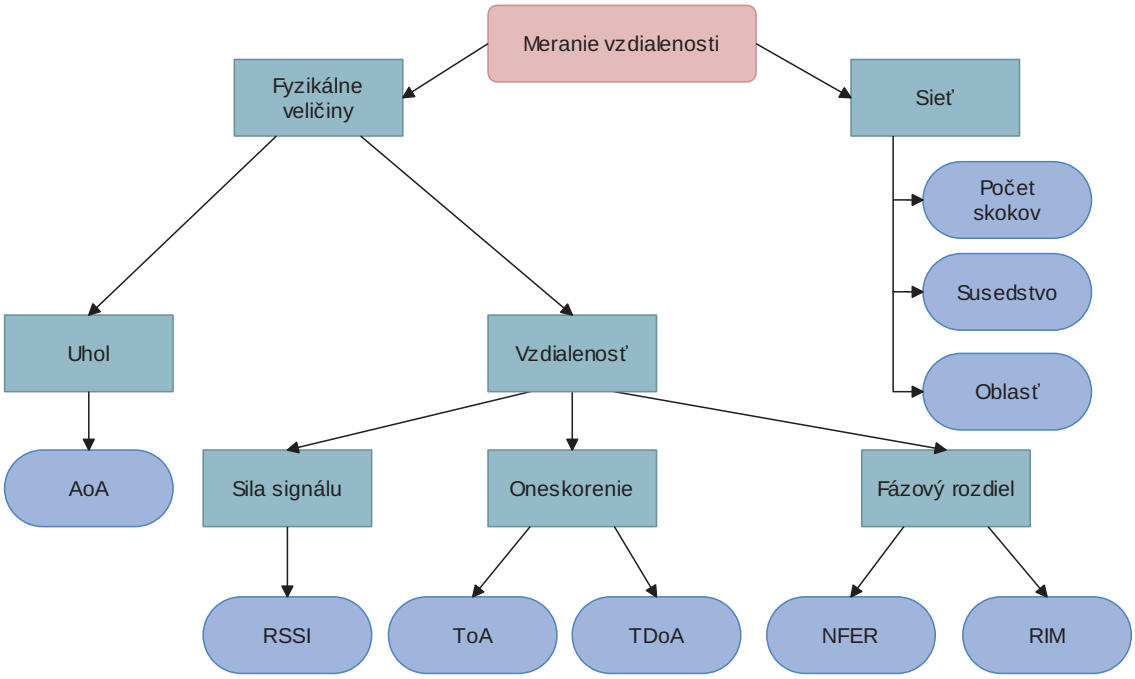
Prvá kapitola obsahuje rozdelenie a popis metód používaných pre určovanie vzdialeností medzi bezdrôtovými jednotkami v sieti. Dôraz sa kladie na metódu využívajúcu parameter RSSI. Nasleduje analýza rádiového modelu a parametrov definujúcich rádiové prostredie. V ďalšej kapitole je popísaný návrh a realizácia systému umožňujúceho určenie vzdialeností medzi bezdrôtovými jednotkami siete v reálnom čase. Následne sa vytvorený systém využije k porovnaniu štyroch scenárov výpočtu vzdialeností založených na parametri RSSI.

1 URČOVANIE VZDIALENOSTI MEDZI JEDNOTKAMI V BEZDRÔTOVEJ SIETI

Meranie vzdialenosti medzi jednotkami v bezdrôtovej sieti je prvým krokom potrebným pre lokalizáciu jednotky. Podľa autorov v [1] je možné meranie rozdeliť podľa obrázka 1.1.

Meranie založené na topológii siete umožňuje využiť vlastnosti siete k odhadu vzdialenosti, ako je počet skokov k cieľkovému uzlu v sieti alebo zoznam susedov cieľového uzla. Tieto techniky sú vhodné pre hrubý odhad vzdialenosti.

Druhá skupina metód využíva k meraniu vzdialenosti vhodnú fyzikálnu veličinu. Tieto metódy je možné ďalej rozdeliť do dvoch podskupín. Jedna podskupina využíva k odhadu vzdialenosti metódu AoA (Angle of Arrival). K výpočtu využíva uhol, ktorý vznikne medzi smerom šírenia signálu a referenčným smerom. Vyplýva z toho, že prijímače musia byť vybavené anténami schopnými určiť smer prijímaného signálu. Na výpočet pozície uzla sa vytvoria priamky v smere prichádzajúceho signálu minimálne z dvoch prijímačov. Ich priesečník je pozícia vysielajúceho uzla. Pri tejto metóde malá chyba v zmeranom uhle môže spôsobiť veľkú chybu v lokalizácii.



Obr. 1.1: Rozdelenie merania vzdialenosti.

Druhá podskupina využíva fyzikálnu veličinu k priamemu prevodu na vzdialenosť. Metóda využivajúca veľkosť prijatej energie (RSSI) je použitá v tejto práci a je popísaná nižšie. Ďalšou možnosťou je pre výpočet vzdialenosti využitie času šírenia signálu. ToA (Time of Arrival) metóda využíva k zisteniu vzdialenosti čas, ktorý uplynul od vyslania dát, až po ich príjem. Na základe tohto času a známej rýchlosti šírenia signálu sa dopočíta vzdialenosť, ktorú urazil. Pri tomto spôsobe merania vzdialenosti je potrebné mať zosynchronizované časovače vysielača a prijímača. Tento nedostatok odstraňuje metóda TDoA (Time Difference of Arrival), kde je časová synchronizácia potrebná len pre referenčné uzly. K určeniu vzdialenosti sa tak využije časový rozdiel príjmu dát od minimálne dvoch referenčných uzlov k cieľovému uzlu.

Využitie fázového rozdielu širenia signálu je taktiež možné využiť pre odhad vzdialenosti. Nízkofrekvenčné šírenie signálu ma dobré penetračné schopnosti a preto je tento spôsob odhadu vzdialenosti vhodný v prostredí obsahujúcom prekážky. Taktiež nie je nutné synchronizovanie času v sieti. Podrobnné informácie a vzťahy potrebné k výpočtom pomocou jednotivých metód je možné nájsť v literatúre [1] a [2].

1.1 Indikátor energie prijatého signálu (RSSI)

RSSI (received signal strength indication) je parameter, ktorý sa získava meraním energie prijatého rádiového signálu.

Po vyslaní signálu zo zdroja pôsobí na tento signál útlm prostredia a jeho energia klesá. Pokles je logaritmický a v rôznych prostrediach je dobre definovateľný. Pretože je známy vysielací výkon a útlm prostredia, je možne určiť vzdialenosť medzi uzlami. To však nie je vždy jednoduché a presné, keďže prostredie, v ktorom meriame, nie je konštantné. Meraná hodnota RSSI je ovplyvňovaná prekážkami v prostredí, viaccestnosťou signálu, teplotou, vlhkosťou, pohybom osôb a podobne. Vďaka tomu môže byť nameraná hodnota často vyššia alebo nižšia ako konvenčne pravá hodnota.

1.1.1 Výpočet RSSI

Pri meraní bol použitý čip ATmega128RFA1. Podľa manuálového listu [3] je do najnižších piatich bitov osembitového regista PHY_RSSI počas príjmu každé 2 µs uložená hodnota automatizovaného merania RSSI. Uložená hodnota je v rozsahu 0 až 28. Hodnota 0 indikuje RSSI menšie ako -90 dBm a hodnota 28 indikuje RSSI vyššie alebo rovné ako -10 dBm. RSSI sa vypočíta podľa vzťahu 1.1.

$$RSSI = RSSI_{\mathrm{BASE\_VAL}} + 3 \cdot (RSSI_{\mathrm{REG}} - 1) \qquad [\mathrm{dBm}] \tag{1.1}$$

$RSSI_{\mathrm{BASE\_VAL}}$ je hodnota minimálnej citlivosti antény. V prípade čipu ATmega128RFA1 to je -90 dBm. Premenná $RSSI_{\mathrm{REG}}$ obsahuje hodnotu prečítanú z registra PHY_RSSI.

1.1.2 Aproximácia RSSI

Ako bolo vyššie spomenuté, parameter RSSI je podstatne nestabilný. Pred tým ako sa použije pre určenie vzdialenosti je potrebné ho aproximovať, aby boli výsledky čo najpresnejšie. Jedna z najčastejšie používaných aproximačných metód je štatisticky stredná hodnota. K jej výpočtu sa použije vzťah 1.2.

$$RSSI = \frac{1}{n} \cdot \sum_{m=1}^{n} RSSI_m \qquad [\mathrm{dBm}] \tag{1.2}$$

Vo vyššie uvedenom vzťahu je premenná n počet hodnôt RSSI, ktorý sa vyhodnocuje. S narastajúcou hodnotou tohto parametra, pri rovnakom čase, stúpa presnosť metódy.

K ďalším aproximačným metódam je možné zaradiť výpočet mediánu, či určnie strednej hodnoty pomocou gaussovej metódy.

2 RÁDIOVÝ MODEL

Vyžitie parametra RSSI na meranie vzdialenosti medzi bezdrôtovými uzlami spočíva na poznatku, že RSSI je funkciou vysielaného výkonu a vzdialenosti medzi vysielačom a prijímačom. V reálnom prostredí je však signál ovplyvňovaný interferenciami a odrazmi. Preto je v tejto práci pre popis rádiového prostredia aplikovaný vzťah 2.1 použitý v článku [4].

$$P_{\mathrm{r}}(d) = P_{\mathrm{r}}(d_0) - 10\eta \log\left(\frac{d}{d_0}\right) \qquad [\mathrm{dBm}] \tag{2.1}$$

Pre výpočet vzdialenosti je odvodený vzťah 2.2.

$$d = d_0 \cdot 10^{\left(\frac{P_{\mathrm{r}}(d_0) - P_{\mathrm{r}}(d)}{10 \cdot \eta}\right)} \qquad [\mathrm{m}] \tag{2.2}$$

$P_{\mathrm{r}}(d)$ je hodnota prijatého signálu vo vzdialenosti d a $P_{\mathrm{r}}(d_0)$ je hodnota prijatého signálu v referenčnej vzdialenosti (d_0). Obe hodnoty sú v jednotkách [dBm]. Premenná η predstavuje index stratovosti prostredia a jej hodnota sa mení na základe prostredia, v ktorom prebieha meranie.

2.1 Index stratovosti prostredia

Ako už bolo spomenuté, na výkon prijatého signálu ma podstatný vplyv prostredie. V rôzných prostrediach neklesá výkon signálu so vzdialenosťou rovnako. Pre zohľadnenie tohto javu v matematických vzťahoch sa používa index stratovosti prostredia η. Ten nadobúda pre rozličné prostredia rôzne hodnoty. K určeniu vzdialenosti na základe výkonu prijatého signálu je nutné najskôr určiť hodnotu η.

Pre určenie indexu stratovosti prostredia je použitý vzťah 2.4 odvodený z 2.3 [4].

$$P_{\mathrm{r}}(d) = P_{\mathrm{t}} \cdot \left(\frac{1}{d}\right)^{\eta} \qquad [\mathrm{mW}] \tag{2.3}$$

$$\eta = \frac{\log\left(\frac{P_{\mathrm{r}}(d)}{P_{\mathrm{t}}}\right)}{\log\left(\frac{1}{d}\right)} \qquad [-] \tag{2.4}$$

$P_{\mathrm{r}}(d)$ je hodnota prijatého signálu vo vzdialenosti d. P_{t} je hodnota vysielaného signálu. Obe hodnoty sú v jednotkách [mW], preto je potrebné previesť hodnotu prijatého signálu z jednotiek [dBm] pomocou vzťahu 2.5.

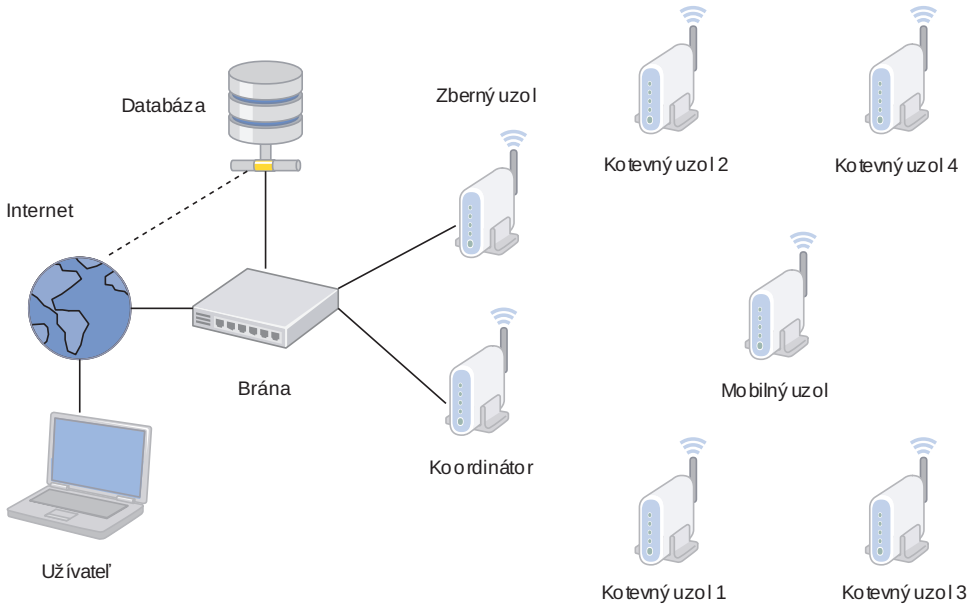
$$P_{(\mathrm{mW})} = 10^{\left(\frac{P_{(\mathrm{dBm})}}{10}\right)} \qquad [\mathrm{mW}] \tag{2.5}$$

Pre určenie indexu stratovosti prostredia sa väčšinou volí referenčná vzdialenosť 1 m. Autori v [5] však meraniami zistili, že najvhodnejšou referenčnou vzdialenosťou nemusí byť 1 m. Preto sa v tejto práci využíva pre určenie indexu stratovosti vzdialenosť určená rozmiestnením referenčných jednotiek v bezdrôtovej sieti.

3 NÁVRH A REALIZÁCIA LOKALIZAČNÉHO SYSTÉMU

3.1 Prehľad systému

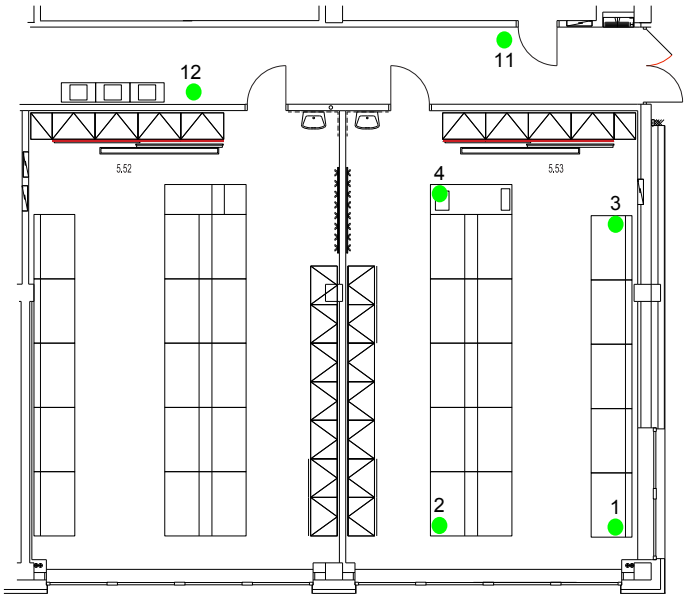
Táto kapitola popisuje návrh a realizáciu systému, ktorý je schopný určovať vzdialenosť medzi bezdrôtovými jednotkami siete v reálnom čase. Na obrázku 3.1 sú znázornené jeho jednotlivé prvky. Odhad vzdialenosti prebieha medzi kotevnými uzlami a mobilným uzlom. Všetky uzly počas behu systému rozosielajú na všesmerovú adresu hello pakety. Po prijatí hello paketu kotevným uzlom je táto udalosť oznamovaná zbernému uzlu. Kotevný uzol mu zasiela údaje o adrese odosielateľa, hodnote RSSI a LQI (link quality indication). Zberný uzol k údajom pridá adresu kotevného uzla, ktorý mu paket zaslal. Tým pádom má údaje o stave spojenia medzi odosielateľom a príjemcom hello paketu. Údaje prepošle pomocou sériového portu bráne. Jej úlohou je importovať prijaté dáta do databázy, v ktorej prebieha lokalizačný výpočet.



Obr. 3.1: Prvky navrhnutého systému.

3.1.1 Použité algoritmy

Vzdialenosť medzi kotevnými uzlami a mobilným uzlom je vypočítavaná pomocou staticky aj dynamicky nastavovaných parametrov. Kotevné uzly boli v meranom prostredí rozmiestnené podľa obrázka 3.2 vždy v pároch. Jeden kotevný uzol bol umiestnený vo výške 0,75 m a jeho párový uzol vo výške 2,9 m. Celkovo prebiehal výpočet podľa štyroch scenárov súčasne.



Obr. 3.2: Umiestnenie kotevných uzlov v meranom prostredí.

- **Prvý scenár** vypočítava vzdialenosť medzi uzlami na základe vopred nastavených parametrov prostredia. Tie boli určené nasledujúcim postupom. V prvom kroku bola nameraná hodnota RSSI v referenčnej vzdialenosti 1 m. Keďže vyžarovanie antény nie je izotropické, meranie bolo opakované štyrikrát, vždy pootočené o 90°. Výsledná hodnota RSSI vznikla spriemerovaním dielčích meraní. Druhým krokom bolo zmeranie hodnoty RSSI vo vzdialenosti 5 m. Odvodením a dosadením do vzťahu 2.1 bola získana hodnota η.
  Pre určenie vzdialenosti sú využívané kotevné uzly umiestnené vo výške 2,9 m.
- **Druhý scenár** pracuje s uzlami umiestnenými vo výške 0,75 m. Vzdialenosť medzi uzlami sa vypočítava na základe algoritmu použitom v článku [5].

Pre dvojicu kotevný uzol a mobilný uzol sa v databáze vyhľadá referenčný kotevný uzol, ktorý má od mobilného uzla najvyššie RSSI. Tým sa predpokladá, že sa nachádza v jeho blízkosti a v rovnakom prostredí. Známe sú teda všetky potrebné premenné k výpočtu vzdialenosti. RSSI medzi kotevným uzlom a mobilným uzlom, vzdialenosť, vysielací výkon a RSSI medzi kotevným a referenčným kotevným uzlom.

- **Tretí scenár** využíva k výpočtu rovnaký algoritmus. Rozdiel je v tom, že sa pracuje s kotevnými uzlami umiestnenými vo výške 2,9 m.
- **Štvrtý scénar** počíta s použitím všetkých kotevných uzlov. V prvom kroku sa vypočíta vzdialenosť medzi kotevným uzlom vo výške 0,75 m a mobilným uzlom za predpokladu, že kalibračný kotevný uzol je najbližší uzol k mobilnému uzlu vo výške 2,9 m. V druhom kroku sa zase využije kotevný uzol vo výške 2,9 m a ako kalibračný kotevný uzol sa využije uzol vo výške 0,75 m. Do úvahy sa berie kratšia zo vzdialeností. Týmto spôsobom sa dosiahne trojrozmernej aproximácie prostredia na rozdiel od plošnej v predchádzajúcich prípadoch.

3.2 Bezdrôtové uzly a SerialNet

Na vytvorenie bezdrôtovej siete sú použité uzly pozostávajúce z dosky deRFnode [6] a jednočipového riešenia deRFmega128 [7]. Ako firmver je použitý SerialNet [8] od spoločnosti Atmel. Jednotlivé nastavenia sa konfigurujú pomocou AT príkazov. Po pripojení uzla do siete je možné zadávať príkazy aj vzdialene. Použité príkazy nastavenia uzla sú zobrazené v tabuľke 3.1. Prvotná konfigurácia uzlov prebieha pomocou python skriptu A.1.

V tabuľke 3.2 je zobrazené rozdelenie adresného priestoru. Kotevným uzlom vo výške 2,9 m bola pridelená adresa vždy o x7F00 vyššia ako ich párovým uzlom vo výške 0,75 m. Zberným uzlom je vyhradených 15 adries z dôvodu aplikovania prípadného vyvažovania záťaže pre väčšie siete.

Tab. 3.1: AT príkazy použité pre konfiguráciu uzla.

Príkaz	**Funkcia**
AT+WNWKPANID=FEEC	skrátená forma PAN ID
AT+WCHMASK=1000000	nastavenie čísla kanála
AT+WROLE=1	funkcia uzla
AT+WSRC=1	skrátená forma adresy uzla
AT+WAUTONET=1	automatické pripojenie do siete
AT+WTXPWR=3	vyslielací výkon
AT+WPASSWORD 0	heslo pre zadávanie vzdialených príkazov
AT+HELL=5	interval rozosielania hello paketov
AT+RSS FFF0	preposielanie dát o prijatých hello paketoch na zberný uzol

Tab. 3.2: Rozdelenie adresného priestoru siete.

Typ uzla	**Adresa v šestnástkovej sústave**	**Adresa v desiatkovej sústave**
Koordinátor	x0000	0
Kotevný uzol; 0,75 m	x0001 – x7EFF	1 – 32511
Kotevný uzol; 2,9 m	x7F01 – xFDFF	32513 – 65023
Mobilný uzol	xFF00 – xFFEF	65280 – 65519
Zberný uzol	xFFF0 – xFFFE	65520 – 65634

3.2.1 Funkcie uzlov

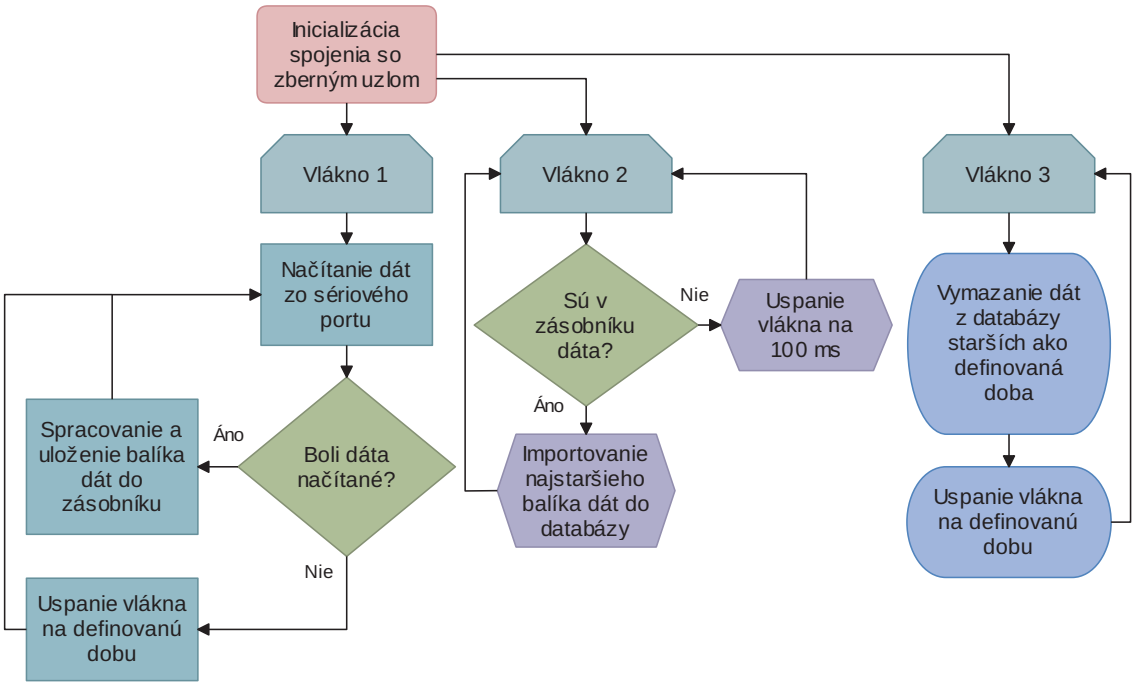
- **Koordinátor** slúži ako základný prvok pri vytvorení siete. V celej sieti sa môže nachádzať iba jeden takýto uzol. Je napájaný pomocou USB kábla priamo z brány. Slúži aj ako rozhranie pre zadávanie vzdialených príkazov ostatným uzlom.
- **Kotevné uzly** sú umiestnené na známych súradniciach v meranom prostredí.

Ich úlohou je v pravidelných intervaloch 5 s zasielať hello paket na všesmerovú adresu. Druhou úlohou je preposielať údaje získané z prijatých hello paketov na zberný uzol. Ich obsah tvorí adresa odosielateľa, hodnota RSSI a LQI.

- **Mobilný uzol** zasiela hello paket na všesmerovú adresu v intervaloch 2 s.
- **Zberný uzol** podobne ako koordinátor je napájaný z brány. Po prijatí paketu od kotevného uzla zašle údaje o adrese odosielateľa, adrese príjemcu, hodnote RSSI a LQI pomocou sériového portu bráne.

3.3 Brána

Brána tvorí rozhranie medzi vytvorenou bezdrôtovou sieťou a internetom. Pri realizácii bol použitý embedded počítač ALIX [9] s operačným systémom Voyage [10]. Jedná sa o zoštíhlenú verziu systému Debian. Pred pripojením koordinátora a zberného uzla pomocou USB kábla je nutné tieto prvky pridať k podporovaným zariadeniam. To sa prevádza pomocou návodu [11]. Po pridaní je možné komunikovať s uzlami pomocou sériového portu. V systéme sú označené ako /dev/ttyUSB0 a /dev/ttyUSB1.



Obr. 3.3: Vývojový diagram aplikácie určenej k importovaniu dát do databázy.

Počas behu systému je spustená aplikácia RelayAgnet.py A.2, slúžiaca k importovaniu prijatých dát zo zberného uzla do databázy. Na obrázku 3.3 je znázornený jej vývojový diagram. Beží v troch vláknach, kde prvé vlákno načíta dáta zo sériového portu a celý balík uloží do zásobníku typu prvý dnu, prvý von. Druhé vlákno pristupuje k zásobníku a importuje dáta do databázy. Posledné vlákno sa stará o mazanie dát z databázy starších ako užívateľom nastavená doba.

3.4 Databáza

Pre uchovávanie dát je použitý voľne dostupný databázový systém PostgreSQL [12]. Je nainštalovaný priamo na zariadení ALIX. K databáze sa preto pri importe dát pristupuje lokálne. V prípade potreby vyššieho výpočetného výkonu je možne nainštalovať PostgreSQL na vzdialený server a pristupovať k databáze vzdialene.

Pre uchovávanie dát je vytvorený systém tabuliek. V databázovom systéme sú vytvorené spúšťače, ktoré reagujú na jednotlivé podnety. Môže sa jednať o pridanie či aktualizovanie riadka v tabuľke, alebo aktualizovanie iba jedinej hodnoty. Zoznam spúšťačov a podnety, na ktoré reagujú, sú uvedené v tabuľke 3.9. Tabuľka 3.8 obsahuje zoznam všetkých funkcií spúšťaných pomocou spúšťačov.

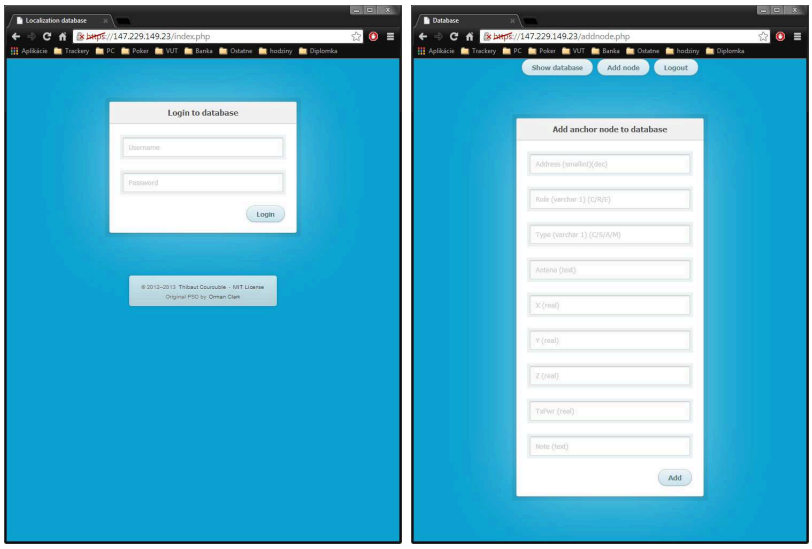
3.4.1 Navrhnutý tabuľkový systém

Prvá tabuľka databázy sa nazýva „nodes“. Slúži na definovanie údajov o uzloch. Dáta sa do tabuľky zadávajú pred spustením celého výpočetného systému, keďže sa využívajú pri výpočtoch. Pre jednoduchšie pridávanie uzlov do tabuľky bolo vytvorené webové rozhranie 3.4.1. Jednotlivé položky navrhnutej tabuľky sú zobrazené v tabuľke 3.3.

Druhá tabuľka databázy sa nazýva „data“ 3.4. Aplikácia RelayAgent.py do nej importuje záznamy s údajmi o zdrojovej adrese, cieľovej adrese, RSSI a LQI z paketov prijatých zberným uzlom. Spolu s tabuľkou „nodes“ tvoria základ dátovej štruktúry databázy. Z týchto dvoch tabuliek sa získavajú údaje potrebné k napĺňaniu ostaných tabuliek.

Tab. 3.3: Názvy stĺpcov, dátové typy a ich význam tabuľky „nodes“.

Názov stĺpca	**Dátový typ**	**Význam**
src	integer PK	zdrojová adresa
role	char var(1)	C – koordinátor, R – smerovač, E – koncové zariadenie
type	char var(1)	C – koordinátor, S – zberný uzol A – kotevný uzol, M – mobilný uzol
antena	text	typ antény
xpos	real	pozícia na osi X
ypos	real	pozícia na osi Y
zpos	real	pozícia na osi Z
txpwr	real	vysielací výkon uzla
note	text	poznámka
timestamp	timestamp	časová značka



Obr. 3.4: Webové rozhranie pre zadávanie dát do tabuľky „nodes“.

Po vložení nového riadka do tabuľky „data“ sa spustí funkcia „add_timestamp_now()“ B.1, ktorá pridelí záznamu aktuálnu časovú značku.

Následne sa spustí funkcia „parse_new_data()“, ktorá je taktiež spúšťaná vložením nového riadka do tabuľky „data“. Funkcia napĺňa ďalšie dve tabuľky. „Mobile“

Tab. 3.4: Názvy stĺpcov, dátové typy a ich význam tabuľky „data“.

Názov stĺpca	**Dátový typ**	**Význam**
id	serial PK	poradové číslo záznamu
src	integer	zdrojová adresa
dst	integer	cieľová adresa
rssi	smallint	hodnota RSSI medzi uzlami
lqi	smallint	hodnota LQI medzi uzlami
timestamp	timestamp	časová značka

3.6 obsahuje záznamy získané medzi mobilným a kotevným uzlom. „Anchor“ 3.5 zase záznamy získané medzi kotevnými uzlami navzájom. Po vložení nového riadka do tabuľky „data“ si funkcia vyfiltruje všetky záznamy týkajúce sa danej dvojice uzlov a zoradí ich vzostupne podľa časovej značky. Vypočíta priemernú hodnotu a medián RSSI za definovanú dobu pre danú dvojicu. Údaje potom importuje do tabuľky „mobile“ alebo „anchor“ podľa zdrojovej a cieľovej adresy. Ak záznam existuje, prevedie sa jeho aktualizácia.

Tab. 3.5: Názvy stĺpcov, dátové typy a ich význam tabuľky „anchor“.

Názov stĺpca	**Dátový typ**	**Význam**
id	serial PK	poradové číslo záznamu
src	integer	zdrojová adresa
dst	integer	cieľová adresa
rssi_mean	integer	priemerná hodnota RSSI za užívateľom definovanú dobu
rssi_med	integer	medián RSSI za užívateľom definovanú dobu
lqi	integer	najnovšia prijatá hodnota LQI
distance	real	skutočná vzdialenosť medzi uzlami vypočítaná z údajov v tabuľke „nodes“
eta	real	útlmový činiteľ medzi dvojicou kotevných uzlov
timestamp	timestamp	časová značka

Po vložení nového riadka do tabuľky „anchor“ alebo „mobile“ sa pomocou funkcie „calc_real_distance()“ vypočíta skutočná vzdialenosť medzi všetkými dvojicami

uzlov. Pre výpočet sa využijú súradnice uzlov definované v tabuľke „nodes“. Výpočet samotný pozostáva z dvoch pytagorových viet, keďže sa uvažuje trojrozmerný priestor. V tabuľke „anchor“ bude táto hodnota použitá ako referenčná vzdialenosť pri výpočte vzdialenosti medzi mobilným a kotevnými uzlami. V tabuľke „mobile“ sa skutočná vzdialenosť využíva pre výpočet chyby odhadu vzdialenosti.

Tab. 3.6: Názvy stĺpcov, dátové typy a ich význam tabuľky „mobile“.

Názov stĺpca	**Dátový typ**	**Význam**
id	serial PK	poradové číslo záznamu
src	integer	zdrojová adresa
dst	integer	cieľová adresa
rssi_mean	integer	priemerná hodnota RSSI za užívateľom definovanú dobu
rssi_med	integer	medián RSSI za užívateľom definovanú dobu
lqi	integer	najnovšia prijatá hodnota LQI
distance	real	skutočná vzdialenosť medzi uzlami vypočítaná z údajov v tabuľke „nodes“
distance_0	real	vypočítaná vzdialenosť pomocou statických parametrov
distance_1	real	vypočítaná vzdialenosť pomocou algoritmu 1
distance_2	real	vypočítaná vzdialenosť pomocou algoritmu 2
distance_3	real	vypočítaná vzdialenosť pomocou algoritmu 3
abs_err_0	real	absolútna chyba odhadu vzdialenosti pre výpočet vzdialenosti pomocou statických parametrov
abs_err_1	real	absolútna chyba odhadu vzdialenosti pre výpočet vzdialenosti pomocou algoritmu 1
abs_err_2	real	absolútna chyba odhadu vzdialenosti pre výpočet vzdialenosti pomocou algoritmu 1
abs_err_3	real	absolútna chyba odhadu vzdialenosti pre výpočet vzdialenosti pomocou algoritmu 1
timestamp	timestamp	časová značka

Aktualizovanie hodnoty skutočnej vzdialenosti v tabuľke „anchor“ alebo zmena RSSI spustí funkciu „calc_eta()“. Tá podľa vzťahu 2.4 vypočíta hodnotu indexu stratovosti prostredia η. Tým pádom sú známe všetky referenčné parametre medzi kotevnými uzlami, potrebné k výpočtu vzdialenosti. Týmito parametrami sú

hodnota RSSI aktualizovaná pri importovaní nových dát, hodnota skutočnej vzdialenosti vypočítana po vytvorení nového riadka alebo aktualizovaní súradnic uzla a index stratovosti prostredia.

Po vložení nového riadka alebo po aktualizovaní hodnoty RSSI v tabuľke „mobile“ sa spustia funkcie „calc_est_distance_0()“ až „calc_est_distance_3()“ B.5, B.6, B.7 a B.8. Každá z nich počíta vzdialenosť podľa jedného zo scenárov popísaných v 3.1.1. Z tabuľky „anchor“ si načíta hodnotu RSSI, vzdialenosť medzi kotevným a referenčným kotevným uzlom a index stratovosti prostredia η. Tieto hodnoty slúžia ako referenčné. Nakoniec si z tabuľky „mobile“ načíta hodnotu RSSI a všetky parametre dosadí do vzťahu 2.2. Vypočítaná vzdialenosť sa aktualizuje v tabuľke.

Posledným krokom je výpočet absolútnej chyby pre každú z použitých metód výpočtu vzdialenosti. Túto úlohu zabezpečuje funkcia „calc_abs_error()“ B.9. Vždy po aktualizovaní vypočítanej vzdialenosti sa spustí výpočet absolútnej chyby.

3.4.2 Funkcie a spúšťače

Všetky funkcie a spúšťače, ktoré sú požívané v databáze, sú vytvorené v jazyku PL/pgSQL (Procedural Language/PostgreSQL) [13]. Sú integrované priamo v databázovom systéme. Vďaka tomu je možné jednotlivé spúšťače zreťaziť. Po tom ako sa vyskytne udalosť, na ktorú čaká spúšťač, sa vykoná definovaná funkcia. Tá môže vložiť nový riadok alebo aktualizovať hodnoty v tabuľke a zároveň tým spustiť ďalší spúšťač. K všetkým výpočtom teda dôjde vo veľmi rýchlom časovom slede. Taktiež sa zvýši výkonnosť a robustnosť systému, keďže sa k databáze nepristupuje z externej aplikácie pomocou query príkazov.

Tabuľka 3.7 obsahuje voľne dostupné funkcie pre PL/pgSQL [14]. Vyžívajú sa k aproximovaniu parametra RSSI. V tejto práci je pre aproximáciu využívaná štatisticky stredná hodnota a medián.

Tabuľka 3.8 obsahuje zoznam funkcii automaticky spúšťaných pomocou databázových spúšťačov. Komentované zdrojové kódy funkcií sú v prílohe B. Ako je možné

Tab. 3.7: Názvy a popis funkcií používaných databázou.

Názov funkcie	**Popis**
average(double precision[])	vracia priemernú hodnotu zadaného poľa
median(double precision[])	vracia medián zadaného poľa
array_index(double precision[])	vracia pole indexov prvkov zadaného poľa zoradeného vo vzostupnom poradí

si všimnúť, funkcie nemajú žiadné vstupné parametre. Je to dôsledok volania funkcií pomocou spúšťačov pri vhodných udalostiach.

Tab. 3.8: Názvy a popis funkcii spúšťaných udalosťami v databáze.

Názov funkcie	**Popis**
add_timestamp_now()	pridá časovú značku k záznamom importovaným do tabuľky „data“
parse_new_data()	napĺňa záznamy do tabuliek „mobile“ a „anchor“ z tabuľky „data“
calc_real_distance()	vypočíta skutočnú vzdialenosť medzi uzlami
calc_eta()	vypočíta index stratovosti prostredia medzi kotevnými uzlami
calc_est_distance_0()	vypočíta vzdialenosť medzi kotevnými a mobilnými uzlami pomocou statických parametrov
calc_est_distance_1()	vypočíta vzdialenosť medzi kotevnými a mobilnými uzlami pomocou algoritmu 1
calc_est_distance_2()	vypočíta vzdialenosť medzi kotevnými a mobilnými uzlami pomocou algoritmu 2
calc_est_distance_3()	vypočíta vzdialenosť medzi kotevnými a mobilnými uzlami pomocou algoritmu 3
calc_abs_error()	vypočíta absolútnu chybu medzi skutočnou a vypočítanou vzdialenosťou

Spúšťače v databáze slúžia na spustenie vytvorenej funkcie po alebo pred nastavenou udalosťou. Tieto udalosti sa delia na vloženie nového riadka, aktualizáciu ľubovoľného počtu polí v jednom alebo viacerých riadkoch a vymazanie riadka. Tabuľka 3.9 obsahuje zoznam spúšťačov použitých v tejto práci. Taktiež obsahuje popis udalosti, ktorú daný spúšťač očakáva. Ak nastane očakávana udalosť, spúšťač spustí

definovanú funkciu.

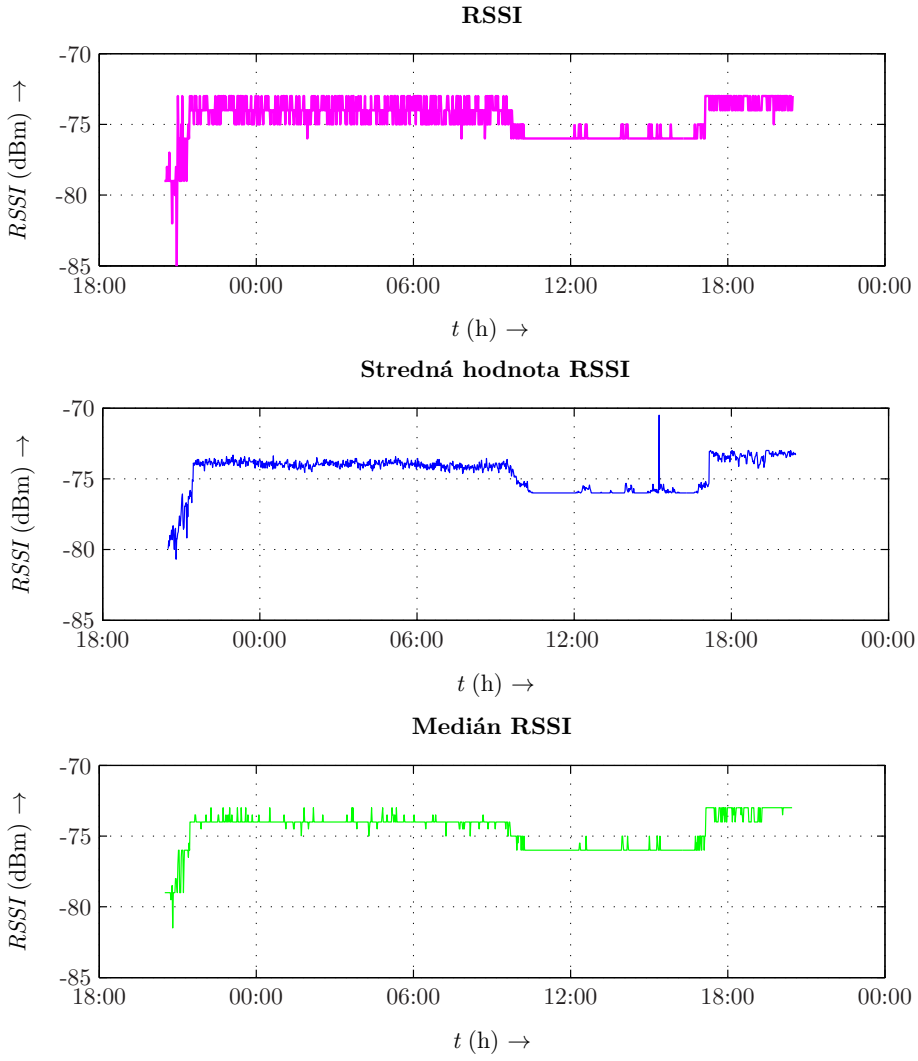
Tab. 3.9: Názvy a funkcia spúšťačov v databáze.

Názov funkcie	**Popis**
add_timestamp	spúšťa funkciu „add_timestamp_now()“ pri vložení nového riadka do tabuľky „data“
parse_new_data	spúšťa funkciu „parse_new_data()“ po vložení nového riadku do tabuľky „data“
update_real_distance	spúšťa funkciu „calc_real_distance()“ pri vložení riadka do tabuľky „mobile“ a „anchor“ alebo pri zmene súradníc uzla v tabuľke „nodes“
update_eta	spúšťa funkciu „calc_eta()“ pred vložením nového riadka alebo aktualizovaní polí „distance“ či „rssi_mean“ tabuľky „anchor“
update_est_distance_0 update_est_distance_1 update_est_distance_2 update_est_distance_3	spúšťajú funkcie „calc_est_distance_x()“ po vložení nového riadku do tabuliek „anchor“ a „mobile“ alebo pri aktualizácii dát v stĺpci „eta“ tabuľky „anchor“ a „rssi_mean“ tabuľky „mobile“
update_abs_error	spustí funkciu „calc_abs_error()“ pred vložením nového riadka alebo aktualizovaním ktoréhokoľvek z polí vzdialenosti tabuľky „mobile“

4 VYKONANÉ MERANIA

4.1 Meranie parametra RSSI

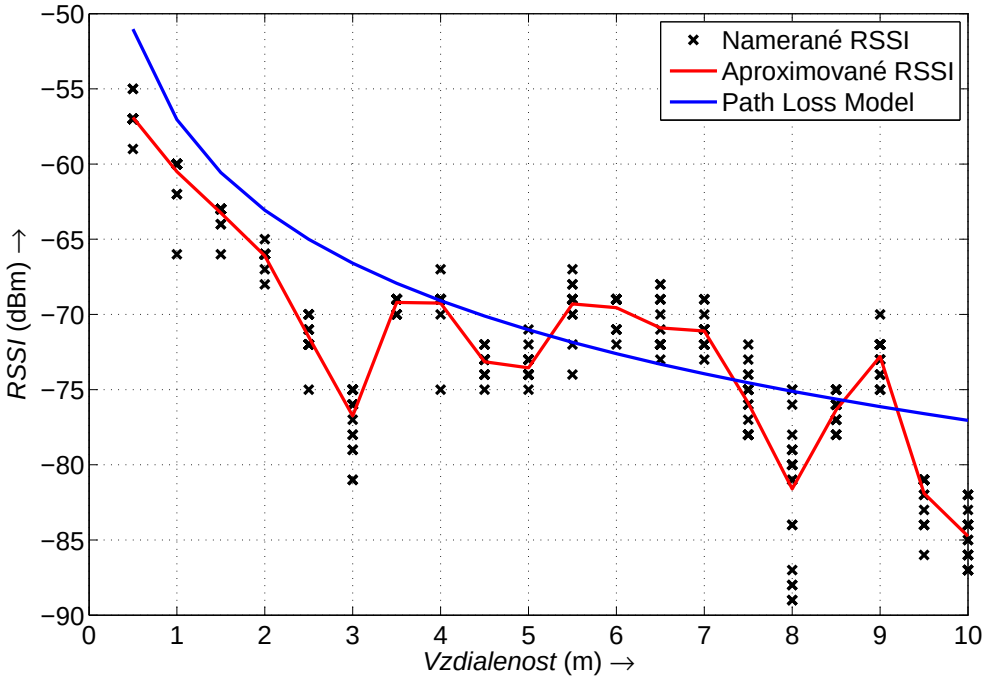
Pri tomto meraní sa počas dňa zachytávali vzorky RSSI každých 5 s. Následne sa z nameraných hodnôt každých 15 s vypočítala stredná hodnota a medián parametru RSSI.



Obr. 4.1: Zmena parametra RSSI počas dňa.

Zo závislosti na obrázku 4.1 vyplýva, že použitie strednej hodnoty a mediánu RSSI dosahuje veľmi podobných výsledkov. V ďalších meraniach je preto používaná stredná hodnota parametra RSSI, keďže je výpočetne menej náročná.

Závislosť 4.2 je výsledkom merania dvadsiatich hodnôt RSSI na každej zo vzdialenosti 0,5 m až 10 m. Aproximované hodnoty RSSI sú porovnané s ideálnym priebehom logaritmického stratového modelu.

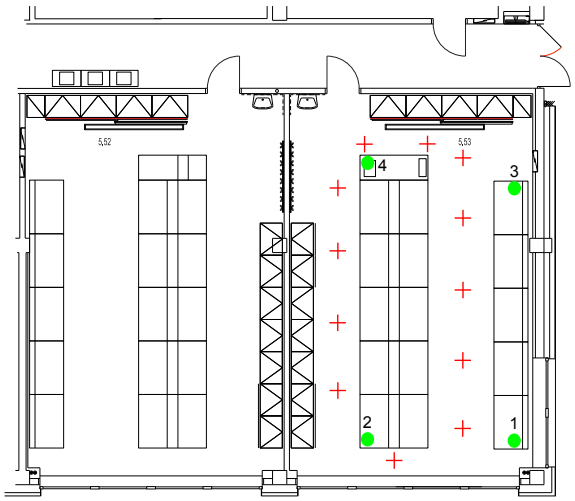


Obr. 4.2: Porovnanie ideálneho priebehu logaritmického stratového modelu a reálnych nameraných hodnôt.

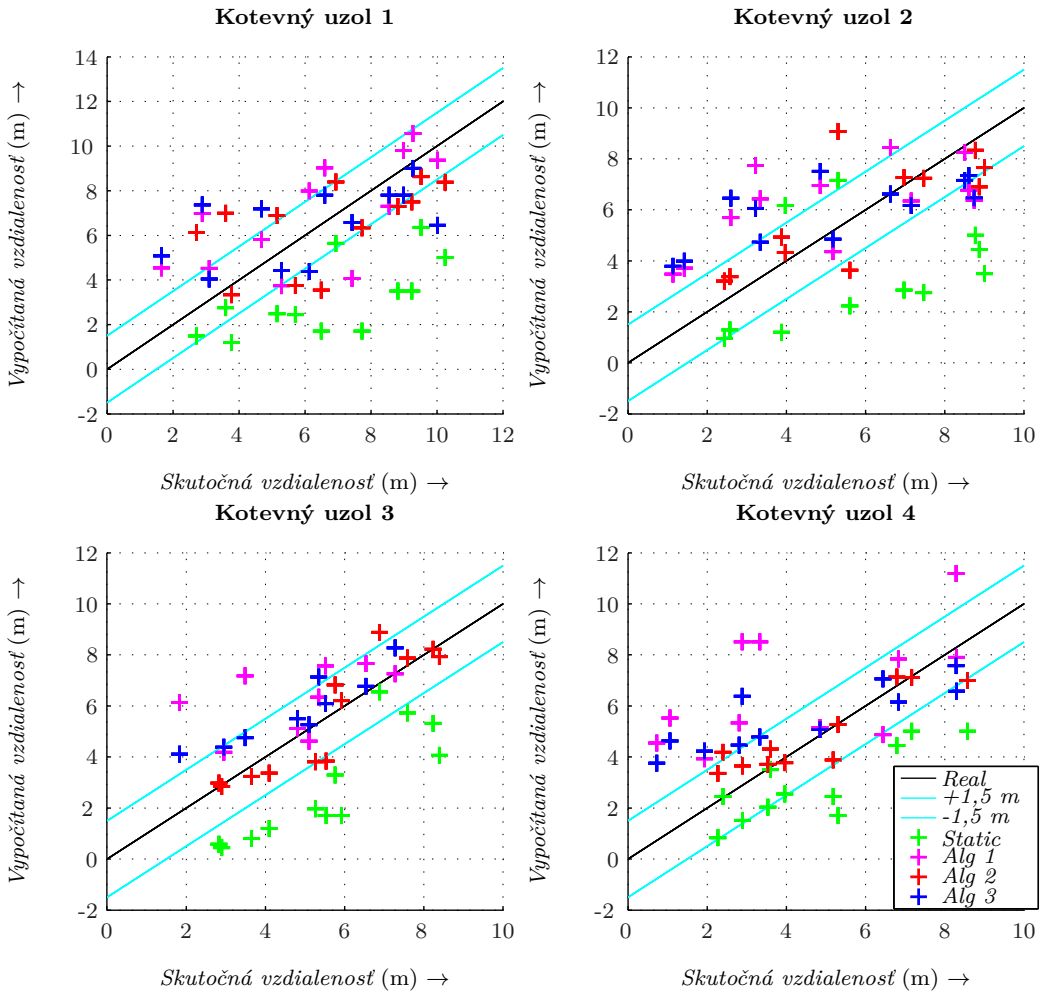
4.2 Porovnanie použitých metód odhadu vzdialenosti

4.2.1 Meranie v rámci laboratória

Nasledujúce meranie je vykonané v priestoroch laboratória. Pri meraní boli kotevné uzly umiestnené v prostredí podľa obrázka 4.3. Na pozíciach jedna až štyri sú kotevné uzly umiestnene vo dojiciach. Jeden uzol je vo výške 0,75 m a druhý vo výške 2,9 m. Mobilný uzol bol postupne umiestnený na vyznačených pozíciach. Meranie prebiehalo podľa všetkých scenárov uvedených v 3.1.1 súčasne.



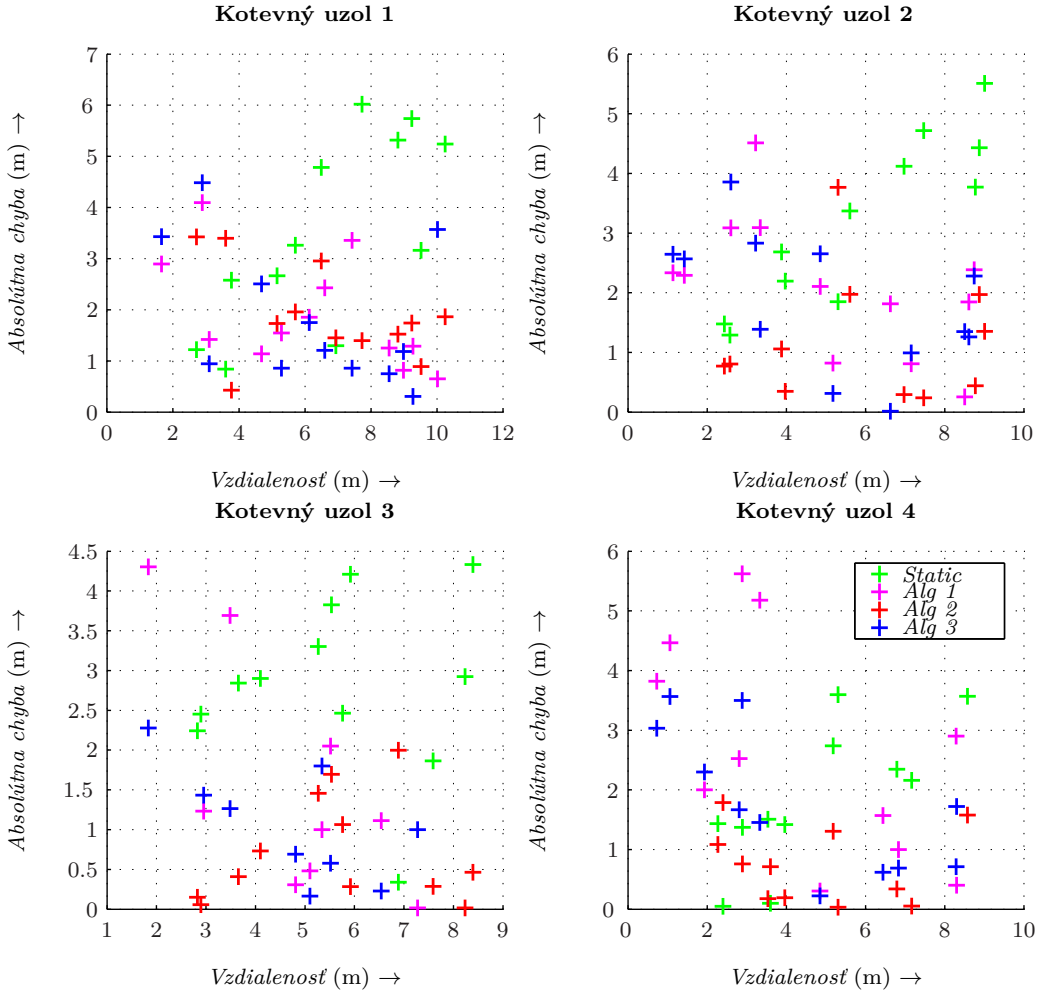
Obr. 4.3: Pôdorys meraného prostredia s vyznačenými meranými pozíciami.



Obr. 4.4: Závislosť skutočnej a vypočítanej vzdialenosti merania v priestoroch laboratória.

Obrázok 4.4 zobrazuje závislosť vypočítanej a skutočnej vzdialenosti medzi mobilným a kotevnými uzlami.

Priamka „real“ znázorňuje nulovú chybu výpočtu vzdialenosti. Značky zelenej farby označujú vypočítané hodnoty pomocou statických parametrov pri použití kotevných uzlov vo výške 2,9 m. Purpurové značky znázorňujú vypočítané hodnoty pomocou prvého scenára, kde sa používajú iba kotevné uzly vo výške 0,75 m. Červené značky označujú vypočítané hodnoty druhého scenára, kde sa využívajú iba kotevné uzly vo výške 2,9 m. Modré značky vyznačujú hodnoty vypočítané podľa štvrtého scenára s využitím všetkých kotevných uzlov.

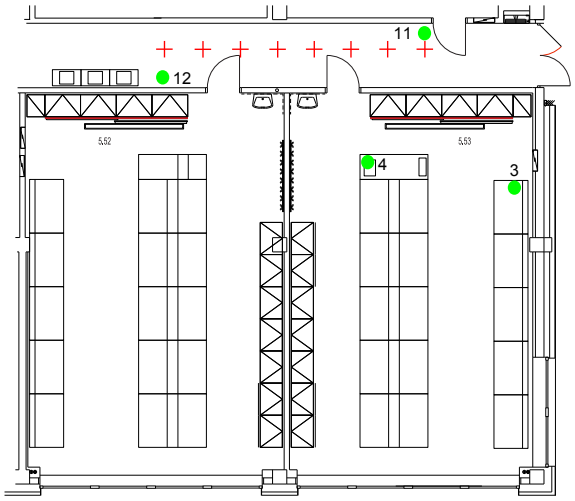


Obr. 4.5: Absolútna chyba vypočítanej vzdialenosti merania v priestoroch laboratória.

Obrázok 4.5 zobrazuje absolútnu chybu jednotlivých meraní. Z obrázkov je možné pozorovať, že najväčšie chyby určenia vzdialenosti boli namerané pri použití statických parametrov rádiového prostredia. Naopak najväčšej presnosti bolo dosiahnuté pri použití dynamického nastavovania parametrov prostredia kotevnými uzlami vo výške 2,9 m podľa tretieho scenára. Merania podľa druhého a štvrtého scenára vykazujú zvýšenú chybu pre vzdialenosti menšie ako 3 m.

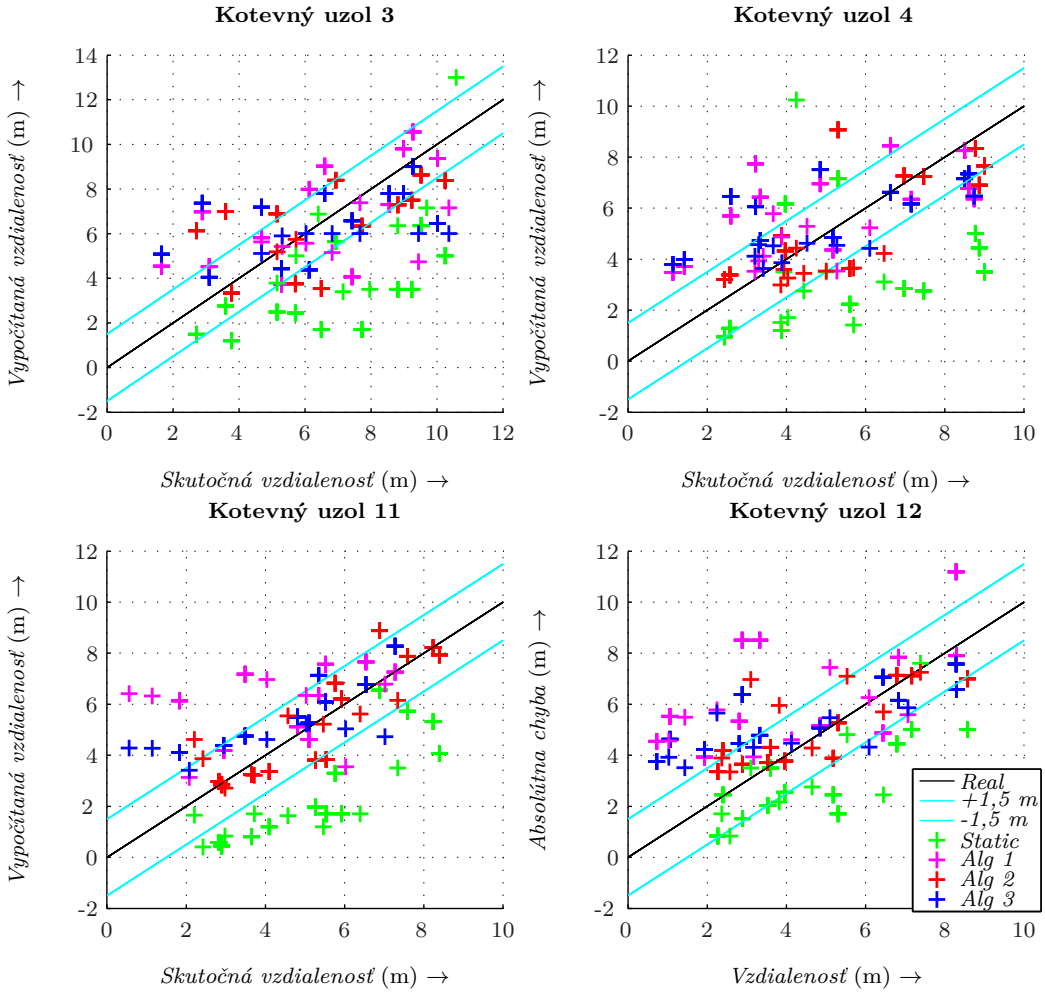
4.2.2 Meranie v rámci dvoch miestností

Druhé meranie je vykonané v priestoroch laboratória a priľahlej chodby. Cieľom merania bolo zistiť ako sa zmení odhadovaná vzdialenosť pri vložení prekážky medzi kotevné uzly. V tomto prípade je medzi kotevnými uzlami 3, 4 a 11, 12 prekážka v podobe steny 4.6. Merania boli postupne vykonané s umiestnením mobilného uzla vo vyznačených pozíciach. Opäť prebehli merania všetkých scenárov súčasne.

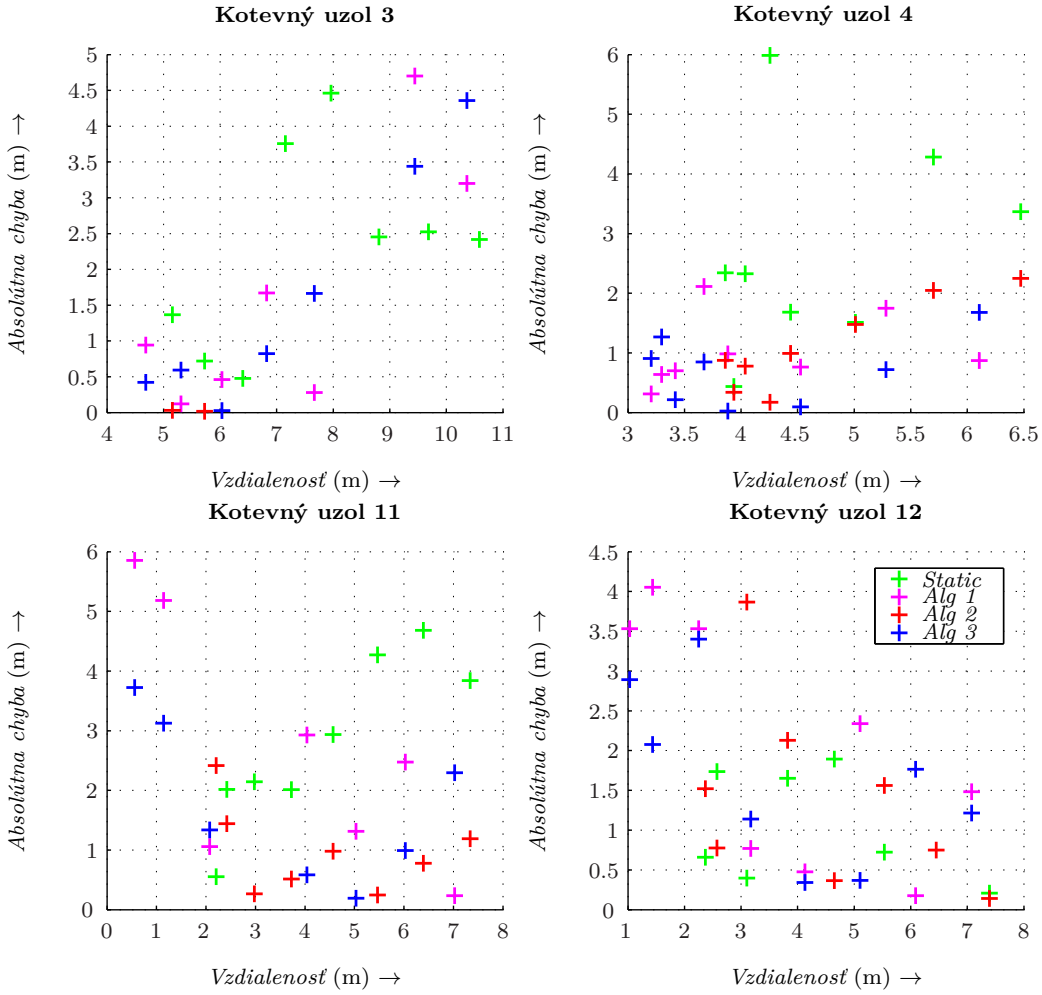


Obr. 4.6: Pôdorys meraného prostredia s vyznačenými meranými pozíciami.

Obrázok 4.7 aj pri tomto meraní popisuje závislosť vypočítanej vzdialenosti na skutočnej vzdialenosti medzi mobilným uzlom a kotevnými uzlami. Najpresnejšie výsledky výpočtu vzdialenosti opäť vykazovala metóda použitá v treťom scenári. Najväčšie odchylky od skutočnej vzdialenosti dosahuje statická metóda. Najzreteľnejšie je to vidieť pre výpočet vzdialenosti medzi mobilným uzlom a kotevnými uzlami 3 a 4 umiestnenými na druhej strane steny 4.8.



Obr. 4.7: Závislosť skutočnej a vypočítanej vzdialenosti merania v rámci dvoch miestností.



Obr. 4.8: Absolútna chyba vypočítanej vzdialenosti merania v rámci dvoch miestností.

4.3 Zhodnotenie merania

Z vykonaných meraní bolo zistené, že najlepšie výsledky podáva výpočet vzdialenosti medzi bezdrôtovými uzlami podľa tretieho scenára, kde sú kotevné uzly umiestnené vo výške 2,9 m.

Určovanie vzdialenosti pomocou staticky nastavených parametrov rádiového modelu dosahuje pomerne dobré výsledky, pokiaľ sa meranie realizuje v prostredí, v ktorom boli parametre kalibrované. Po zmene meraného prostredia, ako napríklad premiestnenie mobilného uzla za stenu, chyba odhadu vzdialenosti rýchlo narastá.

Pri využití kotevných uzlov vo výške 0,75 m v scenári dva a štyri bola zvýšená chybovosť odhadu vzdialenosti kratšej ako 3 m. Tento nedostatok by sa ďalej mohol riešiť úpravov algoritmov tak, aby boli tieto vzdialenosti odfiltrované. V treťom meranom scenári tento jav nenastával, keďže boli využité kotevné uzly vo výške 2,9 m a mobilný uzol bol umiestnený vo výške 1 m. Najmenšia možná vzdialenosť medzi nimi tak mohla byť 1,9 m.

5 ZÁVER

Prvým bodom zadania bolo naštudovanie problematiky určovania vzdialenosti medzi jednotkami v bezdrôtových sieťach. Tento bod je obsiahnutý v úvode práce, kde je popísané rozdelenie a princíp metód používaných k určeniu vzdialenosti. Nasleduje analýza rádiového modelu a parametrov rádiového prostredia. Ďalej sa práca zaoberá návrhom a realizáciou systému umožňujúcemu určovanie vzdialenosti medzi bezdrôtovými jednotkami v reálnom čase. Pre vytvorenie experimentálnej siete, v ktorej prebiehalo určovanie vzdialenosti slúžili uzly deRFnode [6]. Ako brána medzi WSN a internetom slúžil embedded systém ALIX [9]. Slúžil k preposielaniu dát z WSN do databázy. K uloženiu dát bol využitý databázový systém PostgreSQL [12]. Výpočetné algoritmy boli vytvorené pomocou jazyka PL/pgSQL a boli implementované priamo do databázy. Taktiež bolo vytvorené webové rozhranie pre jednoduchšie zadávanie dát pre počiatočnú konfiguráciu. V závere práce sú vykonané merania a porovnanie štyroch scenárov určenia vzdialenosti medzi jednotkami.

LITERATÚRA

[1] ŞAHINOĞLU, Zafer; GEZICI, Sinan; GÜVENÇ, Ismail. *Ultra-wideband positioning systems: theoretical limits, ranging algorithms, and protocols*. New York: Cambridge University Press, 2008, xi, 269 p. ISBN 978-052-1873-093.

[2] AJAY, Malik. *RTLS For Dummies*. Indianapolis, Indiana: Wiley Publishing, Inc., © 2009. ISBN 978-0-470-39868-5. Dostupné z URL: <http://cias.rit.edu/~nmtp/nmtp/teamkittycat/RTLSforDummies.pdf>.

[3] ATMEL CORPORATION. *ATmega128RFA1 Datasheet*. [online]. 2013. [cit. 2013-12-22]. Dostupné z URL: <http://www.atmel.com/Images/doc8266.pdf>.

[4] XU, Jiuqiang. *Distance Measurement Model Based on RSSI in WSN. Wireless Sensor Network*. [online]. 2010, vol. 02, issue 08, s. 606-611 [cit. 2014-05-29]. DOI: 10.4236/wsn.2010.28072. Dostupné z: <http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/wsn.2010.28072>.

[5] BOTTA, Miroslav; ŠIMEK, Milan. *Optimalizácia odhadu vzdialenosti v bezdrôtovej ad-hoc senzorovej sieti*. [online]. [cit. 2014-05-29]. Dostupné z: <http://www.elektrorevue.cz/cz/clanky/komunikacni-technologie/0/optimalizacia-odhadu-vzdialenosti-v-bezdrotovej-ad-hoc-senzorovej-sieti/>.

[6] Dresden Electronik. *deRFnode* [online]. 2013. [cit. 2014-05-28]. Dostupné z: <http://www.dresden-elektronik.de/funktechnik/produkte/boards-und-kits/development-boards/derfnode/>.

[7] Dresden Electronik. *Evaluation modules deRFmega128* [online]. 2013. [cit. 2014-05-28]. Dostupné z: <http://www.dresden-elektronik.de/funktechnik/products/radio-modules/avr-single-chip-modules/description/>.

[8] ATMEL CORPORATION. *SerialNet* [online]. 2012. [cit. 2014-05-28]. Dostupné z: <http://www.atmel.com/Images/doc8389.pdf>.

[9] ALIX PC Engines. [online]. 2014. [cit. 2014-05-28]. Dostupné z: <http://www.pcengines.ch/alix3d2.htm>.

[10] Voyage. [online]. [cit. 2014-05-29]. Dostupné z: <http://linux.voyage.hk/>.

[11] Dresden Electronik. *deRFnode linux driver* [online]. [cit. 2014-05-29]. Dostupné z: <http://www.dresden-elektronik.de/shop/cont13.html#12001>.

[12] PostgreSQL. [online]. [cit. 2014-05-29]. Dostupné z: <http://www.postgresql.org/>.

[13] PostgreSQL. *PL/pgSQL - SQL Procedural Language*. [online]. [cit. 2014-05-29]. Dostupné z: http://www.postgresql.org/docs/9.1/static/plpgsql.html

[14] PGNUMERICS. [online]. [cit. 2014-05-29]. Dostupné z: <http://pgnumerics.projects.pgfoundry.org/>.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

AoA	Angle of Arrival – Uhol príjmu signálu
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance – Metóda s viacnásobným prístupom k médiu s naslúchaním nosnej
FFD	Full-Funkcion Device – Zariadenie s plnou sadou funkcii
RFD	Reduced-Funkcion Device – Zariadenie s redukovanou sadou funkcii
RSSI	Received Signal Strength Indication – Indikátor energie prijatého signálu
RTT	Round Trip Time – Čas od odoslania paketu po príjem potvrdenia
ToA	Time of Arrival – Čas príjmu
WSN	Wireless Sensor Network – Bezdrôtová senzorová sieť
$RSSI_{\mathrm{BASE_VAL}}$	Minimálna citlivosť prijímača [dBm]
$RSSI_{\mathrm{REG}}$	Hodnota odčítaná z registru [dBm]

ZOZNAM PRÍLOH

A ZDROJOVÉ KODY - PYTHON

A.1 SerialNetSetup.py

```
__author__ = 'Juraj Popovec'

import time
import serial
import sys
############## Serial parameters ###################
ser = serial.Serial(
    port = '/dev/ttyUSB1',
    baudrate = 38400,
    parity = serial.PARITY_NONE,
    stopbits= serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    )
############## SerialNet setup ####################
at =        "AT"                # Set "AT" for local or "ATR<address>" for remote
    commands. Example: "ATR25"
atrpass =   "0"                 # Set password for remote commands

instructions = [
##-----------Networking parameters-----------------
#!!!Uncomment if you want to change network settings
        "+WAUTONET=0",
        "+WLEAVE",
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        "+WPANID=FEEC",         #Extended PAN ID
        "+WCHMASK=1000000",     #Channel mask
      # "+WCHPAGE=",            #Channel page
        "+WROLE=1",             #Node role
        "+GSN=7F16",            #Device extended address
        "+WSRC=7F16",           #Node short address
        "+WNWKPANID=FEEC",      #Short (network) PAN ID
        "+WAUTONET=1",          #Automatic networking
##---------------Power Management-----------------
      # "+WPWR=",               #End device sleep parameters
        "+WTXPWR=3",            #TX power level

##---------------Remote management----------------
       "+WPASSWORD 0"           #Set a password. Example: "WPASSWORD 1234"
##---------------Host interface commands-----------
       "X 2"                    #Result code selection
##-----------------Extended commands----------------
```

```
     # "+LED=0",                 #Hex number 0-7 containing the three led status
       "+HELL=5",                #Programs hello message broadcasting at a given
           period in seconds
       "+RSS FFF0",              #Enables/disables messages forwarding to sink in
           format: <id>-<length>-<LQI>-<RSSI>-<timestamp>
##-----------------Generic control-----------------
     # "+WACALIBRATE=60",        #Configure periodic internal clock calibration
       "Z"                       #Warm reset
    ]
################### Functions ######################
def serial_read():
    """
    :rtype : list
    """
    lines = []
    while True:
        lines.append(ser.readline())
        timeout = time.time() + 0.1
        while not ser.inWaiting() and timeout > time.time():
            pass
        if not ser.inWaiting():
            break
    return lines
######################## Main ######################
errors = []

try:
    ser.close()
    ser.open()
    for x in range(0,3):
        ser.write("AT\r")
        ans = serial_read()
        for x in ans:
            pass
        if x == 'OK\r\n':
            print "Connection opened!\n\r"
            break
        else: print ("Error! Cannot connect to serial.")
except serial.SerialException as e:
     print ("Error! Cannot connect to serial: ")
     print str(e)
     sys.exit()

for x in instructions:
    ser.write(at)
    if at != "AT":
       ser.write("," + atrpass + ",")
```

```
    ser.write(x + "\r")
    lines = serial_read()
    for y in lines:
        print y
    if y != 'OK\r\n':
        errors.append(lines[0])

if not errors:
    print "------------------------------\r\nCommands processed successfully!"
else:
    print "------------------------------\r\nERROR! Can't process commands:\r\n"
    for x in errors:
        print x
    print "ERROR!ERROR!ERROR!ERROR!ERROR!\r\n------------------------------\r\n"
```

A.2 RelayAgent.py

```
__author__ = 'Juraj Popovec'

import serial
import time
import datetime
import sys
import re
import psycopg2
import DBconn
import thread
import signal


############### Serial parameters ###################
ser = serial.Serial(
    port = '/dev/ttyUSB1',
    baudrate = 38400,
    parity = serial.PARITY_NONE,
    stopbits= serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout = 0,
    )
#################### Variables ####################
timeToStoreData = 30          #time in seconds for records in 'data' to be stored
timeToStoreMobile = 15        #time in secondss for records in 'mobile' to be stored
timeToStoreAnchor = 30        #time in seconds for records in 'anchor' to be stored
delTimer = 10                 #time in seconds for script to clear old records
```

```
serialReadInterval = 0.2     #time in seconds for script to wait between serial
    reads (CPU utilization)
######### Global variables initialization ##########
fifo = []
#################### Functions ####################
def init():
    print "\nRELAY AGENT RUNNING ..."
##### init serial #####
    try:
        ser.close()
        ser.open()
    except serial.SerialException as e:
        print ("Error! Cannot connect to serial: ")
        print str(e)
        sys.exit()

##### init DB #####
    DBconn.send_query("CREATE TABLE IF NOT EXISTS data(id serial PRIMARY KEY, src
        int, dst int, rssi smallint, lqi smallint, timestamp TIMESTAMP)")

def serial_read():
    """
    :rtype : list
    """
    counter = 0
    lines = []
    while True:
##### read all (99999) bytes from serial #####
        data = ser.read(99999)
##### if
        if len(data) > 0:
            tmp = re.split('\r\n',data)
            lines = lines + tmp
            counter = counter + 1
            time.sleep(0.1)
            if counter >= (serialReadInterval*10):
                while True:
                    try:
                        lines.remove('')
                    except:
                        break
                return lines
                lines = []
        else:
            time.sleep(0.2)
```

```
def add_to_batch(data):
    try:
        newbatch = []
        for x in data:
            tmp = re.split(' |,|:|-',x)
##### check if string starts with DATA and is not bcast #####
            if tmp[0] == 'DATA' and tmp[2] != '1':
                rssi = int(tmp[7],16)
##### change RSSI from signed byte to int #####
                if rssi > 127:
                    mask = 255
                    rssi = ((rssi ^ mask) + 1) * (-1)
##### add new row to batch #####
                row = (int(tmp[4],16), int(tmp[1],16), int(rssi), int(tmp[6],16))
                newbatch.append(row)
    except:
        #print "Error: Line corrupted"
        pass

    finally:
##### add to fifo whole batch or until error #####
        fifo.append(newbatch)
        return

def run_batch_update():
    try:
        con = psycopg2.connect(database="localizationdb", user="dbuser1", password=
            "3f:X89rh(n", host="147.229.149.80", port="5433")
        cur = con.cursor()
##### cut oldest batch from fifo to proceed #####
        batch = fifo.pop(0)
        if not batch:
            return
        print "%s\tINSERTING BATCH WITH %d ROWS... %d BATCHES REMAINING" % (str(
            datetime.datetime.now()), len(batch), len(fifo))
##### print all data in batch #####
        #for x in batch:
        #    print ("src = %d  dst = %d  rssi = %d  lqi = %d" % (x[0], x[1], x[2],
            x[3], ))
##### create string from all lines in batch #####
        args_str = ", ".join(cur.mogrify("(%s, %s, %s, %s)", x)for x in batch)
##### run INSERT query with all lines in batch (insert all lines at once) #####
        cur.execute("INSERT INTO data (src, dst, rssi, lqi) VALUES" + args_str)
        con.commit()

    except psycopg2.DatabaseError, e:
        if con:
```

```
            con.rollback()
        print 'Error %s' % e
        sys.exit(1)

    finally:
        if con:
            con.close()
        return

def thread_clear_db():
    while True:
        DBconn.send_query("DELETE FROM data WHERE timestamp < CURRENT_TIMESTAMP -
            interval '%d' second;" %  timeToStoreData)
        DBconn.send_query("DELETE FROM mobile WHERE timestamp < CURRENT_TIMESTAMP -
             interval '%d' second;" %  timeToStoreMobile)
        DBconn.send_query("DELETE FROM anchor WHERE timestamp < CURRENT_TIMESTAMP -
             interval '%d' second;" %  timeToStoreAnchor)

        print "%s\tCLEANING DATABASE" % str(datetime.datetime.now())
        time.sleep(delTimer)

def thread_fifo():
    while True:
##### import data if fifo is not empty #####
        if fifo:
            run_batch_update()
        else:
            time.sleep(0.1)

def exit_gracefully(signal, frame):
    ser.close()
    print "\nEXIT RELAY AGENT\n"
    sys.exit(0)
######################## Main ######################
init()
signal.signal(signal.SIGINT, exit_gracefully)
thread.start_new_thread(thread_fifo,())
thread.start_new_thread(thread_clear_db,())
while True:
    lines = serial_read()
    #print lines
    add_to_batch(lines)
```

B ZDROJOVÉ KÓDY - PLPQSQL

B.1 add_timestamp_now()

```
-- Author: Juraj Popovec
-- Function: add_timestamp_now()

CREATE OR REPLACE FUNCTION add_timestamp_now()
  RETURNS trigger AS
$BODY$
BEGIN
  NEW.timestamp := now();
  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION add_timestamp_now()
  OWNER TO root;
```

B.2 parse_new_data()

```
-- author: Juraj Popovec
-- Function: parse_new_data()

CREATE OR REPLACE FUNCTION parse_new_data()
  RETURNS trigger AS
$BODY$
DECLARE
  least_mobile_src mobile.src%TYPE := 65280;
  rssi_mean_val mobile.rssi_mean%TYPE;
  rssi_med_val mobile.rssi_med%TYPE := 0;
  rssi_array double precision[] := '{}';

BEGIN
  IF NEW.src >= least_mobile_src
  THEN
    -- get rssi from 'data' for last 5 seconds - for mobile nodes
    rssi_array = array(SELECT rssi FROM data WHERE src = NEW.src AND dst = NEW.dst
        AND timestamp > CURRENT_TIMESTAMP - interval '5' second);
```

```
  ELSE
    -- get rssi from 'data' for last 15 seconds - for anchor nodes
    rssi_array = array(SELECT rssi FROM data WHERE src = NEW.src AND dst = NEW.dst
        AND timestamp > CURRENT_TIMESTAMP - interval '15' second);
  END IF;
  IF rssi_array IS NULL
  THEN
    RETURN NEW;
  END IF;
  --calculate mean for rssi
  rssi_mean_val = average(rssi_array);
  --calculate median for rssi
  --rssi_med_val = median(rssi_array);

  --check if exist nodes pair in mobile
  PERFORM id FROM mobile WHERE src = NEW.src AND dst = NEW.dst;
  IF FOUND THEN
    --update data if exist
    UPDATE mobile SET rssi_mean = rssi_mean_val, rssi_med = rssi_med_val, lqi = NEW
        .lqi, timestamp = NEW.timestamp WHERE src = NEW.src AND dst = NEW.dst;
  ELSE
    --insert if not exist to table mobile (address >= 65280)
    IF NEW.src >= least_mobile_src THEN
      INSERT INTO mobile (src, dst, rssi_mean, rssi_med, lqi, timestamp) VALUES (
          NEW.src, NEW.dst, rssi_mean_val, rssi_med_val, NEW.lqi, NEW.timestamp);
    END IF;
  END IF;
  --check if exist nodes pair in anchor
  PERFORM id FROM anchor WHERE src = NEW.src AND dst = NEW.dst;
  IF FOUND THEN
    --update data if exist
    UPDATE anchor SET rssi_mean = rssi_mean_val, rssi_med = rssi_med_val, lqi = NEW
        .lqi, timestamp = NEW.timestamp WHERE src = NEW.src AND dst = NEW.dst;
  ELSE
    --insert if not exist to table anchor (address < 65280)
    IF NEW.src < least_mobile_src THEN
      INSERT INTO anchor (src, dst, rssi_mean, rssi_med, lqi, timestamp) VALUES (
          NEW.src, NEW.dst, rssi_mean_val, rssi_med_val, NEW.lqi, NEW.timestamp);
    END IF;
  END IF;
  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION parse_new_data()
  OWNER TO root;
```

B.3 calc_real_distance()

```
-- author: Juraj Popovec
-- Function: calc_real_distance()

CREATE OR REPLACE FUNCTION calc_real_distance()
  RETURNS trigger AS
$BODY$
DECLARE
  xpos1 nodes.xpos%TYPE;
  ypos1 nodes.ypos%TYPE;
  zpos1 nodes.zpos%TYPE;

  xpos2 nodes.xpos%TYPE;
  ypos2 nodes.ypos%TYPE;
  zpos2 nodes.zpos%TYPE;

  id anchor.id%TYPE;
  src anchor.src%TYPE;
  dst anchor.dst%TYPE;

  distance anchor.distance%TYPE;
BEGIN
  --for every pair of nodes in from table anchor get addresses
  FOR id, src, dst IN SELECT * FROM anchor
  LOOP
    --get position of pair of nodes
    EXECUTE 'SELECT xpos, ypos, zpos FROM nodes WHERE src = $$'||src||'$$' INTO
        xpos1, ypos1, zpos1;
    EXECUTE 'SELECT xpos, ypos, zpos FROM nodes WHERE src = $$'||dst||'$$' INTO
        xpos2, ypos2, zpos2;
    IF xpos1 IS NOT NULL AND ypos1 IS NOT NULL AND zpos1 IS NOT NULL AND xpos2 IS
        NOT NULL AND ypos2 IS NOT NULL AND zpos2 IS NOT NULL
    THEN
      --calculate distance
      distance = SQRT(SQRT(ABS(xpos2 - xpos1)^2 + ABS(ypos2 - ypos1)^2)^2 + ABS(
          zpos2 - zpos1)^2);
    --update real distance in table anchor
    EXECUTE 'UPDATE anchor SET distance = $$'||distance||'$$ WHERE src = $$'||src||
        '$$ AND dst = $$'||dst||'$$';
    END IF;
  END LOOP;
  --for every pair of nodes in from table mobile get addresses
  FOR id, src, dst IN SELECT * FROM mobile
  LOOP
    --get position of pair of nodes
```

```
    EXECUTE 'SELECT xpos, ypos, zpos FROM nodes WHERE src = $$'||src||'$$' INTO
        xpos1, ypos1, zpos1;
    EXECUTE 'SELECT xpos, ypos, zpos FROM nodes WHERE src = $$'||dst||'$$' INTO
        xpos2, ypos2, zpos2;
    IF xpos1 IS NOT NULL AND ypos1 IS NOT NULL AND zpos1 IS NOT NULL AND xpos2 IS
        NOT NULL AND ypos2 IS NOT NULL AND zpos2 IS NOT NULL
    THEN
      --calculate distance
      distance = SQRT(SQRT(ABS(xpos2 - xpos1)^2 + ABS(ypos2 - ypos1)^2)^2 + ABS(
          zpos2 - zpos1)^2);
    --update real distance in table mobile
    EXECUTE 'UPDATE mobile SET distance = $$'||distance||'$$ WHERE src = $$'||src||
        '$$ AND dst = $$'||dst||'$$';
    END IF;
  END LOOP;
  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION calc_real_distance()
  OWNER TO root;
```

B.4 calc_eta()

```
-- author: Juraj Popovec
-- Function: calc_eta()

CREATE OR REPLACE FUNCTION calc_eta()
  RETURNS trigger AS
$BODY$
DECLARE
  Pt nodes.txpwr%TYPE;
BEGIN
  -- get tx power of node from table nodes
  EXECUTE 'SELECT txpwr FROM nodes WHERE src = $$'||NEW.src||'$$' INTO Pt;
  IF Pt IS NOT NULL AND NEW.src IS NOT NULL AND NEW.rssi_med IS NOT NULL AND NEW.
      distance IS NOT NULL
  THEN
    --eta = [(RSSIr/10) - log(Pt)] / -log(d)
    NEW.eta = ((NEW.rssi_med / 10) - LOG(Pt)) / ((-1) * LOG(NEW.distance));
  END IF;
  RETURN NEW;
END;
```

```
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION calc_eta()
  OWNER TO root;
```

B.5 calc_est_distance_0()

```
-- author: Juraj Popovec
-- Function: calc_est_distance_0()

CREATE OR REPLACE FUNCTION calc_est_distance_0()
  RETURNS trigger AS
$BODY$
DECLARE
  id mobile.id%TYPE;
  src mobile.src%TYPE;
  dst mobile.dst%TYPE;

  -- static parameters
  rssi0 mobile.rssi_med%TYPE := -56.5;
  d0 mobile.distance%TYPE := 1;
  eta anchor.eta%TYPE := 1.93;

  distance_0 mobile.distance_1%TYPE;
  rssi mobile.rssi_med%TYPE;
BEGIN
  -- for every row in table mobile
  FOR id, src, dst IN SELECT * FROM mobile
  LOOP
    -- if dst address is more than 32512 (anchor 2.9m)
    IF dst > 32512
    THEN
      -- get rssi
      EXECUTE 'SELECT mobile.rssi_mean FROM mobile WHERE mobile.src = $$'||src||'$$
           AND mobile.dst = $$'||dst||'$$' INTO rssi;
      IF rssi IS NOT NULL AND rssi > -80
      THEN
        -- calculate distance for rssi more than -80 dBm
        distance_0 = d0 * (10 ^ ((rssi0 - rssi) / (10 * eta)));
        EXECUTE 'UPDATE mobile SET distance_0 = $$'||distance_0||'$$ WHERE src = $$
            '||src||'$$ AND dst = $$'||dst||'$$';
      ELSE
        -- if not rssi or rssi <= -80 update distance to null (not relevant)
```

```
        EXECUTE 'UPDATE mobile SET distance_0 = null WHERE src = $$'||src||'$$ AND
            dst = $$'||dst||'$$';
      END IF;
    END IF;
  END LOOP;
  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION calc_est_distance_0()
  OWNER TO root;
```

B.6 calc_est_distance_1()

```
-- author: Juraj Popovec
-- Function: calc_est_distance_1()

CREATE OR REPLACE FUNCTION calc_est_distance_1()
  RETURNS trigger AS
$BODY$
DECLARE
  id mobile.id%TYPE;
  src mobile.src%TYPE;
  dst mobile.dst%TYPE;

  rssi0 mobile.rssi_med%TYPE;
  distance0 mobile.distance%TYPE;
  distance_1 mobile.distance_1%TYPE;
  eta anchor.eta%TYPE;
  rssi mobile.rssi_med%TYPE;
BEGIN
  -- for all rows in mobile
  FOR id, src, dst IN SELECT * FROM mobile
  LOOP
    -- if dst address is <= 32512 (anchor 0.75m)
    IF dst <= 32512
    THEN
      -- algorithm 1: get rssi between mobile and anchor (0.75m) + parameters
          between anchor and referenced anchor (0.75m)(closest [highest rssi] to
          mobile, but not the same)
      EXECUTE 'SELECT anchor.rssi_mean, anchor.distance, anchor.eta, mobile.
          rssi_mean FROM anchor INNER JOIN mobile ON anchor.src = mobile.dst
```

```
      WHERE mobile.src = $$'||src||'$$ AND mobile.dst = $$'||dst||'$$ AND anchor.
          dst = (
      SELECT mobile.dst FROM mobile WHERE src = $$'||src||'$$ AND dst <= 32512 AND
          dst != $$'||dst||'$$ ORDER BY mobile.rssi_mean DESC LIMIT 1)' INTO rssi0,
           distance0, eta, rssi;
      IF rssi0 IS NOT NULL AND rssi > -80 AND distance0 IS NOT NULL AND rssi IS NOT
           NULL AND eta IS NOT NULL
      THEN
        -- calculate distance for rssi more than -80 dBm
        distance_1 = distance0 * (10 ^ ((rssi0 - rssi) / (10 * eta)));
        EXECUTE 'UPDATE mobile SET distance_1 = $$'||distance_1||'$$ WHERE src = $$
            '||src||'$$ AND dst = $$'||dst||'$$';
      ELSE
        -- if not rssi or not parameters or rssi <= -80 update distance to null (
            not relevant)
        EXECUTE 'UPDATE mobile SET distance_1 = null WHERE src = $$'||src||'$$ AND
            dst = $$'||dst||'$$';
      END IF;
    END IF;
  END LOOP;
  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION calc_est_distance_1()
  OWNER TO root;
```

B.7 calc_est_distance_2()

```
-- author: Juraj Popovec
-- Function: calc_est_distance_2()

CREATE OR REPLACE FUNCTION calc_est_distance_2()
  RETURNS trigger AS
$BODY$
DECLARE
  id mobile.id%TYPE;
  src mobile.src%TYPE;
  dst mobile.dst%TYPE;

  rssi0 mobile.rssi_med%TYPE;
  distance0 mobile.distance%TYPE;
  distance_1 mobile.distance_1%TYPE;
```

```
  eta anchor.eta%TYPE;
  rssi mobile.rssi_med%TYPE;
BEGIN
  -- for all rows in mobile
  FOR id, src, dst IN SELECT * FROM mobile
  LOOP
    -- if dst address is > 32512 (anchor 2.9m)
    IF dst > 32512
    THEN
      -- algorithm 2: get rssi between mobile and anchor (2.9m) + parameters
          between anchor and referenced anchor (2.9m)(closest [highest rssi] to
          mobile, but not the same)
      EXECUTE 'SELECT anchor.rssi_mean, anchor.distance, anchor.eta, mobile.
          rssi_mean FROM anchor INNER JOIN mobile ON anchor.src = mobile.dst
      WHERE mobile.src = $$'||src||'$$ AND mobile.dst = $$'||dst||'$$ AND anchor.
          dst = (
      SELECT mobile.dst FROM mobile WHERE src = $$'||src||'$$ AND dst > 32512 AND
          dst != $$'||dst||'$$ ORDER BY mobile.rssi_mean DESC LIMIT 1)' INTO rssi0,
           distance0, eta, rssi;
      IF rssi0 IS NOT NULL AND rssi > -80 AND distance0 IS NOT NULL AND rssi IS NOT
           NULL AND eta IS NOT NULL
      THEN
        -- calculate distance for rssi more than -80 dBm
        distance_1 = distance0 * (10 ^ ((rssi0 - rssi) / (10 * eta)));
        EXECUTE 'UPDATE mobile SET distance_2 = $$'||distance_1||'$$ WHERE src = $$
            '||src||'$$ AND dst = $$'||dst||'$$';
      ELSE
        -- if not rssi or not parameters or rssi <= -80 update distance to null (
            not relevant)
        EXECUTE 'UPDATE mobile SET distance_2 = null WHERE src = $$'||src||'$$ AND
            dst = $$'||dst||'$$';
      END IF;
    END IF;
  END LOOP;
  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION calc_est_distance_2()
  OWNER TO root;
```

B.8 calc_est_distance_3()

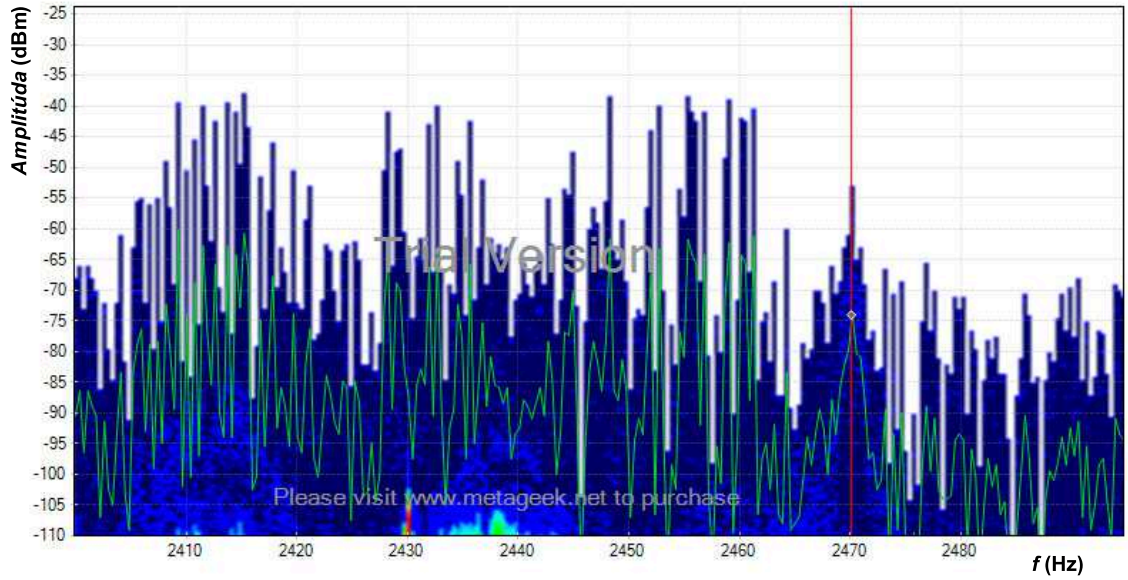
```
-- author: Juraj Popovec
-- Function: calc_est_distance_3()

CREATE OR REPLACE FUNCTION calc_est_distance_3()
  RETURNS trigger AS
$BODY$
DECLARE
  id mobile.id%TYPE;
  src mobile.src%TYPE;
  dst mobile.dst%TYPE;
  cal mobile.src%TYPE;
  cal_l mobile.src%TYPE;
  dst_h mobile.src%TYPE;

  rssi0 mobile.rssi_med%TYPE;
  distance0 mobile.distance%TYPE;
  distance_3_1 mobile.distance_1%TYPE;
  distance_3_2 mobile.distance_1%TYPE;
  eta anchor.eta%TYPE;
  rssi mobile.rssi_med%TYPE;
BEGIN
  -- for all rows in mobile
  FOR id, src, dst IN SELECT * FROM mobile
  LOOP
    -- if dst address is <= 32512 (anchor 0.75m)
    IF dst <= 32512
    THEN
      -- calculate dst address of anchor pair (2.9m)
      dst_h = dst + 32512;
      -- get address of calibrating node (2.9m)
      EXECUTE 'SELECT dst FROM mobile WHERE mobile.src = $$'||src||'$$ AND mobile.
          dst > 32512 AND mobile.dst != $$'||dst_h||'$$ ORDER BY mobile.rssi_mean
          DESC LIMIT 1' INTO cal;
      IF cal IS NULL
      THEN
        RETURN NULL;
      END IF;
      -- get parameters to calculate distance 1 ... mobile - anchor(0.75m) -
          calibrating anchor (2.9m)
      EXECUTE 'SELECT anchor.rssi_mean, anchor.distance, anchor.eta, mobile.
          rssi_mean FROM anchor INNER JOIN mobile ON anchor.src = mobile.dst WHERE
      mobile.src = $$'||src||'$$ AND mobile.dst = $$'||dst||'$$ AND anchor.dst = $$
          '||cal||'$$' INTO rssi0, distance0, eta, rssi;

      IF rssi0 IS NOT NULL AND rssi > -80 AND distance0 IS NOT NULL AND rssi IS NOT
           NULL AND eta IS NOT NULL
```

```
      THEN
        distance_3_1 = distance0 * (10 ^ ((rssi0 - rssi) / (10 * eta)));
      ELSE
        distance_3_1 = null;
      END IF;
      cal_1 = cal - 32512;
      -- get parameters to calcualate distance 2 ... mobile - anchor(2.9m) -
          calibrating anchor (0.75m)
      EXECUTE 'SELECT anchor.rssi_mean, anchor.distance, anchor.eta, mobile.
          rssi_mean FROM anchor INNER JOIN mobile ON anchor.src = mobile.dst WHERE
      mobile.src = $$'||src||'$$ AND mobile.dst = $$'||dst_h||'$$ AND anchor.dst =
          $$'||cal_1||'$$' INTO rssi0, distance0, eta, rssi;

      IF rssi0 IS NOT NULL AND rssi > -80 AND distance0 IS NOT NULL AND rssi IS NOT
           NULL AND eta IS NOT NULL
      THEN
        distance_3_2 = distance0 * (10 ^ ((rssi0 - rssi) / (10 * eta)));
      ELSE
        distance_3_2 = null;
      END IF;
      -- find lower value of disance and update
      IF distance_3_1 < distance_3_2
      THEN
        IF distance_3_1 IS NOT NULL
        THEN
          EXECUTE 'UPDATE mobile SET distance_3 = $$'||distance_3_1||'$$ WHERE src
              = $$'||src||'$$ AND dst = $$'||dst||'$$';
        END IF;
      ELSE
        IF distance_3_2 IS NOT NULL
        THEN
          EXECUTE 'UPDATE mobile SET distance_3 = $$'||distance_3_2||'$$ WHERE src
              = $$'||src||'$$ AND dst = $$'||dst||'$$';
        END IF;
      END IF;
    END IF;
  END LOOP;
  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION calc_est_distance_3()
  OWNER TO root;
```

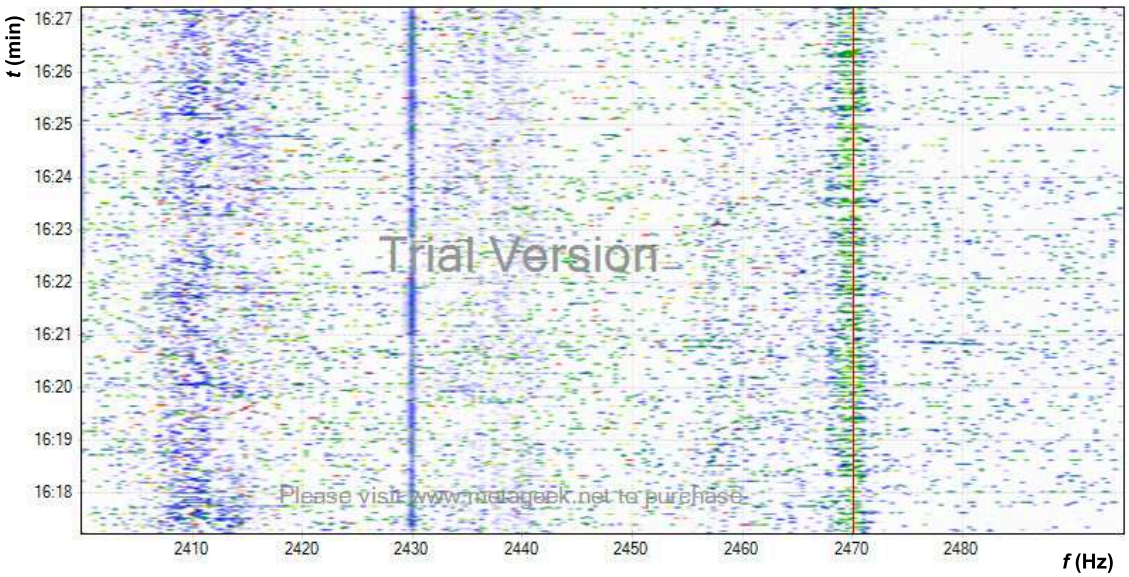
B.9 calc_abs_error()

```
-- author: Juraj Popovec
-- Function: calc_abs_error()

CREATE OR REPLACE FUNCTION calc_abs_error()
  RETURNS trigger AS
$BODY$
BEGIN
  -- calculate absolute error from new values (insert/update)
  NEW.abs_err_0 = ABS(NEW.distance - NEW.distance_0);
  NEW.abs_err_1 = ABS(NEW.distance - NEW.distance_1);
  NEW.abs_err_2 = ABS(NEW.distance - NEW.distance_2);
  NEW.abs_err_3 = ABS(NEW.distance - NEW.distance_3);

  RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION calc_abs_error()
  OWNER TO root;
```

C FREKVENČNÉHO PÁSMO



Obr. C.1: Využitie frekvenčného pásma počas merania pri využití kanálu 24



Obr. C.2: Dátovy tok počas merania pri využití kanálu 24