VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Milan Bubniak

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO VČASNOU DETEKCI DEGRADACE PŘENOSOVÝCH PARAMETRŮ MIKROVLNNÝCH SPOJŮ

APPLICATION FOR EARLY DETECTION OF TRANSMISSION PARAMETERS DEGRADATION OF MICROWAVE UNITS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Milan Bubniak**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. et Ing. Petr Musil**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Milan Bubniak ***ID:*** 197708
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Aplikace pro včasnou detekci degradace přenosových parametrů mikrovlnných spojů

POKYNY PRO VYPRACOVÁNÍ:

V rámci bakalářské práce důkladně prostudujte přenosové parametry mikrovlnných spojů a jejich vzájemnou souvislost. Použitím periodicky vyčítaných parametrů mikrovlnného spoje vytvořte aplikaci, která bude průběžně monitorovat stav jednotlivých mikrovlnných spojů a získaná data ukládat do vhodného databázového systému. Výstupem bakalářské práce bude aplikace schopná detekovat prudký pokles přenosových parametrů i potenciálně hrozící poruchy, které by byly odhaleny analýzou trendů v dlouhodobě shromažďovaných datech. Vytvořená aplikace by měla disponovat signalizací pro obsluhu, také být přehlednou a nenáročnou pro použití s velkým množstvím spojů. Aplikace musí využívat protokolu SNMP ve verzích, které jsou podporovány vybranou skupinou provozovaných mikrovlnných jednotek. Součástí práce je i vyhodnocení datové náročnosti síťových přenosů aplikace i ukládaných databázových dat.

DOPORUČENÁ LITERATURA:

[1] MAURO, Douglas R. a Kevin J. SCHMIDT. Essential SNMP. 2nd ed. Sebastopol, CA: O'Reilly, 2005. ISBN 0596008406.

[2] BETÁKOVÁ, Janka, Robert ZEMAN, Ján DVORSKÝ a Katarína HAVIERNIKOVÁ. Selected aspects of network management. Firts edition. Brno: Tribun EU, 2016. Knihovnicka.cz. ISBN 978-80-263-1063-1.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Ing. et Ing. Petr Musil

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Venkovní rádiové jednotky mikrovlnných spojů, vystavené často nepříznivým povětrnostním podmínkám, jsou ohroženy výskytem poruch, majících někdy za následek selhání i celé datové trasy, přičemž stejně tak může dojít i k poruše vnitřních jednotek. Tato bakalářská práce si klade za cíl vytvořit aplikaci schopnou včasného varování před blížícími se potencionálními poruchami, prostřednictvím analyzování trendů v monitorovaných parametrech spojů.

Teoretická část práce se věnuje základní charakteristice mikrovlnných spojů a struktuře protokolu SNMP.

Praktická část se zabývá návrhem aplikace s popisem její vnitřní struktury i grafického uživatelského rozhraní. Je navrženo a popsáno několik metod detekce poruch, dále jsou předloženy výsledky monitorování těmito metodami. Zahrnuto je také měření datové náročnosti síťové komunikace, i velikosti databázového úložiště nashromážděných dat.

Závěr shrnuje výsledky vývoje a předkládá možný budoucí vývoj aplikace s možností implementace nových funkcí.

KLÍČOVÁ SLOVA

mikrovlnný spoj, rádioreléový spoj, bezdrátový spoj, spoj bod-bod, monitorování sítě, detekce poruch, SLA, NOC, SNMP, .NET, WPF, InfluxDB, SQLite, OpenWeatherMap

ABSTRACT

Radio equipment of microwave links operating in an outdoor environment under the influence of weather conditions is at risk of a link failure or device defect occurrence, as well as their indoor units. This bachelor thesis aims to present the creation of a network monitoring application capable of timely detection of potential failures, based on analyzing trends in monitored device parameters.

The theory section deals with basic characteristics of microwave links and the structure of Simple Network Management Protocol.

The practical section describes an application design, including core architecture, as well as a graphical user interface. Several failure detection methods are designed and described. In the next section, practice monitoring results are presended. Furthermore, network traffic and data storage requirements are measured.

The conclusion summarizes the results of the application development and outlines it's future with new possible features.

KEYWORDS

microwave link, radiorelay link, wireless link, point-to-point link, network monitoring, failure detection, SLA, NOC, SNMP, .NET, WPF, InfluxDB, SQLite, OpenWeatherMap

BUBNIAK, Milan. *Aplikace pro včasnou detekci degradace přenosových parametrů mikrovlnných spojů*. Brno, 2020, 128 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. et Ing. Petr Musil

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Aplikace pro včasnou detekci degradace přenosových parametrů mikrovlnných spojů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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 autorského 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.

Brno ....

podpis autora

PODĚKOVÁNÍ

Děkuji svému vedoucímu práce panu Ing. et Ing. Petru Musilovi, za odborné vedení, trpělivost a podnětné návrhy k práci. Dále společnosti CBL Communication by light s.r.o., za poskytnutí testovacího rozsahu mikrovlnných spojů, zvláště pak panu Davidu Smékalovi za ochotnou spolupráci. Také bych chtěl poděkovat své rodině a všem, kteří mi byli při tvorbě práce oporou.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Mikrovlnné spoje jsou často využívaným přenosovým prostředkem datových linek krátkých i dálkových telekomunikačních tras. Jsou oblíbené pro svou jednoduchost nasazení a relativně nízkou nákladnost, oproti konkurenčním technologiím ve formě metalických a optických spojů. Nevýhodou je sdílené přenosové médium rádiových vln a z toho plynoucí náchylnost k rušení při operování vícero spojů ve stejných frekvenčních pásmech, a také degradace přenosových parametrů za nepříznivých meteorologických vlivů. Do smluvních podmínek zákazníků poskytovatelů služeb mikrovlnných spojů jsou také často zahrnuty poměrně přísné garance úrovně služeb, tzv. SLA, kdy neplnění kritérií těchto dohod, z důvodů poruch na zařízení, může mít pro provozovatele vážné finanční důsledky.

Cílem práce je vytvoření monitorovacího nástroje ve formě aplikace, vytvořené na míru konkrétním charakteristikám mikrovlnných spojů. Tato aplikace pak bude provádět monitorování stavových a přenosových parametrů mikrovlnných jednotek v reálném čase – jejich vzdálené vyčítání ze zařízení, jakož i záznam hodnot těchto parametrů do databázového úložiště a vyhodnocování zaznamenaných dat, s upozorněním na případné odchylky a anomálie.

Přínosem aplikace by mělo být potenciální odhalení blížící se poruchy mikrovlnné jednotky ještě před jejím samotným selháním, a to v případech, kdy je možné tuto poruchu na základě různých symptomů předpovědět. Jedná se o situace např. postupného selhávání napájecího zdroje, projevujícího se kolísáním napájecího napětí; výrazného nárůstu teploty zařízení; či nestability síly a kvality signálu. Tímto způsobem by mohlo být možné předejít havárii na síti daného poskytovatele služeb mikrovlnných spojů, resp. jeho zákazníka. Další důležitou funkcí by měla být včasná indikace již nastalých poruch. Dokončená aplikace by měla být připravená k nasazení v dohledových centrech poskytovatelů služeb, a přispět zde ke zvýšení spolehlivosti jejich spojů ve spojení s možnostmi včasné výměny vadné části spoje, např. v rámci preventivní údržby.

1 Mikrovlnné spoje

Jako mikrovlnné spoje jsou nazývána bezdrátová rádiová spojení mezi dvěma body, operující na frekvencích standardně vyšších než 1 GHz. Tato dvoubodová topologie je označována výrazem point-to-point. Mikrovlnné paprsky se šíří po přímce, na protější stanici jsou směřovány v úzkém vyzařovacím svazku pomocí parabolické směrové antény. Nemají schopnost ohybu okolo překážek či zemského povrchu, pro kvalitní spoj point-to-point je tedy podmínkou dosažení přímé viditelnosti mezi spojovanými body. Tento požadavek lze obejít využitím opakovačů signálu, tzv. retranslačních stanic (viz kapitola 1.2 Retranslace). Zařízení realizující bezdrátová point-to-point spojení jsou v telekomunikační praxi běžně označována jako mikrovlnná pojítka.

Historicky lze mikrovlnné spoje rozdělit na

- analogové spoje (využívající analogovou modulaci),
- digitální spoje (využívající digitální modulaci) [1].

V současnosti je téměř ve všech nasazeních využíváno spojů digitálních. Spoje můžeme dělit i dle využívaného frekvenčního pásma na

- bezlicenční spoje, pracující ve volných frekvenčních pásmech,
- licenční spoje, pracující ve frekvenčních pásmech podléhajících licenci ČTÚ [2].

Dále můžeme rozlišovat mezi spoji na bázi standardů IEEE 802.11 (Wi-Fi), případně IEEE 802.16 (WiMAX) a spoji radioreléovými využívajících proprietárních technologií.

1.1 Frekvenční pásma

Mikrovlny jsou elektromagnetické vlny rádiového spektra s frekvencí od 300 MHz do 300 GHz [2].

Rozdělení jednotlivých částí rádiového spektra do frekvenčních pásem je na národní úrovni úlohou Českého telekomunikačního úřadu (ČTÚ), který mimo jiné přejímá normy a nařízení Mezinárodní telekomunikační unie (ITU, z angl. *International Telecommunication Union*). Pro potřebu mikrovlnných spojů jsou podstatná následující pásma:

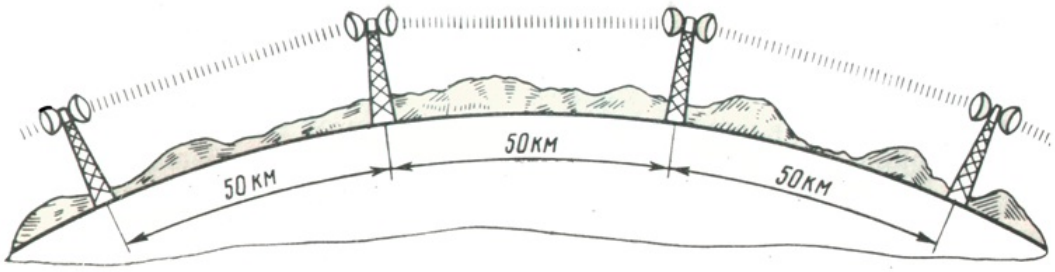
- Volná: 2,4; 5; 10; 17; 24; a 80 (71–76; 81–86) GHz. Dále 57–66 GHz pouze pro využití uvnitř budov.
- Licencovaná: 3,5; 3,8–4,2; 6; 7; 11; 13; 15; 18; 23; 26; 32; 38; 42; 48–50; 51–52; 57–59 GHz [3, 4].

Licencovaná pásma podléhají získání licence od ČTÚ, tj. nutné registraci provozovatele a hrazení pravidelných licenčních poplatků, jejichž výše se odvozuje od vybraného pásma, délky spoje, vysílacího výkonu a šířky použitého kanálu.

Pro spoje na větší vzdálenosti (delší než 15 km) je doporučována volba nižších frekvencí (do 8 GHz). Do vzdálenosti 25 km je možno použít i frekvencí 10/11 GHz. Pro menší vzdálenosti (méně než 15 km) lze použít frekvence nad 11 GHz [2].

1.2 Retranslace

Běžná mikrovlnná pojítka mají dosah v řádu maximálně desítek kilometrů. Pro překonání větších vzdáleností, případně v situacích, kdy není možné dosáhnout přímé viditelnosti mezi dvěma spojovanými body, se využívá tzv. retranslace, resp. retranslačních stanic. Jedná se o opakovače signálu sestávající ze dvou mikrovlnných pojítek – první pojítko signál přijme, zesílí a druhým pojítkem se vyšle dál požadovaným směrem. Topologie spoje s retranslací je ilustrována na obrázku 1.1. Celá trasa spoje s retranslací tedy sestává z více než jednoho skoků[1].



Obr. 1.1: Radioreléový spoj využívající více úrovní retranslace. Převzato z [5].

Lze využívat i pasivní retranslace, kdy na retranslačních stanicích není využito žádného aktivního prvku. Signál je zde buďto směrován pomocí odrazné desky (mikrovlnného „zrcadla“), která odrazí mikrovlnný paprsek z původní dráhy do požadovaného směru, anebo je využito dvou parabolických antén, které jsou vzájemně propojeny pomocí vhodného vlnovodu. Nevýhodou těchto pasivních řešení je snížení úrovně přijímaného výkonu signálu v koncových stanicích, a tedy nutnost kompenzace takovýchto ztrát zvýšením vysílacího výkonu, anebo nasazením větších antén. [2]

[1] V angl. *hops*, běžně užívaný pojem v oblasti směrování a síťových topologií.

1.2.1 Podmínky mikrovlnné retranslace

Jelikož se mikrovlnný signál šíří po přímce, je nutné aby případné retranslační stanice spoje byly umístěné metodou cik-cak a nikoli v jedné přímce [2]. Tím je zamezeno vzájemnému rušení jednotlivými úseky více-skokového spoje.

Pro úhel odchylky φ mezi dvěma retranslačními anténami jednoho mikrovlnného spoje na retranslační stanici jsou doporučována následující pravidla [2]:

- $\varphi < 20°$ – úseky nemohou pracovat na shodném kanálu. Polarizace mikrovlnného signálu by měla být opačná.
- $\varphi < 80°$ – úseky nemohou pracovat na shodném kanálu. Polarizace mikrovlnného signálu může být shodná.
- $80° < \varphi < 135°$ – úseky mohou pracovat na shodném kanálu. Polarizace mikrovlnného signálu by potom měla být opačná.
- $135° < \varphi < 180°$ – úseky mohou pracovat na shodném kanálu. Polarizace mikrovlnného signálu může být shodná.

1.3 Modulace

Modulace je proces, který pomocí změny amplitudy, fáze či frekvence nosného, obvykle vysokofrekvenčního signálu, přenáší požadovaný datový digitální či analogový modulující signál. Užívané typy modulace lze popsat s pomocí předpisu pro harmonickou funkci f jakožto funkce nosného signálu [2]:

$$f(x) = A\ \cos(2\pi f_c t + \varphi), \tag{1.1}$$

kde A je amplituda signálu, f_c je frekvence nosné, t je čas a φ je fázový posun [2].

- Amplitudová modulace (AM, ASK) používá změnu amplitudy A nosného signálu podle změny modulujícího signálu. Hodnoty f_c a φ se nemění.
- Frekvenční modulace (FM, FSK) používá změnu frekvence f_c nosného signálu podle změny modulujícího signálu. Hodnoty A a φ se nemění.
- Fázová modulace (PM, PSK) používá změnu fáze φ nosného signálu podle změny modulujícího signálu. Hodnoty A a f_c se nemění.
- Kvadraturní amplitudová modulace (QAM) používá změnu amplitudy A i fázového posunu φ nosného signálu podle změny modulujícího signálu. Hodnota f_c se nemění [2].

V oblasti mikrovlnných spojů, jsou užívány nejčastěji digitální modulace PSK (dvoustavová BPSK a čtyřstavová QPSK) a vícestavová M-QAM, kde M značí počet stavů nosné. Digitální modulace se oproti analogovým liší právě v konečném počtu stavů nosné (analogové mohou nabývat nekonečného počtu stavů).[2] Modulující digitální signál sestává z bitů, čili může nabývat pouze dvou hodnot: logické 0 a logické 1. Zatímco u dvoustavových modulací je každému bitu modulačního signálu přiřazen jeden stav nosné, u čtyřstavových modulací je to již na každý stav nosné dvojice bitů (tzv. dibit), u osmistavových modulací trojice bitů (tribit) atd. Tato závislost se dá zapsat vztahem $M = 2^n$, kde M udává počet stavů, a n počet bitů přiřazených jednomu stavu. Jednotlivé modulační stavy jsou nazývány taktéž jako symboly, přičemž počet přenesených symbolů za jednotku času udává veličina zvaná modulační rychlost (někdy též symbolová rychlost). Jednotkou modulační rychlosti je Baud, značený Bd, vyjadřující počet přenesených symbolů za jednu sekundu [6, 7].

Z výše uvedeného vyplývá, že zvyšováním počtu stavů modulace lze dosáhnout zvýšení přenosové (bitové) rychlosti při zachování modulační rychlosti. Zároveň se však přidává nutnost zvyšování úrovně odstupu signálu od šumu (viz kapitola 1.3.3), pro zachování kvalitativních parametrů přenosu. Typ zvolené modulace je tedy primárním parametrem, ovlivňujícím přenosovou rychlost a chybovost mikrovlnného spoje [6].

Pro grafické znázornění některých digitálních modulací, včetně PSK a QAM, je užíváno komplexní roviny IQ: I jako *In-phase* – synfázní složka na reálné ose, Q jako *Quadrature* – kvadraturní složka, reprezentující fázový posuv o 90°, resp. 270°, na imaginární ose. Do této roviny jsou zakreslovány vektory (obvykle však pouze jejich koncové body), odpovídající jednotlivým modulačním stavům, resp. jejich fázím a amplitudám (v případě modulace QAM). Výsledné zobrazení se nazývá konstelační diagram, zakreslené koncové body pak konstelační body [6, 7].

1.3.1 PSK

Modulace PSK, z angl. *Phase-shift keying* – klíčování fázovým posuvem, je metoda digitální modulace, která používá změny fáze nosného hermonického signálu. Je užíváno několik variant PSK lišících se dle počtu symbolů: dvoustavová BPSK, čtyřstavová QPSK, dále i vícestavové M-PSK, přičemž z vícestavových variant je užívána 8-PSK a 16-PSK, pro více stavů se již obvykle užívá složitější typ QAM modulace.

[2] V některé literatuře jsou digitální modulace nazývány jako klíčování, zatímco samotným slovem modulace jsou označovány pouze modulace analogové. Tato terminologie tak není zcela sjednocená.

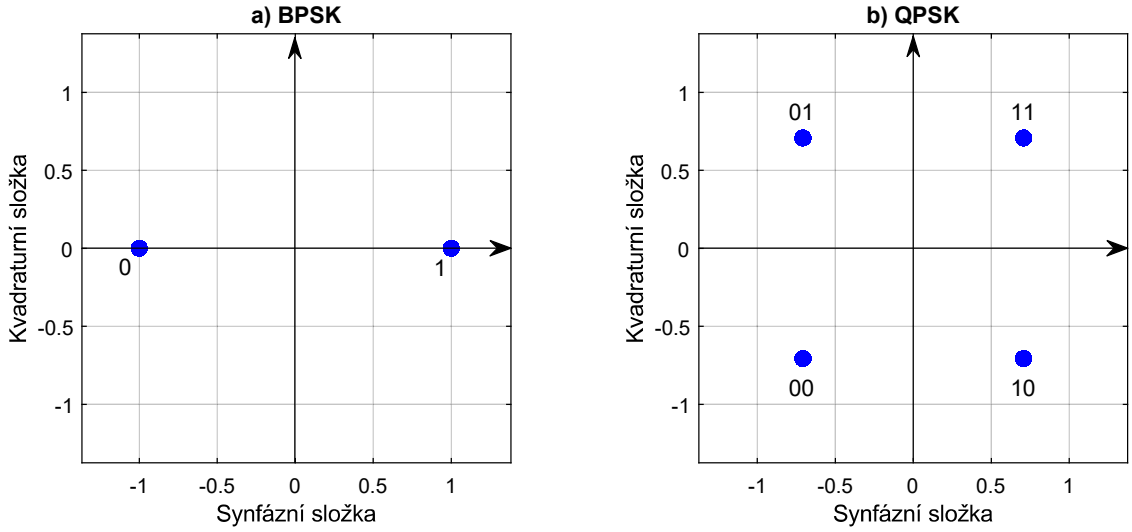
Konstelační body konstelačního diagramu jsou u modulace PSK umístěny na kružnici se středem v počátku, tzn. že všechny symboly jsou vysílány se stejnou energiií (amplitudou). Po obvodu kružnice jsou umístěny se stejnou úhlovou roztečí, čímž je zajištěna maximální efektivita a odolnost vůči chybám [6].

BPSK

Modulace BPSK, z angl. *Binary phase-shift keying* – binární klíčování fázovým posuvem, taktéž ji lze označit jako 2-PSK, je nejjednodušší dvoustavová varianta modulace PSK. Modulovaný signál (harmonická nosná) je fázově posunut buďto o 0° pro logickou hodnotu 1 modulujícího signálu, anebo o 180° pro logickou hodnotu 0. Viz konstelační diagram a) na obrázku 1.2 [6].

QPSK

Modulace QPSK, z angl. *Quadrature phase-shift keying* – kvadraturní klíčování fázovým posuvem, taktéž ji lze označit jako 4-PSK, je čtyřstavová varianta modulace PSK. Každý ze čtyř stavů přenáší dva bity. Modulovaný signál (harmonická nosná) je fázově posunut o 45°, 135°, 225° či 315° pro jednotlivé dvojice bitů. Viz konstelační diagram b) na obrázku 1.2 [6].

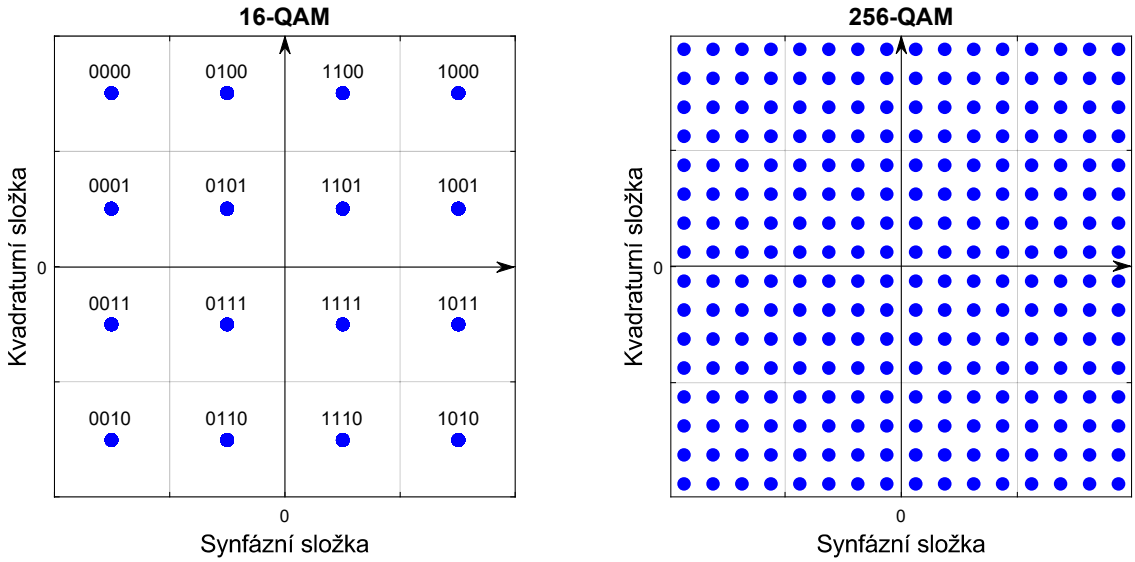


Obr. 1.2: Konstelační diagramy modulací BPSK a QPSK.

1.3.2 QAM

Modulace QAM, z angl. *Quadrature amplitude modulation* – kvadraturní amplitudová modulace, je metoda digitální modulace (případně i analogové modulace), která používá změn jak fáze, tak amplitudy nosného harmonického signálu. Samotný modulační proces probíhá ve dvou paralelních větvích, kdy každá větev využívá samostatného nosného sinusového signálu, který je amplitudově modulován digitálním modulujícím signálem, jehož tok byl nejprve rozdělen do těchto dvou větví (pomocí sériově-paralelního převodníku). První větev má fázový posuv nosného signálu 0°, a je nazývána jako synfázní. Druhá větev má fázový posuv nosného signálu 90°, a je nazývána jako kvadraturní. Výstupní signály obou větví jsou následně sečteny v součtovém členu, kde je výstupem již výsledný signál modulovaný jak fázově, tak amplitudově.

Jednotlivé varianty QAM jsou označovány jako M-QAM, kde M udává počet stavů. Množství amplitudových úrovní, kterých může nabývat signál modulovaný v synfázní a kvadraturní větvi, pak odpovídá druhé odmocnině z M. V současnosti jsou v oblasti mikrovlnných spojů užívány varianty od 16-QAM až po 4096-QAM. Celý princip je znázorněn na konstelačních diagramech na obrázku 1.3 [7].

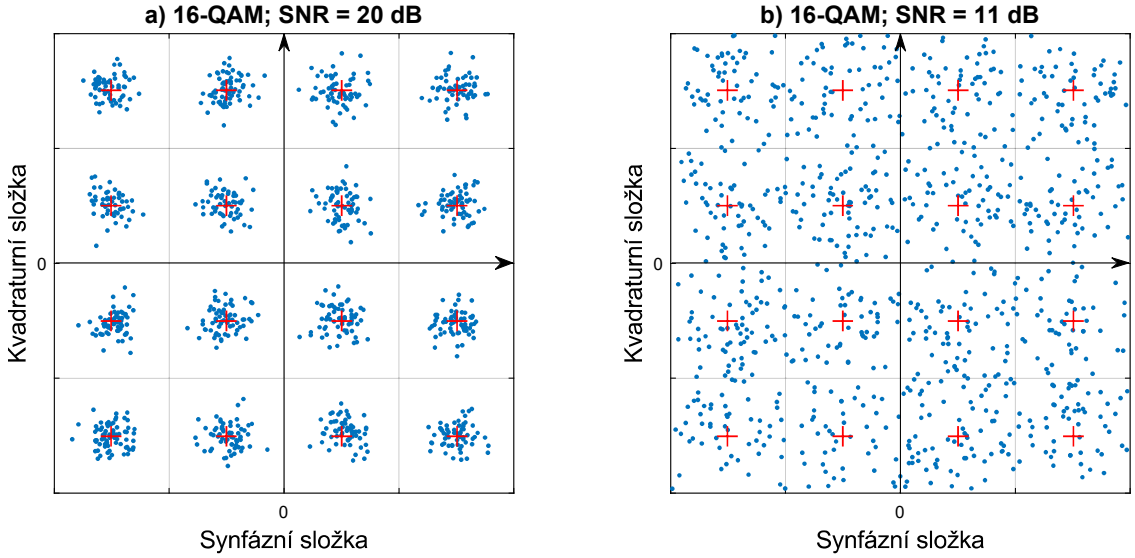


Obr. 1.3: Konstelační diagramy modulací 16-QAM a 256-QAM.

Výhodou modulace QAM je vysoké dosažitelné množství stavů a z toho vyplývající vysoká přenosová rychlost. Z dříve uvedeného vztahu $M = 2^n$ vyplývá, že teoretická přenosová rychlost, např. modulace 256-QAM, je čtyři krát vyšší, než u modulace QPSK, jelikož dokáže přenášet 8 bitů na jeden symbol. Nevýhodou je, že se zvyšujícím se počtem stavů, vzrůstá i náchylnost k rušení [6].

1.3.3 Rušení v přenosovém kanálu

Se zvyšujícím se počtem stavů vzrůstají nároky na demodulátor v přijímači, který musí rozlišit mnohem menší změny fáze, amplitudy či frekvence, které jsou navíc ovlivněny negativními vlivy během přenosu v komunikačním kanálu. Jedná se především o působení šumu, přítomném v každém komunikačním kanálu, a různých typů rušení. Při vyšším počtu stavů modulace je tedy pro bezchybný přenos, resp. chybovost v rámci tolerovaných mezí, zároveň vyžadován i vyšší odstup signálu od šumu (SNR, z angl. *Signal-to-noise ratio*, udávaný v dB, viz kapitola 1.7.3). Při příliš nízkém SNR, dochází v kostelačním diagramu na straně přijímače ke kolísání přijatých symbolů, resp. jejich konstelačních bodů, okolo ideální (referenční) polohy. Při příliš velkém rozptylu poloh těchto bodů, může dojít k chybnému demodulování přenesených symbolů a vzniku symbolové, resp. bitové chybovosti [6, 7]. Obrázek 1.4 v části a) zobrazuje přenesený signál modulace 16-QAM, ovliněný širokopásmovým šumem, při úrovni SNR 20 dB, kdy je již patrný viditelný rozptyl konstelačních bodů, ale stále je možné signál demodulovat se snesitelnou mírou chybovosti. V části b) je zobrazen přenesený signál modulace 16-QAM při úrovni SNR 12 dB, kdy je již korektní demodulace signálu nemožná.



Obr. 1.4: Konstelační diagramy modulací 16-QAM a 256-QAM, ovlivněných různými úrovněmi širokopásmového šumu.

Řešením je použití jednodušší modulace s nižším počtem stavů (čili snížení přenosové rychlosti), či snížení útlumu mikrovlnné trasy např. zkrácením vzdálenosti mezi spojovanými body, nebo zesílením vysílacího výkonu (většinou pouze omezené možnosti zesílení z důvodu regulačních předpisů ČTÚ).

1.4 Spoje radioreléové

Při nutnosti bezdrátového spojení se v podnikové sféře a na páteřních či distribučních trasách poskytovatelů telekomunikačních služeb používají výhradně spoje radioreléové, resp. zejména tam, kde zavedení optických či metalických kabelových tras není finančně rentabilní. Jedná se o výhradně směrové spoje typu point-to-point. Komunikovat mohou v licenčních i volných pásmech. Jsou užívány taktéž jako jeden ze způsobů připojení základnových stanic celulárních mobilních sítí (BTS – z angl. *Base Transceiver Station*) k páteřním sítím mobilních operátorů [4]. V některých částech světa jsou jediným reálně nasaditelným vysokorychlostním komunikačním prostředkem, schopným překonat stovky i tisíce kilometrů v prostředích, jako jsou lesy, hory, pouště, apod. To mimo jiné díky tzv. retranslaci. V extrémních podmínkách je také možné mikrovlnná pojítka či retranslační stanice napájet pomocí solárních článků.

Hardware stanic radioreléových spojů sestává z vnitřní jednotky (IDU, z angl. *Indoor Unit*), venkovní jednotky (ODU, z angl. *Outdoor Unit*) a parabolické antény [2]. Populárním řešením je také koncepce „full-outdoor“, kdy je IDU i ODU sloučena do jediné kompaktní venkovní jednotky. Možná je i koncepce „full-indoor“, kdy se IDU i ODU nachází ve vnitřním prostředí, kde s anténou je tato vnitřní jednotka propojena vhodným vlnovodem [8].

ČR patří mezi úspěšné producenty radioreléových spojů. Mezi nejznámější tuzemské výrobce mikrovlnných jednotek patří firmy Alcoma, Summit Development či Racom. Ze zahraničních výrobců jsou to pak například společnosti Ericsson, Ceragon, Siklu, Siae, Motorola a další.

1.4.1 IDU

Indoor Unit, neboli vnitřní jednotka, zajišťuje připojení ke kabelové síti, multiplexování[3] dat, modulaci vysílaných dat a demodulaci přijímaného signálu, napájení ODU a také dohled a servisní management jednotky. Sestává z několika desek provádějících výše jmenované funkce [8].

I/O[4] deska je vybavena rozhraními pro připojení do sítí typu Ethernet (fyzická rozhraní v podobě portu RJ45 nebo modulu SFP), případně rozhraními pro sítě PDH/SDH (linky E1/T1) [8]. Dále může být vybavena samostatným managementovým ethernetovým portem, sériovým terminálovým portem a dalšími [2].

[3] Proces spojení několika datových toků (zde z různých rozhraní) do jednoho signálu.

[4] Z angl. input/output, vstupně/výstupní. Označuje hardware zajišťující komunikaci zařízení s uživatelem nebo jinými zařízeními.

Modemová deska zajišťuje modulaci vysílaných dat do ODU a demodulaci přijímaného signálu z ODU. Moduluje se na nosný tzv. mezifrekvenční kmitočet, který je teprve v ODU převeden na kmitočet vysílací (a naopak, je přijímán přijímací kmitočet, v ODU převeden na mezifrekvenční a následně demodulován). Jako mezifrekvenční kmitočet se nejčastěji užívá 70 a 140 MHz pro nižší frekvenční pásma nebo 350 a 850 MHz pro vyšší pásma [8, 2]. Užívané techniky modulace jsou rozebrány v kapitole 1.3.

Napájecí deska zajišťuje napájení ODU. To je realizováno pomocí koaxiálního kabelu, po kterém je přenášen i samotný vysokofrekvenční signál [2].

Firmwary[5] různých IDU podporují různé funkce, v závislosti na vybavených rozhraních, ceně atd. Z těch široce podporovaných lze jmenovat např. standard IEEE 802.1Q[6] u ethernetových rozhraní, nebo dohledový protokol SNMP, který je stěžejní pro praktickou část této práce a bude podrobně rozebrán v kapitole 2.

1.4.2 ODU

Outdoor Unit, neboli venkovní jednotka, sestává ze dvou částí – vysílače a přijímače.

Vysílač sestává z oscilátoru, generujícího frekvenci odpovídající zvolenému kanálu příslušného mikrovlnného pásma. Ta je ve směšovači smíšena s přiváděným mezifrekvenčním signálem z IDU, který byl nejprve zesílen na vhodnou úroveň. Následně je signál přiváděn do RF filtru, kde jsou odstraněny nadbytečné kmitočty. Jelikož je signál po průchodu směšovačem a filtrem velice slabý, prochází ještě zesilovačem kde je zesílen na finální vysílanou úroveň. Odtud je signál přiváděn přes sdružovač do napáječe antény [1, 2].

Přijímač zesiluje signál z antény, mění jej na mezifrekvenční signál a odvádí jej do demodulátoru v IDU. Signál nejprve prochází pásmovým filtrem, následně nízkošumovým zesilovačem kde je zesílen na pracovní úroveň. Opět prochází pásmovým filtrem a přichází do směšovače, kde je smíšen s mezifrekvenčním kmitočtem generovaným patřičným oscilátorem. Následně putuje do mezifrekvenčních zesilovačů, které disponují automatickým řízením zisku, jelikož se úroveň přijímaného signálu v čase mění (vlivem úniků) – přičemž do demodulátoru, kam je signál následně odváděn, je nutno přivádět signál stále úrovně. Důležitou vlastností přijímače je mimo jiné selektivita, tj. schopnost potlačovat všechny nežádoucí signály, tak aby bylo zajištěno zpracování signálu pouze na zvoleném kanálu [1, 2].

[5] Označení řídícího softwaru zařízení, trvale uloženého v nevolatilní paměti (obvykle typu flash).

[6] Standard umožňující rozdělení fyzické ethernetové sítě na vícero virtuálních sítí (tzv. VLAN), pomocí rozšíření hlavičky ethernetového rámce.

ODU může být instalována buďto přímo na anténě, a tvořit s ní tak kompaktní celek, anebo být umístěna samostatně, a s napáječem antény být propojena vhodným vlnovodem [2].

1.4.3 Anténa

Anténa vyzařuje do prostoru mikrovlnný výkon generovaný ve vysílači ODU a přivádí přijímaný mikrovlnný signál do přijímače ODU. U antén radioreléových spojů je vyžadováno, aby anténa měla vyzařovací diagram[7] s malými postranními laloky a co největší účinnost. Většinou jsou navrženy tak, aby mohly pracovat se všemi kanály v daném frekvenčním pásmu [2]. Téměř výlučně mají podobu rotačních parabolických reflektorů (zrcadel), s přímým nebo Cassegrainovým napáječem [1].

Přímý napáječ je realizován tak, že reflektor ve formě rotačního paraboloidu z vysoce odrazného materiálu, je ze svého ohniska ozařován trychtýřovým zářičem. Dochází zde však k částečnému zastínění a bočnímu vyzařování, jelikož zářič musí být napájen vlnovodem, který svým umístěním protíná výstupní vlnu. Tento efekt je potlačován tzv. límcem – válcovitým přesahem po obvodu reflektoru, který zmírňuje vyzařování v nežádoucích směrech [1].

V případě Cassegrainova napáječe, se ústí zářiče nachází ve středu reflektoru, který je zpětně ozařován subreflektorem v podobě rotačního hyperboloidu, umístěným v jeho ohnisku. Díky tomu, že vlnovod zářiče nezasahuje do vyzařovacího prostoru antény, má takováto anténa příznivější vyzařovací diagram [1].

Běžně používané parabolické antény radioreléových spojů mají velikostní průměr reflektoru v rozsahu 0,35–2,40 m. [2].

1.4.4 Linkové parametry radioreléových spojů

Tabulka 1.1 uvádí přehled typických hodnot linkových parametrů radioreléových spojů u nejběžněji nasazovaných frekvenčních pásem. Údaje jsou převzaty z publikace *Technologie k připojení základnových stanic mobilní sítě k páteřní síti* [4]. Kalkulace předpokládá užití modulace 1024-QAM, SNR 34 dB, výkon 20 dBm a rezervu na únik 12 dB. Potřebná velikost průměru antény ve vyšších frekvenčních pásmech klesá, díky rostoucímu anténnímu zisku se vzrůstající frekvencí.

Z tabulky lze vysledovat, že přenosovou rychlost spoje nejvýrazněji ovlivňuje šířka frekvenčního pásma. Některé z uvedených parametrů jsou dále rozebrány v kapitole 1.7.

[7] Též směrový diagram, je grafickým vyjádřením citlivosti antény v různých směrech.

Tab. 1.1: Ukázka linkových parametrů radioreléových spojů [4].

Pásmo	(GHz)	7	11	13	15	18	26	38	42
Šířka pásma	(MHz)	40	80	28	28	110	56	56	112
Přenosová rychlost	(Mbit/s)	320	640	224	224	880	448	448	896
Průměr antény	(m)	1,2	1,2	0,9	0,9	0,9	0,65	0,65	0,65
Zisk antény	(dBi)	37,3	41,3	40,2	41,5	43,0	43,4	46,7	47,6
Šum	(dBm)	−84	−81	−86	−86	−80	−83	−83	−80
Útlum spoje	(dB)	133	137	140	142	140	143	150	149
Vzdálenost	(km)	14,7	16,3	18,3	21,2	12,8	13,5	19,8	15,5

1.5 Spoje založené na standardu IEEE 802.11

IEEE[8] 802.11 je soubor norem definujících fungování bezdrátových sítí na lokální úrovní, tj. uvnitř budov, případně uzavřených venkovních míst jako jsou dvory, náměstí apod. Takto vytvořené sítě se označují zkratkou WLAN (z angl. *Wireless Local Area Network* – bezdrátová místní síť). Jsou navrhnuty primárně pro síťovou topologii point-to-multipoint, kdy se v dané WLAN nachází jeden přístupový bod, běžně označovaný jako AP (Access Point), ke kterému se mohou připojit jeden či více bezdrátových klientů.

Bezdrátové technologie vyhovující vybraným standardům IEEE 802.11 jsou běžně označovány jako Wi-Fi[9]. Toto označení je ve skutečnosti obchodní známkou pro zařízení, která prošla certifikačním procesem organizace Wi-Fi Alliance. Zařízení nedisponující touto certifikací mohou, ale nemusí standardům IEEE 802.11 vyhovovat [9].

Standardy IEEE 802.11 jsou však v nenáročných podmínkách, navzdory původnímu určení, užívány i pro venkovní spoje, a to jak typu point-to-multipoint, tak typu point-to-point, tedy bezdrátovému mostu mezi dvěma body. Nejčastěji jsou to spoje posledních mílí[10] pro připojení domácností nebo malých firem. Jedná se o značně levnější a dostupnější řešení oproti radioreléovým spojům, nevýhodou je však omezený dosah, nižší rychlosti a velká náchylnost k rušení (i z důvodu užití bezlicenčních pásmem, nejčastěji 2,4 GHz a 5 GHz). To z nich činí nepříliš vhodnou

[8]Z angl. *Institute of Electrical and Electronics Engineers* – je mezinárodní organizací pro standardizaci v oblasti elektrotechniky a síťových komunikací.

[9]Zkratka nemá oficiální význam, v některých zdrojích je chybně uváděno *Wireless Fidelity*, což má být analogií s názvem Hi-Fi, neboli *High Fidelity* (označením pro audiotechniku vysoké kvality) [10].

[10]Poslední míle, často též angl. *Last Mile* – je posledním spojovacím úsekem mezi distribuční sítí poskytovatele telekomunikačních služeb a lokální sítí zákazníka.

a spolehlivou variantu pro připojení podnikových sítí s garancí SLA[11], a zcela nevhodnou variantu pro spoje point-to-point na distribučních nebo páteřních sítích poskytovatelů telekomunikačních služeb.

Populární nízkonákladovou volbou pro venkovní Wi-Fi spoje jsou v ČR například zařízení společností Mikrotik a Ubiquiti.

1.5.1 Verze standardu IEEE 802.11

Ke komunikaci je využíváno bezlicenčních kmitočtových pásem 2,4 GHz a 5 GHz. Verze 802.11ad a vyvíjená verze 802.11ay pak využívají nově i 60 GHz. Existují však i verze operující v licenčním pásmu 3,6 GHz a dokonce i v sub-1GHz televizních pásmech. Viz tabulka 1.2 užívaných standardů IEEE 802.11 a příslušných pásem [11].

Tab. 1.2: Přehled verzí standardu IEEE 802.11 [11].

Verze standardu	Označení Wi-Fi Alliance	Pásmo (GHz)	Šířka pásma (MHz)	Fyzická vrstva	Rychlost (Mbit/s)
802.11	–	2,4	22	DSSS, FHSS	2
802.11b	Wi-Fi 1	2,4	22	DSSS	11
802.11a	Wi-Fi 2	5	20	OFDM	54
802.11g	Wi-Fi 3	2,4	22	OFDM	54
802.11y	–	3,7	20	OFDM	54
802.11n	Wi-Fi 4	5/2,4	40	MIMO-OFDM	600
802.11ac	Wi-Fi 5	5	160	MIMO-OFDM	3 466,8
802.11ad	–	60	2160	OFDM	6 757
802.11af	–	0,054–0,79	8	MIMO-OFDM	568,9
802.11ah	–	0,9	16	MIMO-OFDM	347
802.11ax	Wi-Fi 6	1–6	80	MIMO-OFDM	4 803

Pro venkovní spoje je nejčastěji užíváno verzí 802.11a/n/ac operujících v pásmu 5 GHz. V dřívějších letech byla užívána některými lokálními poskytovateli internetových služeb k připojení domácností i zařízení operující v pásmu 2,4 GHz, jakožto mimořádně nízkonákladové řešení. Z důvodu zarušení tohoto pásma je však tento způsob venkovního spojení již zcela nevhodný.

[11] Z angl. *Service Level Agreement* – označuje smluvní parametry poskytovaných služeb, u datových linek to často bývá zaručená maximální procentuální úroveň nedostupnosti, hladina odezvy, jitteru apod. Jejich nedodržení bývá penalizováno dle uzavřené smlouvy.

1.5.2 Fyzická vrstva

Standardy IEEE 802.11 jsou typu half-duplex, tj. v jeden okamžik lze data pouze přijímat či odesílat. Při větším počtu koncových stanic v síti je síť sdíleným médiem mezi všemi připojenými stanicemi. Pro eliminaci vzniku kolizí (stavu, kdy v jeden okamžik vysílá více než jedna stanice) je využíváno metody CSMA/CA. Ta při pokosu o vysílání spočívá v náhodně zvolené době, po kterou stanice naslouchá, a není-li mezitím médium obsazeno jinou stanicí, je odvysílán datový rámec a následně se čeká na jeho potvrzení [11].

Na fyzické vrstvě je využíváno technologií přenosu s rozprostřeným spektrem. Tyto technologie jsou díky velké šířce pásma značně odolnější vůči úzkopásmovému rušení než klasické rádiové systémy. Toho je zajištěno přidáním velké míry redundance, která je vyvážena právě šířkou pásma. Tři využívané typy jsou následující:

- FHSS – Frequency Hopping Spread Spectrum,
- DSSS – Direct Sequence Spread Spectrum,
- OFDM – Orthogonal Frequency Division Multiplexing,

přičemž s výjimkou nejstarších a v současnosti již obsoletních standardů 802.11 a 802.11b je u všech zbylých specifikací využíváno technologie OFDM [12].

OFDM

Princip technologie OFDM spočívá v rozdělení datového přenosu na mnoho paralelních datových přenosů na samostatných nosných frekvencích, s menší přenosovou rychlostí. Tyto tzv. subkanály jsou vnitřně modulovány metodami BPSK, QPSK či QAM, které jsou blíže rozebrány v kapitole 1.3 Modulace. Redundantní informace jsou k uživatelským datům přidány ještě před rozdělením toku do jednotlivých subkanálů, díky čemuž lze detekovat a opravit chybně přenesené části. Frekvenční spektra subkanálů se vzájemně překrývají, vzájemně se však neovlivňují, jelikož splňují tzv. podmínku ortogonality – jednotlivé subkanály jsou od sebe vzdáleny o celočíselný násobek převrácené hodnoty trvání jednoho symbolu (symbolové frekvence). Tím je zajištěno efektivní využívání frekvenčního pásma [12].

1.6 Spoje založené na standardu IEEE 802.16

IEEE 802.16 je soubor norem definujících fungování bezdrátových sítí pro venkovní spoje point-to-point a point-to-multipoint. Jedná se o sítě typu WMAN (z angl. *Wireless Metropolitan Area Networks* – bezdrátová metropolitní síť). Technologie standardu IEEE 802.16 jsou běžně označovány jako WiMAX[12] (analogie s názvem

[12]Z angl. *Worldwide Interoperability for Microwave Access.*

Wi-Fi u standardů 802.11). I tentokrát se jedná o obchodní název organizace WiMAX Forum, zabývající se certifikací kompatibilních zařízení a propagací standardu. WiMAX umožňuje přenos dat při přímé i nepříme viditelnosti koncových bodů.

Při přímé viditelnosti koncových bodů lze dosáhnout maximální teoretické přenosové rychlosti na fyzické vrstvě 268 Mbit/s, a maximální vzdálenosti koncových bodů cca 50 km. Bez přímé viditelnosti lze na fyzické vrstvě dosáhnout maximální přenosové rychlosti 75 Mbit/s, a vzdálenosti koncových bodů přibližně až 5 km [4].

V ČR je WiMAX provozován obvykle v licenčním pásmu 3,5 GHz. V zahraničí jsou dále využívána také pásma 2,3; 2,5 a 5,8 GHz [13].

Na fyzické vrstvě využívá technologie OFDMA (modifikace klasického OFDM uvedeného v kapitole 1.5.2, spočívající v rozdělení subkanálů mezi více klientů a umožnění tak několika současných přenosů od různých klientů) s vnitřními modulacemi BPSK, QPSK či QAM. Datový provoz je plánován s využitím časového nebo frekvenčního multiplexu, čímž je odbouráno riziko kolizí [13].

1.7 Parametry spojů

1.7.1 Vysílací výkon

Vysílací výkon, označovaný na mikrovlnných jednotkách jako Tx Power (*transmit power*), udává velikost mikrovlnného výkonu vyzářeného do prostoru. Je nastavován v jednotkách dBm, udávajících výkonový poměr vůči úrovni 1 mW. U běžných mikrovlnných spojů se pohybuje v rozmezí od -10 do 20 dBm. Jeho maximální hodnota je přísně regulována ČTÚ.

1.7.2 RSSI

Received Signal Strength Indication, neboli RSSI (někdy pouze RSS), udává úroveň výkonu přijímaného signálu. Je měřen v jednotkách dBm. Čím vyšší je hodnota RSSI, tím vyšší je výkonová úroveň přijímaného signálu. U mikrovlnných spojů se pohybuje v rozmezí od -90 dBm do -30 dBm. Mikrovlnné jednotky za pomoci této veličiny vyjadřují sílu signálu (parametr Signal Strength).

1.7.3 SNR

Signal-to-noise ratio, odstup signál-šum, neboli SNR, vyjadřuje poměr výkonové úrovně užitečného signálu vůči výkonové úrovni šumu. Udává se v jednotkách dB. Za šum lze považovat veškeré signály zasahující do spektra užitého mikrovlnného kanálu.

1.7.4 Eb/No

Eb/No je veličina, vyjadřující odstup signálu od šumu v přepočtu na jeden bit přenesené informace. Udává se v jednotkách dB. Je dána středním výkonem užitečného signálu, bitovou propustností spoje a spektrální výkonovou hustotou aditivního šumu. Pomocí Eb/No lze porovnávat chybovost různých digitálních modulací, jelikož není ovlivněna použitou modulací a velikostí symbolů. Mikrovlnné jednotky za pomoci této veličiny vyjadřují kvalitu signálu (parametr Signal Quality).

1.7.5 BER

Bit Error Ratio, neboli BER, je poměr průměrného počtu chybně detekovaných, resp. rekonstruovaných bitů vzhledem k celkovém počtu přenesených bitů přenesených za určitý časový interval. Je primárním parametrem určujícím chybovost spoje. Míra BER je přímo závislá na velikosti SNR. Za normálních podmínek, pakliže je spoj správně dimenzován, byl velikost BER měla být téměř nulová. Za hranici použitelnosti je považována hodnota BER 10^{-3} [1, 6].

1.7.6 Provozní parametry

Krom parametrů uvedených výše, lze z mikrovlnných jednotek získávat také fyzikální parametry snímané pomocí senzorů jednotky. V zásadě se jedná o teplotu zařízení, a velikost napájecího napětí. Oba dva parametry jsou snímány u většiny mikrovlnných jednotek. V případě full-outdoor jednotek, je snímán často pouze jediný teplotní údaj, v případě jednotek IDU-ODU, lze často vyčítat teplotu vnitřní i venkovní jednotky (u vnitřní jednotky například teplota procesoru, apod.).

1.8 Odolnost vůči meteorologickým vlivům

Nepříznivé meteorologické podmínky mohou významně ovlivnit, či zcela znemožnit, mikrovlnný přenos spoje. U vln s vlnovou délkou λ menší než 5 cm, tj. frekvencí nad 6 GHz, mohou déšť, sníh či mlha, absorbovat energii elektromagnetických vln mikrovlnného paprsku. Pro frekvence pod 6 GHz lze toto působení zanedbat. Ke znatelným problémům dochází u frekvnencí nad 10 GHz, kde již útlum způsobený těmito vlivy podstatně omezuje možnou vzdálenost mezi stanicemi [2]. Tento problém se dá částečně obejít pomocí technologie ACM, integrované většinou výrobců mikrovlnných jednotek.

1.8.1 ACM

Adaptive Coding and Modulation, neboli ACM, je metoda automatické změny modulace či kódového poměru datového přenosu mikrovlnné linky, vzhledem ke kvalitě přenosového kanálu. Za slunečného počasí může přenos probíhat vysokostavovými modulacemi, např. 1024-QAM či 2048-QAM, přičemž se zhoršujícími se meteorologickými podmínkami, se aktivní modulace postupně mění za modulace s nižšími počty stavů, přičemž v případě extrémního zhoršení podmínek může modulace klesnout až na typ QPSK či BPSK. To sice má za následek citelně nižší datovou propustnost linky, nicméně spojení zůstává zachováno. Pakliže by ke změně modulace nedošlo, došlo by zcela ke ztrátě spojení až do obnovení příznivých podmínek. V kombinaci s vhodným nastavením QoS mohou být negativní dopady na zákazníka minimální [14].

2 SNMP

Protokol SNMP, z angl. *Simple Network Management Protocol* je síťový protokol aplikační vrstvy (7. vrstva modelu ISO/OSI[1], resp. 4. vrstva modelu TCP/IP[2]), sloužící k získávání či nastavování hodnot na vzdáleném síťovém zařízení. Využívá transportního protokolu UDP[3], a to standardně na portech 161 a 162, resp. 10161 a 10162 při užití zabezpečené komunikace poslední verze protokolu (SNMPv3).

SNMP může fungovat na jakémkoli zařízení vybaveném jeho softwarovou podporou, nejčastěji jsou to síťové aktivní prvky (routery, switche, přístupové body, mikrovlnné jednotky, modemy, atd.), mohou to ale být i servery či koncové pracovní stanice (podpora protokolu SNMP je integrována i v systémech Microsoft Windows, také unixové systémy jsou samozřejmostí). Je nasazován jak v podnikových sítích (LAN i WAN) tak i rozlehlých sítích poskytovatelů telekomunikačních služeb (např. MPLS sítě aj.), kde je základním prostředkem pro centralizovaný vzdálený dohled a hromadnou správu prvků síťové infrastruktury, za pomocí monitorovacího softwaru implementujícího SNMP (resp. komponentu SNMP Manager, viz 2.1) [15, 16].

2.1 Komponenty

SNMP sestává ze dvou základních logických komponent: manageru a agenta [16].

2.1.1 Manager

Zařízení se spuštěnou komponentou Manager, označována také jako NMS (z angl. *Network Management Stations*, je monitorující prvek, obvykle server či koncová stanice, provádějící tzv. *polling* – odesílání dotazů monitorovaným zařízením (agentům), a zpětný sběr a následné zpracování či uložení takto získaných dat. Také naslouchá a přijímá zprávy typu *Trap*, asynchronně zasílané z agentů, oznamující vznik události na monitorovaném zařízení. Manager bývá implementován v monitorovacím softwaru, dohledujícím obvykle větší množství agentů [16].

[1] Z angl. *Open Systems Interconnection* – je referenční model vydaný organizací ISO, posléze přijatý za mezinárodní normu. Vyjadřuje rozdělení síťových protokolů do sedmi vrstev, které by, v ideálním případě, měly být na sobě nezávislé a protokoly v rámci jednotlivých vrstev nahraditelné.

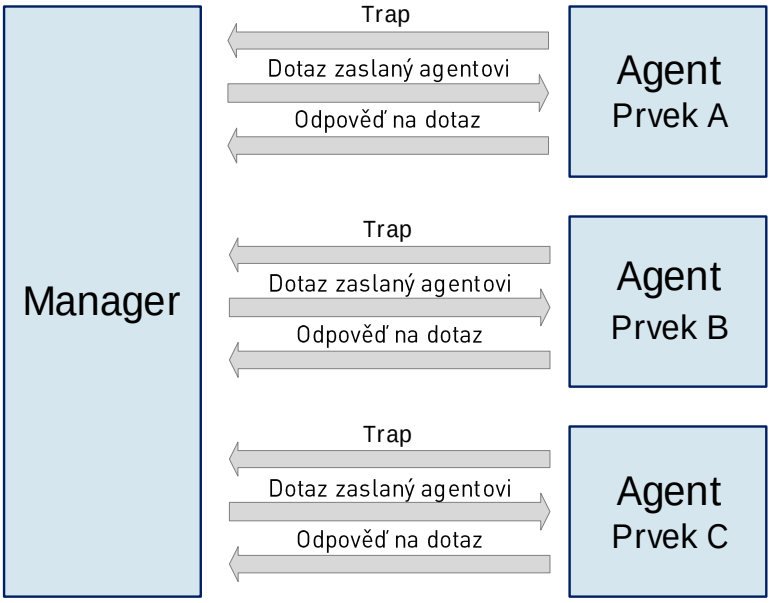
[2] Model TCP/IP, vydaný v rámci stejnojmenné sady síťových protokolů, je zjednodušením referenčního modelu ISO/OSI, redukující počet vrstev na čtyři.

[3] Z angl. *User Datagram Protocol* – je síťový protokol transportní vrstvy (4. vrstva modelu ISO/OSI). Jedná se o nespojový protokol, neposkytující záruku doručení přenášených datagramů (narozdíl od „konkurenčního“ protokolu TCP).

2.1.2 Agent

Zařízení se spuštěnou komponentou Agent je monitorovaný prvek, odpovídající na dotazy managera. Dále generuje a zasílá managerovi zprávy *Trap*, oznamující vznik událostí (např. aktivace či deaktivace síťového rozhraní, překročení teplotního prahu na vnitřním prvku zařízení, apod.). Uvnitř agenta je integrována MIB (z angl. Management Information Base), ve které jsou uloženy identifikátory všech dostupných proměnných (viz kapitola 2.5). Agent může existovat ve formě samostatného softwaru, obvykle však bývá přímo implementován jako součást operačního systému zařízení. V současnosti standardně každý aktivní spravovatelný síťový prvek ověřeného výrobce obsahuje alespoň základní implementaci SNMP agenta [16].

Polling a zasílání trap může nastávat nezávisle na sobě, kdykoli v čase. Vztah mezi managerem a agentem ilustruje diagram na obrázku 2.1.



Obr. 2.1: Vztah mezi SNMP managerem a agentem.

2.2 Verze

Stejně jako řada jiných protokolů, i SNMP prošel určitým historickým vývojem. Aktuálně existují tři majoritně používané verze protokolu, přičemž poslední verze SNMPv3 byla specifikována v IETF[4] RFC[5] v roce 2002. Jednotlivé verze jsou mezi sebou kompatibilní pouze částečně, či vůbec, je tedy nutná podpora shodné verze na straně manageru i agenta [15].

SNMPv1

Nejstarší verze protokolu definována v RFC 1157. K zabezpečení používá pouze prostý textový řetězec, tzv. *community string*, který je v nešifrované podobě přenášen uvnitř každé zprávy protokolu (struktura zpráv viz kapitola 2.3). Při přístupu k síťovému médiu lze tedy snadno odposlechnout či podvrhnout zasílané zprávy. Nasazení této verze je tedy bez obav možné pouze na sítích dostatečně zabezpečených před nežádoucím přístupem (typicky managementové sítě poskytovatelů telekomunikačních služeb). I přes tento nedostatek je, mimo jiné právě díky své jednoduchosti, stále široce využíván [15, 16, 17].

SNMPv2

Definován v RFC 1441 a 1452. Obsahuje podporu nových PDU: `GetBulkRequest`, `InformRequest` a `Report` (viz kapitola 2.3) a nových datových typů. Dále zavádí nový systém autentizace pomocí logických identifikátorů zvaných *parties*. Tento systém autentizace byl však považován za příliš komplikovaný a komplexní, a verze jako taková se nikdy neujala. Původní SNMPv2 bývá také označován jako SNMPv2p. Jelikož se původní protokol neujal, vniklo několik modifikací SNMPv2:

SNMPv2c (Community-Based SNMPv2) – obsahuje všechny změny obsažené v SNMPv2, s výjimkou kontroverzního autentizačního mechanismu. Zabezpečení je u této verze tedy identické s protokolem SNMPv1, využívá stále textových řetězců *community string*. Je definován v RFC 1901–1908, a dosáhl značně větší popularity než původní SNMPv2p. V současnosti jako SNMPv2 označována právě tato (nejúspěšnější) varianta druhé generace protokolu.

SNMPv2u (User-Based SNMPv2) – alternativa k SNMPv2c, užívající systému uživatelů. Je považován za jednodušší než SNMPv2p, ale stále bezpečnější než SNMPv2c. Je definován v RFC 1909 a 1910.

[4] Z angl. *Internet Engineering Task Force* – je mezinárodní organizace zabývající se standardizací internetových a síťových technologií.

[5] Z angl. *Request for Comments* – je název dokumentů vydávaných IETF, technicky specifikujících internetové standardy.

SNMPv2* – varianta kombinující prvky SNMPv2p a SNMPv2u. Definován soukromým výrobcem, nikdy nebyl formálně standardizován IETF.

I tato nekonzistentnost ve verzích SNMPv2 vedla k tomu, že mnoho výrobců i administrátorů zůstalo u užívání SNMPv1 [15, 16, 17].

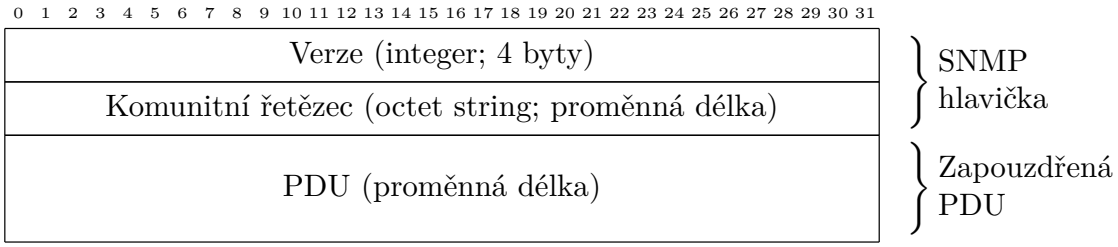
SNMPv3

Poslední verze protokolu, využívající volitelně autentizace pomocí jména a hesla, i volitelné šifrování. Přejímá *User-Based Security Model* (USM) – uživatelský systém z SNMPv2u, a přidává nově *View-Based Access Control Model* (VACM) – systém vytváření tzv. pohledů. Jedná se o uživatelsky definované sady MIB objektů, které mohou být přiřazeny jednotlivým uživatelům či skupinám. Těm jsou poté omezena přístupová práva pouze na objekty obsažené v tomto pohledu. Struktura PDU je shodná jako u SNMPv2. SNMPv3 je definován v RFC 3411–3418 [15, 16, 17].

Ačkoli je SNMPv3 poslední verzí protokolu, i v současnosti řada síťových prvků, včetně jednotek mikrovlnných spojů, podporuje pouze SNMPv1 či SNMPv2c. Následující text práce tedy pojednává výhradně o těchto verzích, založených na ověřování textovým řetězcem *community string*.

2.3 Struktura datových jednotek SNMP a PDU

Struktura SNMP paketu je graficky znázorněna na obrázku 2.2.



Obr. 2.2: Struktura SNMPv1 a SNMPv2c paketu.

Paket sestává z následujících polí:

- **Verze** – celočíselná hodnota o velikosti 4 byty. Pro SNMPv1 nabývá hodnoty 0, pro SNMPv2c hodnoty 1.[6]
- **Komunitní řetězec** – textový řetězec proměnné velikosti. V praxi je maximální délka řetězce omezována výrobci, např. na zařízeních spol. Cisco je to 32 znaků [18]. Slouží k jednoduchému ověřování identity.
- **PDU** (Protocol Data Unit) – datová jednotka proměnné velikosti, zapouzdřující konkrétní typ SNMP dotazu, odpovědi, či trap [15].

2.3.1 PDU

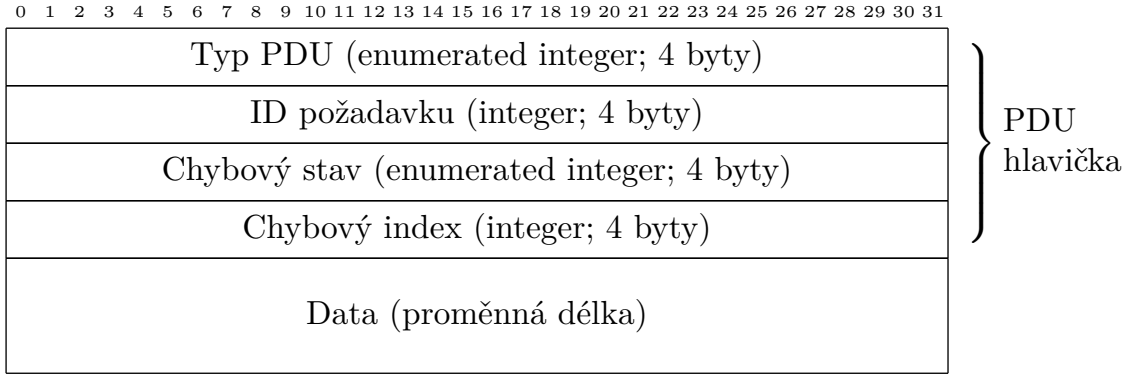
PDU má charakter samostatné datové jednotky zapouzdřené uvnitř SNMP paketu. Typy dotazů a odpovědí, zapouzdřených v PDU, jsou následující:

- **GetRequest** – dotaz na hodnotu proměnné specifikované zadaným OID (identifikátorem objektu v MIB stromu).
- **GetNextRequest** – dotaz na hodnotu proměnné následující po zadané OID ve struktuře MIB. Pomocí tohoto dotazu lze postupně projít celým stromem MIB, případně jeho částí, bez znalosti konkrétních OID.
- **GetBulkRequest** – přidán ve verzi SNMPv2. Lze si představit jako několik iterací `GetNextRequest` za sebou, kromě OID se zadává i požadovaný počet kroků. Vrací celou množinu hodnot proměnných následujících po zadané OID. Slouží jako zefektivnění `GetNextRequest` při procházení většího množství dat.
- **Response** – odpověď agenta s hodnotou proměnné (či proměnných).
- **SetRequest** – požadavek na nastavení hodnoty zadaného OID objektu. Objekt musí být zapisovatelného typu.

[6] Pro SNMPv2p je hodnota pole 2, pro SNMPv3 hodnota 3. Tyto verze však mají zbytek struktury paketu odlišný.

- **Trap** – zpráva zasílaná asynchronně agentem (bez vyžádání) na předem nakonfigurovanou adresu managera. Informuje o různých událostech nastalých na agentově zařízení.
- **InformRequest** – přidán ve verzi SNMPv2. Stejná funkcionalita jako Trap, vyžaduje však potvrzení o přijetí od managera pomocí PDU typu Report.
- **Report** – potvrzení (acknowledge) na `InformRequest` zaslaný agentem [15].

Struktura datové jednotky PDU je znázorněna na obrázku 2.3. Pro typ PDU `GetBulkRequest` a původní PDU `Trap` SNMPv1 je užívána samostatná mírně odlišná struktura od té uvedené na obrázku.



Obr. 2.3: Struktura datových jednotek PDU SNMPv1 a SNMPv2c.

Datová jednotka PDU sestává z následujících polí:

- **Typ PDU** – celočíselná hodnota o velikosti 4 byty. Výčtová hodnota, nabývá hodnot 0–8, každé je přiřazena příslušná PDU.
- **ID Požadavku** – celočíselná hodnota o velikosty 4 byty. Jedná se o číslo generované managerem, sloužící k identifikaci odpovědí zasílaných agentem. Agent kopíruje toto pole a vkládá jej do PDU Response.
- **Chybový stav** – celočíselná hodnota o velikosti 4 byty. Výčtová hodnota, nabývá hodnot 0–18, každé je přiřazen příslušný chybový kód. 0 označuje bezchybný stav (v dotazech zasílaných managerem má vždy hodnotu 0).
- **Chybový index** – celočíselná hodnota o velikosti 4 byty. Jedná se o ukazatel na objekt, který vygeneroval chybový kód (v dotazech zasílaných managerem má vždy hodnotu 0).
- **Data** – hodnota proměnné velikosti, obsahující identifikátor OID, a v případě PDU `Response`, i hodnotu objektu [15].

2.4 Datové typy

Návratová hodnota PDU `Response` může nabývat různých datových typů. Datové typy jsou definovány v tzv. SMI (z angl. *Structure of Management Information*), specifikované v RFC 1155 pro SNMPv1 (verze SMIv1) a v RFC 2578 pro SNMPv2 (verze SMIv2). SMI podrobně určuje způsob notace objektů v databázi objektů MIB. Seznam podporovaných datových typů je uveden v tabulce 2.1 [16].

Kromě datových typů uvedených v tabulce 2.1 existují i speciální typy označované jako textové konvence (*textual conventions*). Jedná se o typy vyšší úrovně, složené z dílčích základních typů. Jsou využívány například pro přenos adresních hodnot IPv6, které nemají vlastní základní datový typ. Užívání textových konvencí je podrobně specifikováno v RFC 2579 (pro SMIv2) [15].

Tab. 2.1: Přehled datových typů objektů v SNMPv2 [15, 16, 17]

Datový typ	Popis	Obsaženo v	
		SMIv1	SMIv2
`Integer32`	32b celočíselná hodnota se znaménkem. Nabývá hodnoty -2 147 483 648–2 147 483 647.	Ano	Ano
`Octet String`	Řetězec proměnné délky binárních či textových dat.	Ano	Ano
`Null`	Prázdný typ.	Ano	Ne
`Bits`	Výčtový typ pojmenovaných bitových sekvencí.	Ne	Ano
`Unsigned`	32b celočíselná hodnota bez znaménka. Nabývá hodnoty 0–4 294 967 295.	Ne	Ano
`IpAddress`	IPv4 adresa, kódovaná jako 4B octet string.	Ano	Ano
`Counter32`	32b celočíselná hodnota bez znaménka. Nabývá hodnoty 0–4 294 967 295. Slouží jako čítač, jeho hodnota roste. Po překročení max. hodnoty se vrací zpět na 0.	Ano	Ano
`Gauge32`	32b celočíselná hodnota bez znaménka. Nanabývá hodnoty 0–4 294 967 295. Slouží jako čítač, jeho hodnota však může růst i klesat.	Ano	Ano
`TimeTicks`	32b celočíselná hodnota bez znaménka. Nabývá hodnoty 0–4 294 967 295. Zaznamenává uplynulý čas v setinách sekund.	Ano	Ano
`Opaque`	Datová struktura ASN.1, kódovaná jako octet string proměnné délky.	Ano	Ano
`Counter64`	64b celočíselná hodnota bez znaménka. Nabývá hodnoty 0–18 446 744 073 709 551 615. Slouží jako čítač, jeho hodnota roste. Po překročení max. hodnoty se vrací zpět na 0.	Ne	Ano

2.5 MIB

MIB (z angl. *Management Information Base* je databáze objektů, ke kterým lze pomocí SNMP přistupovat. Jedná se o hierarchický stromový jmenný prostor, sestávají z identifikátorů objektů, tzv. OID (z angl. *Object Identifier*). OID jednoznačně identifikuje každou proměnnou, ze které lze číst či zapisovat. Je tvořena posloupností čísel oddělených tečkou, tato hodnota vznikne tak, že se vezme OID nadřazeného prvku a doplní se tečka a aktuální stromové číslo [16].

Pro samostatné získání dat od SNMP agenta, manager nepotřebuje mít znalost MIB databáze. Stačí mu pouze konkrétní OID, kterou vloží do příslušného dotazu. MIB je však klíčová pro identifikaci dat, jelikož obsahuje slovní popis OID i získaných hodnot. MIB je uložena v textovém souboru, používající zápis SMIv1, resp. SMIv2, vycházející z obecné specifikace ASN.1[7]. MIB lze číst buďto obyčejným textovým prohlížečem, nebo specializovanými nástroji, obsahujícími syntaktický analyzátor převádějící MIB do interaktivní stromové podoby.

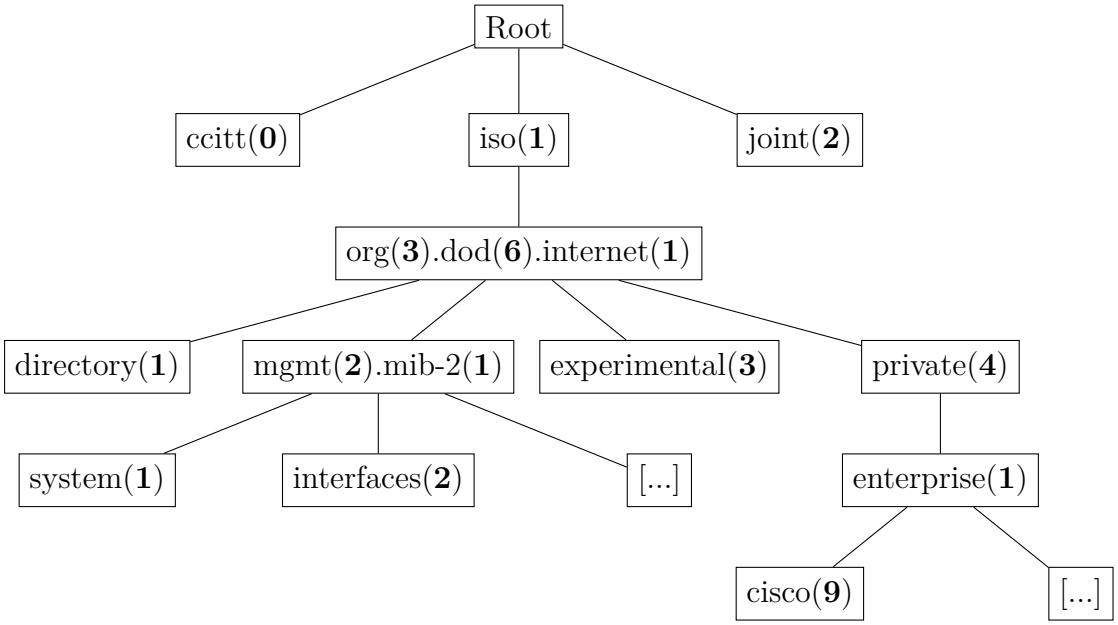
MIB musí zahrnovat generickou strukturu, označovanou jako MIB-II, definovanou v RFC 1213. Podpora tohoto RFC je povinná pro každé zařízení vybavené podporou SNMP. MIB-II specifikuje získávání různých systémových informací, nezávislých na typu zařízení. Základní specifikace sestává z deseti podstromů:

- **system** – systémové informace jako název systému, uptime, kontakt, umístění, atd.
- **interfaces** – informace o instalovaných síťových rozhraních, např. zda jsou aktivní, množství přenesených dat, aktuální datový tok atd.
- **at** – překlad adres, již se nepoužívá.
- **ip** – množství informací protokolu IP, například routovací tabulky.
- **icmp** – statistiky protokolu ICMP.
- **tcp** – statistiky protokolu TCP, tabulka navázaných spojení.
- **udp** – statistiky protokolu UDP.
- **egp** – statistiky protokolu EGP (zastaralý).
- **transmission** – objekty vztažené ke specifickým metodám přenosu na linkové vrstvě.
- **snmp** – statistiky protokolu SNMP [15, 16, 17].

Existuje celá řada dalších RFC, definujících doplňující MIB pro protokoly specifikované IETF. Například routovací protokoly RIP, OSPF, a další. Vedle obecných MIB definovaných v RFC existují, i tzv. *enterprise* MIB. Jedná se o MIB sestavené výrobci hardwaru, zahrnující definici parametrů specifických pro konkrétní modely zařízení výrobce.

[7]Standard ITU-T pro popis datových struktur v telekomunikacích a počítačových sítích.

MIB všech výrobců hardwaru se nacházejí v podstromu `1.3.6.1.4.1`, neboli `iso.org.dod.internet.private.enterprise`. Čísla stromů jsou výrobcům přidělována organizací IANA[8]. Na obrázku 2.4 je ukázka kořenové struktury MIB specifikované SMIv1, s vybranými příklady podstromů ve větvích `1.3.6.1.2.1` (MIB-II) a `1.3.6.1.4.1` (enterprise). U každého objektu je zobrazen název podstromu a v závorce jeho číselný identifikátor.



Obr. 2.4: Kořenová struktura MIB dle SMIv1 [17].

Pro příklad uveďme OID `1.3.6.1.2.1.1.3`, součást MIB-II, označující hodnotu systémového uptimu. Ekvivaletní zápis OID ve formě umístění ve stromové struktuře je:
`iso.org.dod.internet.mgmt.mib-2.system.sysUpTime`. Ukázka definice objektu v souboru MIB je zobrazena ve výpisu 2.1. `SYNTAX` definuje datový typ, `ACCESS` zda je hodnota pro čtení či zápis, `STATUS` zda je implementace objektu povinná, a `DECRIPTION` obsahující slovní popis objektu.

[8]Z angl. *Internet Assigned Numbers Authority*, organizace přidělující veřejné IP adresy, čísla autonomních systémů, a další prostředky nutné pro chod Internetu.

Výpis 2.1: Definice objektu sysUpTime v MIB-II dle RFC 1213 [17].

```
sysUpTime OBJECT-TYPE
            SYNTAX  TimeTicks
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The time (in hundredths of a second)
                    since the network management portion
                    of the system was last re-initialized."
            ::= { system 3 }
```

2.5.1 Datové struktury v MIB

Jednotlivé OID mohou být součástí vyšších datových struktur:

- skalár – obyčejná proměnná, nezávislá na jiných objektech. OID je v MIB specifikována plnou cestou.
- sloupec – struktura sestávající z množiny skalárních veličin. MIB neobsahuje konečné indexy OID jednotlivých prvků množiny, jelikož jsou obvykle dynamicky vytvářené (např. indexy rozhraní). Dynamické indexy lze získat dotazy GetNextRequest, případně GetBulkRequest. Těmito příkazy lze přečíst celou sloupcovou strukturu.
- tabulka – struktura sestávající z množiny sloupcových struktur.

2.5.2 MIB mikrovlnných jednotek

Výstupní aplikace této práce bude testována na mikrovlnných jednotkách výrobců Summit Development a Ceragon. Obě dvě společnosti vydávají vlastní *enterprise* MIB pod identifikátory:

- `iso.org.dod.internet.private.enterprises.summit-development` (1.3.6.1.4.1.23688),
- `iso.org.dod.internet.private.enterprises.ceragon` (1.3.6.1.4.1.2281).

Vybrané objekty z těchto MIB, potenciálně vhodné pro nasazení ve vyvíjené aplikaci, jsou pro spoje Summit uvedeny v příloze D této práce [19] a v příloze E pro spoje Summit modelové řady BT [20].

3 Užité technologie

Tato kapitola pojednává o technologických nástrojích, jako jsou aplikační rámce, knihovny, databázové systémy a API, využitých při tvorbě aplikace.

3.1 .NET Framework

.NET Framework je aplikační rámec společnosti Microsoft pro její vlastní operační systém Microsoft Windows. Sestává z:

- programovacího prostředí (IDE) – Microsoft Visual Studio,
- virtuálního stroje – CLR, provádějícím překlad z bajtkódu CIL do strojového kódu,
- knihoven – pokrývajících celou řadu aplikačních funkčností,
- podpory několika programovacích jazyků – nejrozšířenějším je jazyk C#, vytvořen přímo společností Microsoft pro potřeby platformy .NET.

Programy napsané v .NET Frameworku jsou kompilovány do tzv. bajtkódu (v angl. *byte code*). Tento kód je po stránce čitelnosti a úrovně abstrakce na pomezí mezi vyšším a nižším programovacím jazykem. Každá .NET aplikace je uložena a distribuována v tomto kódu. Virtuální stroj CLR (z angl. *Common Language Runtime*) při spuštění aplikace začne provádět její kompilaci do strojového kódu, a spustí ji ve svém chráněném prostředí. Tento model je výhodný díky své bezpečnosti a usnadněné správě paměti pro programátora. CLR v pravidelných intervalech spouští činnost tzv. garbage collectoru – komponenty zajišťující odstraňování již nepotřebných objektů z paměti. Odpadá tak nutnost alokace a dealokace paměťových bloků programátorem.

Počítače spouštějící programy psané v .NET Frameworku, musí mít v systému nainstalován příslušný softwarový balík s podporou běhu CLR, označovaný jako .NET Framework Runtime. Zatímco v dřívějších dobách tato skutečnost mohla uživatele vystavit problémům, v současnosti je již základní verze balíku integrována přímo do systému Windows a průběžně aktualizována společně se systémovými aktualizacemi.

.NET Framework lze použít pro tvorbu konzolových i grafických aplikací. Pro tvorbu grafických uživatelských rozhraní je vybaven knihovnou WPF (z angl. *Windows Presentation Foundation*), případně starší již nevyvíjenou knihovnou Windows Forms.

V roce 2016 Microsoft vydal nový multiplatformní framework .NET Core, vycházející z .NET Framework, umožňující vývoj aplikací i pro systémy Linux a Apple macOS. Existuje také odnož .NET Xamarin (dříve Mono), pomocí které lze vyvíjet

aplikace i pro mobilní platformy Android a iOS. Všechny tři frameworky sdílí společný knihovní základ a API[1], zvaný .NET Standard, což umožňuje vytváření a používání kompatibilních knihoven napříč platformami. Bohužel, .NET Core v současnosti není vybaven multiplatformní knihovnou pro tvorbu grafického uživatelského rozhraní [22, 23].

3.2 InfluxDB

InfluxDB je otevřená databázová platforma pro ukládání a práci s daty časové řady (v angl. *time-series data*). Vyznačuje se vysokým výkonem a účinnou kompresí těchto dat. Je uvolněna pod licencí MIT. Díky své odlišnosti od klasických databázových systému (MySQL, MS SQL, atd.) a specifickému zaměření bývá řazena do kategorií tzv. NoSQL databází. Využívá vlastního dotazovacího jazyka InfluxQL, který v základech vychází z klasického SQL[2]. Pro komunikaci mezi databázovým strojem a databázovými klienty je možno využívát integrované HTTP[3] API. [24, 25].

Aktuálně je platforma vyvíjena ve dvou verzích:

- 1.x (poslední verze 1.8) – stabilní verze, jednotlivé součásti platformy jsou distribuovány zvlášť,
- 2.0 beta – nová, testovaná verze, integruje všechny součásti do jediného kompaktního celku. [25].

Jednotlivými součástmi uvedené platformy InfluxDB jsou:

- InfluxDB – samotný databázový stroj,
- Telegraf – agent pro vyčítání dat, spolupracuje s různými API,
- Chronograf – grafický prohlížeč databázových dat,
- Kapacitor – nástroj pro datovou analýzu [25].

Důležitými funkcemi InfluxDB je tvorba tzv. retenčních politik (RP, *retention policy*) a kontinuálních dotazů (CQ, *continuous query*).

Retenční politiky udávají, po jak dlouhou dobu mají v dané části databáze zůstávat nashromážděná data. InfluxDB v pravidelných intervalech provádí porovnání aktuálního času a časů zaznamenaných dat, přičemž položky překračující stanovenou retenční dobu jsou mazány. Pro jednu databázi může být vytvořeno několik RP [26].

Kontinuální dotazy jsou automaticky spouštěny v pravidelných intervalech. Nejčastěji jsou užívány k převodu dat mezi retenčními politikami, kdy data obsažená v jedné RP s krátkou retenční dobou, ale vysokou vzorkovací frekvencí, zprůměrují na nižší vzorkovací frekvenci, a uloží je do druhé RP s vyšší retenční dobou [26].

[1] Z angl. Application Programming Interface – rozhraní aplikace, knihovny, či jádra operačního systému, pro komunikaci s jinou aplikací.

[2] Z angl. Structured Query Language – dotazovací jazyk relačních databází.

[3] Z angl. Hypertext Transfer Protocol – protokol aplikační vrstvy pro přenos informací.

3.3 SQLite

SQLite je kompaktní systém relační databáze, jehož databázový stroj je distribuován přímo s aplikací která jej využívá, prostřednictvím přiložené knihovny. Je nenáročný a bezúdržbový – nevyžaduje žádnou konfiguraci ze strany uživatele. Celá SQLite databáze je na disku uložena v jediném souboru [27].

SQLite je také pravděpodobně nejrozšířenějším databázovým systémem na světě. Je integrovaný nejen v desktopových aplikacích, ale pro svou jednoduchost především v mnoha mobilních platformách a vestavěných systémech [27].

3.4 Knihovny

3.4.1 InfluxDB Client for .NET

InfluxDB Client for .NET doplňuje InfluxDB HTTP klienta do .NET aplikace. Autorem knihovny je Mikael Guldborg Rask Andersen, uvolněna je pod licencí MIT, jmenný prostor knihovny je `Vibrant.InfluxDB`. Práce s daty může probíhat pomocí klasických modelových C# objektů s nastavenými knihovními indexy, či pomocí dynamických tříd, vhodných při zpracování dat s předem neznámou strukturou. Metody knihovny jsou asynchronní. Knihovna je k dispozici na serveru GitHub či v integrovaném balíčkovacím systému NuGet [28].

3.4.2 SQLite-net

SQLite-net je knihovna umožňující práci v databázemi SQLite v prostředí .NET. Autorem knihovny je Frank A. Krueger, uvolněna je pod licencí MIT, jmenný prostor knihovny je `SQLite`. Stejně jako knihovna InfluxDB Client for .NET využívá modelových objektů C# doplněných o databázové indexy. Podporuje synchronní i asynchronní volání. Knihovna je k dispozici na serveru GitHub či v systému NuGet [29].

3.4.3 SharpSNMP

SharpSNMP je knihovna autora Lex Li, doplňující do .NET funkcionalitu zasílání a přijímání SNMP PDU a jejich snadné zpracování. Disponuje podporou všech verzí SNMP protokolu. Disponuje synchronními i asynchronními metodami, jmenný prostor knihovny je `Lextm.SharpSnmpLib`. Je uvolněna pod licencí MIT a k dispozici ke stažení na serveru GitHub či v systému NuGet [30].

3.4.4 LiveCharts

LiveCharts je knihovna pro .NET aplikace, poskytující možnost vytváření grafických průběhů a vizualizace dat. Umožňuje vykreslování několika typů grafů. Autorem knihovny je Alberto Rodríguez, základní verze je uvolněna pod licencí MIT. Jmenný prostor knihovny je `LiveCharts`. Knihovna je k dispozici na serveru GitHub či v systému NuGet [31].

3.5 Webová API

3.5.1 OpenWeatherMap

OpenWeatherMap je webové API poskytující celosvětová data o počasí. V bezplatném plánu umožňuje provádět až 60 dotazů za minutu, a to pro libovolnou zeměpisnou souřadnici.

Nevýhodou API je neposkytování informací o srážkovém úhrnu na území ČR (získáváno je pouze ID počasí s informací o stupni intenzity deště, nikoli konkrétní hodnota srážkového úhrnu).

Pro použití API je nutné zaregistrovat se na stránkách poskytovatele a získat osobní přístupový klíč [32].

3.5.2 MapQuest Open Elevation API

MapQuest Open Elevation je bezplatné webové API poskytující informace o terénním profilu. Pro dotazované pole souřadnic navrací jejich nadmořskou výšku, či z daných dat nadmořské výšky přímo dokáže vykreslit grafický průběh terénu.

Pro toto API, je v případě zjišťování terénního profilu mezi dvěma souřadnicovými body, nutno dopočítat souřadnice mezi nimi. Výhodou je však, v tomto případě výpočtu mezilehlých souřadnic na straně klienta, možnost volby i velmi vysokého rozlišení grafického průběhu (v závislosti na počtu dopočítaných souřadnic).

I pro toto API je nutné získat přístupový klíč pomocí bezplatné registrace [33].

3.5.3 Mapy.cz API

API Mapy.cz je JavaScriptové API, pro vykreslování symbolů a informací na mapových podkladech. API je zcela zdarma a podporuje vykreslení základních jednoduchých značek bez popisků i s popisky, jakož i vykreslování složitějších vektorových obrazců.

Pro jeho užití je nutno vytvořit webovou stránku, načíst poskytnutý JavaScriptový skript, a pomocí vlastního skriptu specifikovat žádoucí zobrazení [34].

4 Návrh aplikace

Při návrhu aplikace bylo koncepčně čerpáno z již existujících řešení pro monitoring síti a jejich správu, využívající protokolu mimo jiné protokolu SNMP. Z volně dostupných aplikací je to například nástroj MRTG[1] pro vyčítání a grafické vykreslování veličin, na něm založený Cacti, či dohledový systém Zabbix. Z komerčních proprietárních aplikací pak dohledový systém PRTG[2] společnosti Paessler či desktopová aplikace SNMPc[3] společnosti Castle Rock Computing. Jedná se o všestranné, bohatě konfigurovatelné aplikace, umožňující monitoring parametrů všech typů, a to na tisících zařízeních současně [35, 36, 37, 38].

Často je využíván návrhový koncept vzdálené sondy, provádějící zasílání SNMP dotazů a komunikaci s aplikačním serverem, který disponuje webovým uživatelským rozhraním (PRTG, Zabbix). Případně koncept serveru a desktopového[4] klienta, pro zobrazení vyčtených dat a stavů.

Zmíněné nástroje jsou však značně univerzálního charakteru. Některé z nich sice lze rozšířit o vlastní funkce pomocí přídavných rozšíření (tzv. „plugins“ či „addons“), možnosti jsou však i v tomto případě omezené. Specializovaná volně dostupná či prodejná řešení pro dohled mikrovlnných spojů nejsou dostupná. Z části i proto, že výrobci mikrovlnných spojů představují svá vlastní dohledová řešení, ta jsou však často nekompatibilní se spoji ostatních výrobců.

4.1 Koncept

Bylo rozhodnuto aplikaci vytvořit jako kompaktní celek, tvořený desktopovou aplikací, integrující jádro s vnitřní logikou i grafické uživatelské rozhraní. Toto rozhodnutí bylo učiněno z důvodu vysoké komplexity a nároků na zabezpečení v případě realizace „rozděleného“ konceptu, sestávajícího ze sondy/serveru a odděleného uživatelského klienta, resp. webového prostředí.

Pro aplikaci se předpokládá nasazení u malých či středních poskytovatelů služeb mikrovlnných spojů; na fyzickém či virtualizovaném stroji, připojeném do managementového sektoru sítě poskytovatele (přímo, či skrze VPN). Operátor dohledového centra může ke stroji přistupovat fyzicky, či prostřednictvím vzdálené plochy.

[1] Multi Router Traffic Grapher.
[2] Paessler Router Traffic Grapher.
[3] SNMP Console.
[4] Aplikace s grafickým uživatelským rozhraním (GUI) spouštěná na uživatelském počítači.

4.2 Realizace

Pro vývoj je jako základ použit .NET Framework, s kódem psaným v jazyce C#. Grafické rozhraní je realizováno prostřednictvím systémových knihoven WPF. Jádro aplikace sestává z objektů tvořících celek SNMP managera. Dále z objektů, zajišťujících ukládání dat do databázového systémů, zpětné načítání dat, jejich analýzu, práci s konfiguračními soubory atd. Grafické uživatelské rozhraní obstarává prezentaci získaných dat formou grafických průběhu v reálném čase, a informuje uživatele o vytvořených alarmech.

Jako alarmy jsou v aplikaci označovány notifikace pocházející z analyzátorů, informující uživatele o podezřelých nálezech ve sledovaných veličinách, vyčítaných ze zařízení. Mají danou strukturu a mohou nabývat různých priorit. Uživatel má možnost alarm potvrdit, čímž se přesune do příslušné sekce.

Monitorované mikrovlnné jednotky jsou spravovány a zobrazovány v rámci logických skupin – linek (spojů), jejichž namapování v aplikaci odráží reálné dispozice spoje. Výchozí konstelací jsou dvě monitorované jednotky v rámci jedné linky, tedy klasický dvoubodový spoj typu bod-bod. Aplikace však pamatuje i na možnost retranslace, tj. mikrovlnné spoje s větším počtem skoků (viz kapitola 1.2). Pro tento případ je možno do jedné logické linky namapovat dvě koncové stanice a dvě retranslační stanice, přičemž každá retranslační stanice sestává ze dvou mikrovlnných jednotek. Dohromady je tedy možné v rámci jedné linky spravovat až šest zařízení. Zařízení jsou značena jako:

- `A` a `B` – koncové stanice,
- `R1`, `R2`, `R3`, `R4` – retranslační stanice.

Linky namodelované v aplikaci, resp. jejich jednotlivá zařízení, je možné ze strany uživatele libovolně přidávat, mazat, či pozastavovat jejich monitoring.

Stěžejní částí aplikace je vyčítání hodnot veličin a jejich následná analýza. V aplikaci je definováno několik veličin charakteristických pro mikrovlnné jednotky, které lze vyčítat a sledovat:

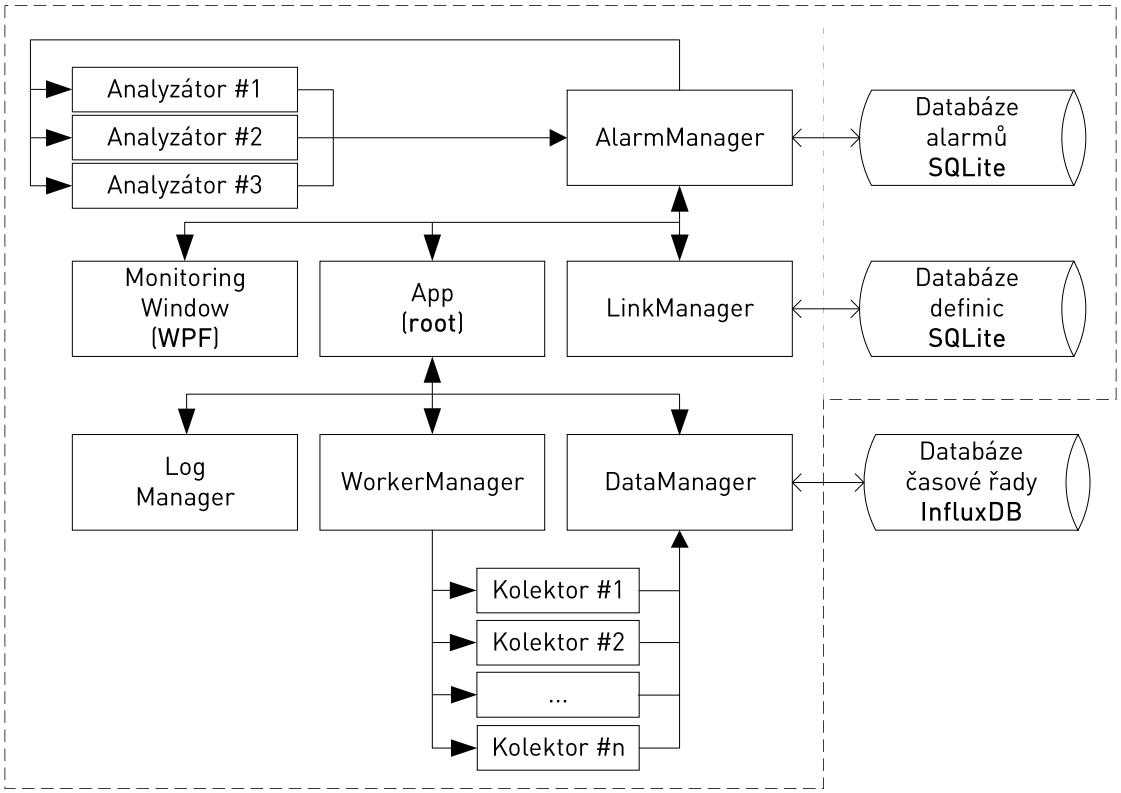
- `síla signálu`; `kvalita signálu`; `teplota ODU`; `teplota IDU`;
- `napájecí napětí`; `latence`; `datový přenos Tx`, `datový přenos Rx`.

Dále je automaticky pro každé zařízení vyčítán uplynulý čas od startu jednotky (tzv. uptime) a název jednotky. Pokud výrobce neposkytuje možnost vyčítání některé veličiny pomocí SNMP, lze monitorování této veličiny u dané mikrovlnné jednotky vypnout. Pro každou veličinu lze nastavit obnovovací frekvenci, s jakou se má daná veličina vyčítat. Nastavování parametrů probíhá pro každé zařízení zvlášť, je tak možno provádět individuální změny v konfiguraci. V globálním nastavení aplikace lze konfigurovat všechny důležité parametry analyzátorů dat, jako jsou procentuální prahy, velikosti časových oken apod., případně povolovat režim ladění.

5 Jádro aplikace

Diagram hlavních aplikačních objektů a vztahů mezi nimi je zobrazen na obrázku 5.1. Objekty aplikace jsou členěny do několika základních objektových skupin:

- **Managers** – tyto objekty, tzv. manažeři, zapouzdřují metody pro práci s daty, řídí vytváření kolektorů, či směrují aplikační výpisy.
  - `WorkerManager` – provádí vytváření, spouštění, a zastavování kolektorů.
  - `LinkManager` – zajišťuje práci s objekty `Link` a `Device`.
  - `DataManager` – provádí zápis dat časové řady, zjišťuje jejeich čtení.
  - `AlarmManager` – spouští a spravuje analyzátory, spravuje kolekce alarmů.
  - `LogManager` – spravuje výpis událostí a oznámení v aplikaci.
- **Workers** – neboli kolektory, zajišťující vyčítání dat ze vzdálených zařízení a stahování informací.
- **Analysers** – provádí analýzu vyčtených dat a vytváří alarmy.
- **Models** – jedná se oobjekty databázového typu, úložiště informací.
- **GUI** – objekty provázané s WPF, komponenty uživatelského rozhraní.



Obr. 5.1: Diagram ústředních objektů a jejich vztahů.

5.1 Pracovní vlákna

Každé běžící aplikaci je operačním systémem přidělován určitý procesorový čas – operační systém při zpracovávání úloh běžících aplikací mezi těmito aplikacemi v pravidelných intervalech přepíná,[1] čímž je zdánlivě zajištěn současný chod více aplikací, než je počet procesorů či procesorových jader v počítači (tzv. multitasking). U většině současných aplikací však nastává potřeba, rozdělit i chod samotné aplikace do několika tzv. vláken, mezi jimiž zpracováváním je přepínáno stejně jako v případě zmíněného multitaskingu. Na samostatném vláknu obvykle běží uživatelské rozhraní aplikace, na jiném vláknu pak samotná výpočetní logika aplikace. Při čekání na zpracování výpočetních úloh by totiž hrozilo zablokování prostředí aplikace před uživatelem, tzv. zamrznutí, do doby, než by byly tyto úlohy zpracovány. Na samostatná vlákna je tedy nutno delegovat tyto typy úloh [22]:

- výpočetně vázané úlohy, tzv. „CPU bound“,
- vstupně/výstupně vázané úlohy, tzv. „I/O bound“,
  - čtení objemných dat z disku,
  - čekání na odezvu vzdáleného zařízení na síti,
  - naslouchání na síti apod.

V této aplikaci je nutno využít některé základní principy tzv. asynchronní architektury, jelikož některé objekty, zejména kolektory, a také některé typy analyzátorů, jsou výrazně vstupně-výstupně vázané. Kolektory jsou sběrače dat, které zašlou dotaz, a následně několik jednotek až stovek milisekund naslouchají na síti a čekají na odpověď (doba závisí na síťové latenci[2] mezi aplikací a vzdáleným zařízením, a také času zpracování dotazu cílovým zařízením).

Jelikož SNMP využívá transportního protokolu UDP, který nezaručuje doručení dat, a potvrzení na úrovni aplikační vrstvy je vyžadováno pouze v případě PDU `Inform` (viz kapitola 2), může se stát, že dotaz (či odpověď na něj) není cíli doručen vůbec, případně může být protějším zařízením zahozen (při vysoké zátěži dotazovaného zařízení). V tom případě kolektor blokuje vlákno do doby, než vyprší nastavený časový limit pro čekání na odpověď zaslanému dotazu (tzv. timeout), nebo do doby, kdy je vlákno uměle přerušeno jiným vláknem.

Vedle SNMP kolektorů je v samostatných vláknech nutno spouštět také kolektory zasílající ICMP[3] dotazy `Echo request`, čekající na odpověď `Echo replay`.[4] Vlastní vlákna si v aplikaci vytváří také některé analyzátory, u kterých se jedná o kombinaci faktoru vstupně/výstupně vázané úlohy (čekání na čtení dat z databáze), i výpočetně náročné úlohy.

[1] V systému Windows je tento interval typicky 20 ms [22].

[2] Doba, za kterou paket urazí cestu do cíle a zpět.

[3] Internet Control Message Protocol – protokol síťové vrstvy, slouží k zasílání řídících zpráv sítě.

[4] Tzv. ping test, sloužící k měření latence a testu dostupnosti cíle.

5.1.1 Omezení

Vlákna však nelze vytvářet zcela bez omezení. Každé vytvořené vlákno si alokuje určitou část systémových prostředků, a operační systém musí vykonat určitou režii při přepínání mezi jednotlivými vlákny. S přibývajícím počtem vláken se může zvyšovat využití procesoru a odezva aplikace se tak bude zpomalovat. V některých systémech také může být dosažen strop počtu vláken, kterých lze v rámci jednoho procesu vytvořit. U některých typů úloh může být vytváření většího než optimálního počtu vláken dokonce i neefektivní [39]. Řešením je řetězení úloh v omezeném počtu vláken, a využívání technik „thread poolingu“, neboli vytváření fondu vláken [22, 39].

V této aplikaci je nyní využito pouze jednoduchého principu vytváření vláken, což v praxi znamená, že pro každý kolektor je vytvořeno samostatné vlákno. Při množství sledovaných zařízení cca do počtu 150, se nejedná o problém, při vyšším by již bylo nutno implementovat výše uvedenou techniku thread poolingu.

Toto řešení má však i své výhody. Je vyloučeno vzájemné blokování se kolektorů, a fakt, že každý kolektor má své vlastní vlákno, umožňuje nastavení i velice krátkých obnovovacích intervalů. Během testování aplikace tak mohla mít velká část kolektorů nastaven obnovovací interval na dobu 1 sekundy.

5.1.2 Thread safety

Při vytváření vícevláknové aplikace je nutno dodržovat principy tzv. „thread safety“. Pakliže dvě vlákna ve stejný okamžik přistupují k některému sdílenému objektu, např. kolekci, a jedno z vláken změní obsah této kolekce, je vyvolána výjimka. Proto je nutno zajistit jednotlivým vláknům ke sdíleným objektům exkluzivní přístup.

K tomu je využíván tzv. synchronizační objekt, a klíčové slovo `lock`. Před přístupem do sdíleného objektu vlákno zavolá příkaz `lock` se synchronizačním objektem jakožto argumentem. Následně vykoná blok kódu vyžadující exkluzivní přístup. Synchronizační objekt je po celou tuto dobu uzamknut, přičemž zavolá-li jiné vlákno také příkaz `lock`, se stejným objektem jakožto argumentem, je zařazeno do fronty a čeká, až první vlákno vykoná svůj kód [22].

5.2 Manažeři

5.2.1 `WorkerManager`

Třída zajišťuje startování, ukončování, a inicializaci kolektorů, které uchovává v kolekci `workers`. Důležitými metodami jsou:

- `InitWorkers` – provede inicializaci a spuštění kolektorů pro všechna zařízení v kolekci `devices` předané parametrem. Provádí také inicializaci kolektoru počasí. Je volána pouze při startu programu.
- `InitDevice` – provede inicializaci a spuštění kolektoru pro jedno konkrétní zařízení.
- `StartDevice` – spouští kolektory již inicializovaného zařízení.
- `StopDevice` – ukončuje kolektory spolu s jejich vlákny pro předané zařízení.
- `RemoveDevice` – volá `StopDevice` a poté provede deinicializaci zařízení.
- `RestartDevice` – restartuje kolektory předaného zařízení.

Parametrem všech metod s výjimkou metody `InitWorkers` je model zařízení `Device`.

Třída dále uchovává slovníkovou veřejnou kolekci typu `<int, DeviceDisplay>` s názvem `DeviceToFront`, kde index tvoří ID zařízení. Typ `DeviceDisplay` je model uchovávající poslední vyčtené hodnoty, které do něj s každou novou hodnotou ukládají kolektory. Tato data jsou poté využívána uživatelským prostředím a analyzátory. Kolekce tedy slouží jako jednosměrný komunikační prostředek mezi kolektory a zbytkem aplikace.

5.2.2 `LinkManager`

Třída zajišťuje čtení a zápis dat z databází SQLite (viz 3.3). Aplikace využívá dvou databázových souborů:

- `DeviceData.db` – úložiště definicí zařízení a linek,
- `AlarmData.db` – úložiště dat všech vygenerovaných alarmů.

Názvy databází jsou pouze výchozí hodnoty, v případě potřeby je možno editovat příslušné položky v konfiguračním souboru `MicrowaveMonitor.exe.config` v sekci `connectionStrings`.

Databázové připojení je reprezentováno typem `SQLiteConnection` z knihovny SQLite-net, který je ve třídě vytvořen ve dvou privátních instancích pro dvě výše uvedené databáze, a to pod názvy `LinkDatabase`, resp. `AlarmDatabase`.

Konstruktor třídy na těchto objektech volá pro každou tabulku, jíž model je metodě předán parametrem, metodu `CreateTable`, která v případě že daný databázový soubor neexistuje (nebo neobsahuje předanou tabulku), vytvoří nový databázový soubor se strukturou předané tabulky (resp. vytvoří pouze strukturu předané tabulky).

Pro přístup k záznamům v databázi třída obsahuje množství veřejných metod typu `Get...`, `Update...`, `Remove...` či `Find...`, dle potřeb volajících objektů.

5.2.3 `DataManager`

Třída v pravidelných intervalech zapisuje do InfluxDB[5] a obsahuje metody pro čtení z dané databáze. Pro přístup k databázi je využita knihovna Vibrant.InfluxDB.

Pro každou zapisovanou veličinu se ve třídě nachází veřejná kolekce typu fronta, do které mohou zapisovat všechny kolektory svá nejnovější vyčtená data. Konstanta `DbWriteInterval` pak definuje, v jakém intervalu jsou data zapisována do databáze. Hodnota je nastavena na 10 s

Parametry pro vytvoření připojení jsou přebírány z konfiguračního souboru ze sekce `connectionStrings`. Jedná se o položky:

- `InfluxServer` – adresa databázového serveru (výchozí `127.0.0.1`),
- `InfluxData` – název databáze (výchozí `MMDB`),
- `InfluxWriteRetention` – název retenční politiky pro zápis (výchozí `week`),
- `InfluxUser` – uživatelské jméno (výchozí `monitor`),
- `InfluxPass` – uživatelské heslo (výchozí `monitor`),
- `InfluxRetentionWeek` – název týdenní retenční politiky (výchozí `week`),
- `InfluxRetentionMonth` – název měsíční retenční politiky (výchozí `month`),
- `InfluxRetentionYear` – název roční retenční politiky (výchozí `year`).

Jak vyplývá z konfiguračních údajů, retenční politiky (RP) jsou pro InfluxDB nastaveny tři. Parametry jejich kontinuálních dotazů (*Continuous Queries*, viz kapitola 3.2) jsou:

- RP: 1 týden, výchozí RP pro zápis;
- RP: 30 dní, CQ pro většinu veličin činí 30 s,
- RP: 365 dní, CQ pro většinu veličin činí 1 m.

Třída `DataManager` dále obsahuje veřejné metody:

- `QueryRows` – vrací záznamy jedné řady hodnot,
- `QuerySeries` – vrací záznamy několika řad, při užití klauzule `GROUP BY`,
- `QueryValue` – vrací jediný záznam.

Parametrem všech tří metod je řetězec s databázovým dotazem.

[5] InfluxDB byla nasazena ve verzi 1.8.

5.2.4 `AlarmManager`

Třída spravuje kolekce zobrazení alarmů, vytváří nové alarmy a deaktivuje staré. Je jednou z nejobsáhlejších vytvořených tříd, obsahuje řadu obslužných privátních metod. Veřejné kolekce typu `ObservableCollection`, obsahující alarmovou strukturu zobrazení `AlarmDisplay`, jsou celkem čtyři:

- `alarmsCurrent`,
- `alarmsAck`,
- `alarmsSettledAck`,
- `alarmsSettledUnack`.

Kolekce jsou datově provázány s příslušnými panely se seznamy alarmů v uživatelském prostředí (viz 6.3).

Důležitými metodami třídy jsou `GenerateAlarm` a `SettleAlarm`. Tyto metody volají analyzátory (které jsou vytvářeny v konstruktoru této třídy) pro vytvoření nového, či deaktivaci již vytvořeného alarmu. Ihned po vytvoření alarmu je takovému přiděleno nové ID a je uložen do databáze `AlarmsData.db`, následně je vytvořeno jeho i zobrazení `AlarmDisplay` a přidáno do kolekce `alarmsCurrent`.

Třída ovládá analyzátory metodou `LoadSettings`, která čte nová uživatelská nastavení (5.3) a předává je analyzátorům.

Dále třída obsahuje i metody nazvané jako „triggery“, které jsou blíže popsány v kapitole 7.4.

5.2.5 `LogManager`

Tato třída slouží k přesměrování konzolových hlášení do notifikačního okna v uživatelském prostředí aplikace (viz 6.4). Zajišťuje také formátování výstupních zpráv dle typu hlášení. Umožňuje též případnou budoucí implementaci záznamů aplikace do textových souborů.

Třída dědí ze systémové abstraktní třídy `System.IO.TextWriter`, a přetěžuje její metody `Write`, `WriteLine`, `Flush`, a `Close`, a také specifikuje kódování znaků UTF-8 (požadavky abstraktní třídy). Do objektu `LogManager` je při startu aplikace přesměrován konzolový výstup. Třída pak s každým konzolovým hlášením odesílá přijatý řetězec na původní standardní konzolový výstup `Console.Out`, ale také volá metodu `AppendNotificationDispatch` WPF třídy `EventLogPane` (notifikační okno), a jako argument této metodě předává vytvořenou strukturu `LogRow`, obsahující kromě samotného hlášení i informace nutné k jeho naformátování (čas, úroveň, text hlášení, barvu času, barvu úrovně, barvu textu).

Aby bylo možné odlišit jednotlivá konzolová hlášení a řádně je naformátovat, na první pozici každého konzolového hlášení se nachází jednociferné ID, určující typ (úroveň zprávy). Pakliže se taková číslice na první pozici nenachází, je hlášení považováno za neošetřený konzolový výstup a formátováno jako výjimka – předpona `<EXCEPTION>`.

5.3 Uživatelská nastavení aplikace

Aplikace využívá integrovaného prostředí pro čtení a zápis konfiguračních dat v .NET Framework. Jedná se o třídu `System.Settings`. Je možné využívat dva typy nastavení: tzv. „application scope“ a „user scope“.

Application scope jsou nastavení společná pro všechny uživatele systému, pro která není předpoklad, že by docházelo k jejich častým změnám. Jedná se tedy o data nastavovaná například před prvním spuštěním aplikace. Tato nastavení jsou uložena v souboru v kořenovém adresáři aplikace, se stejným názvem, jako binární spustitelný soubor, avšak s přidanou příponou `.config`. V případě této aplikace je to tedy již zmíněný XML soubor `MicrowaveMonitor.exe.config`.

Mezi takto uložená data aplikace patří klíče API (počasí, výškový profil) a názvy retenčních politik InfluxDB. V souboru se nachází také sekce `connectionStrings`, obsahující přihlašovací údaje k databázím, jedná se však o samostatný druh dat, nespadající do application scope nastavení.

User scope jsou nastavení ukladaná zvlášť pro každého uživatele systému. Jejich výchozí hodnoty se nacházejí rovněž v souboru `MicrowaveMonitor.exe.config` (pod XML tagem `userSettings`), jakmile však uživatel provede změnu některého nastavení, jsou nové hodnoty zapsány do nově vytvořeného souboru, který je při každém spuštění aplikace načítán přednostně. Tento soubor nese název `user.config` a nachází se v umístění:

```
C:\Users\<jméno_uživatele>\AppData\Local\<jméno_vývojáře>\
<appdomainname>_<eid>_<hash>\<version>
```

kde:

- `jméno_vývojáře` – jméno společnosti nastavené ve vlastnostech projektu,
- `appdomainname` – obvykle název spustitelného `.exe` souboru,
- `eid` – nese hodnotu `Url`, `StrongName`, či `Path`, v závislosti na konfiguraci,
- `hash` – SHA1 hash entity označené parametrem `eid`,
- `version` – verze aplikace nastavená ve vlastnostech projektu [40].

Pro tuto aplikaci cesta vypadá následovně:

```
c:\Users\<jméno_uživatele>\AppData\Local\VUT\
MicrowaveMonitor.exe_Url_<hash>\1.0.0.0\user.config
```

Všechna user scope nastavení aplikace se týkají analyzátorů, rozebíraných podrobně v kapitole 7. Pro každý analyzátor jsou uvedeny i jednotlivé položky nastavení s výchozími hodnotami. Snímky konfiguračního okna se nacházejí v příloze A.

5.4 Kolektory

Základní stavební třídou kolektorů (s výjimkou kolektoru počasí) je abstraktní třída `Collector`. Tato třída implementuje obecné vlastnosti, jako `DeviceId`, `Address`, `Display` (model `DeviceDisplay` daného zařízení), `RefreshInterval`, atd. Obsahuje též definici vlákna kolektoru `tCollecotr`.

Dále obsahuje metody pro hlášení totálního výpadku a překročení nastavených prahů, které jsou blíže rozebírány v kapitole 7.4.

5.4.1 SNMP

Stavební třídou SNMP kolektorů je abstraktní třída `SnmpCollector`, dědící z abstraktní třídy `Collector`. Kolektory SNMP využívají pro zasílání dotazů knihovnu SharpSNMP ze které se používá především statická třída `Messenger` a její metoda `Get`, pro odesílání SNMP PDU `GetRequest`.

Knihovnu SharpSNMP bylo nutno upravit a překompilovat, jelikož ve standardním sestavení její metoda `Get` v případě, kdy vypršel časový limit pro odpověď protistrany (timeout), vyvolávala výjimku. Přesto, že na straně aplikace byla tato výjimka zachycována, toto chování vzhledem k počtu odesílaných a ztracených paketů (viz kapitola 9 věnující se měření provozu) aplikaci zatěžovalo, a to především v ladícím módu. Chování metody `Get` při vypršení čas. limitu tedy bylo nahrazeno návratem hodnoty `null`.

V případě přijmutí odpovědi na zaslaný SNMP dotaz, je volána abstraktní metoda `RecordData`. Ta je implementována jednotlivými koncovými SNMP kolektory příslušných veličin, dědícími z této abstraktní třídy `SnmpCollecotr`. Tato implementace je z důvodu odlišného zpracování vyčtených dat mezi jednotlivými SNMP veličinami, koncové SNMP kolektory veličin jinou, než přetíženou metodu `RecordData` neobsahují. `RecordData` provádí aktualizaci dat v modelu `DeviceDisplay` a data ukládá do transakční fronty objektu `DataManager`.

Koncové třídy SNMP kolektorů, implementující `RecordData`, jsou následující:

- `SnmpSignal`, `SnmpSignalQ` – síla signálu, kvalita signálu,
- `SnmpTempOdu`, `SnmpTempIdu` – teplota ODU, teplota IDU,
- `SnmpTx`, `SnmpRx` – datová rychlost Tx (vysílání) rádia, Rx (řijímání) rádia,
- `SnmpVoltage` – napájecí napětí,
- `SnmpUptime`, `SysName` – čas uběhlý od startu jednotky, název jednotky.

5.4.2 ICMP

Kolektor založený na protokolu ICMP, tvořený jednou třídou `PingCollector`, není „sběračem“ v pravém slova smyslu, jelikož žádná konkrétní data ze zařízení nesbírá. Provádí tzv. ping – měří pouze čas[6], který uběhne od odeslání ICMP datagramu `Echo Request` do přijetí odpovědi `Echo Reply`. K zasílání datagramů je využívána systémová třída `System.Net.NetworkInformation.PingReply`. Zbytek funkce je obdobný jako u kolektorů SNMP.

5.4.3 Počasí

Tento kolektor je implementován třídou `WeatherCollector`. Má zcela odlišný charakter od zbytku kolektorů zmíněných výše. Je vytvořen pouze v jediné instanci, dotazuje se sekvenčně po jednotlivých zařízeních, a data nezískává ze sledovaných zařízení, nýbrž stahuje z API serveru v internetu pomocí protokolu HTTP.

K implementaci byla využita knihovna OpenWeatherMap-API, od autora pod pseudonymem swiftyspiffy, volně dostupná pod licencí MIT [41]. Ta dále využívá pro zpracování stáhnutých souborů ve formátu JSON knihovnu Json.NET od společnosti Newtonsoft, dostupnou taktéž pod licencí MIT [42].

Aktualizace stavu počasí probíhá postupně po jednotlivých zařízeních. Jelikož volná verze API umožňuje zasílat maximálně 60 dotazů za minutu, je nutné mezi jednotlivými dotazy vytvářet y minimálně o délce 1 s. Aktualizace se provádí jednou za 5 minut, je-li aktualizace provedena před vypršením tohoto intervalu, po zbytek doby je vlákno uspáno. Trvá-li však aktualizace delší dobu (v případě, že aplikace sleduje velké množství jednotek), vlákno uspáváno není a nastává ihned další iterace cyklu.

Pro komunikaci s OpenWeatherMap API kolektor vytváří objekt `weatherApi` typu `OpenWeather` z výše zmíněné knihovny. Knihovna sestaví HTTP požadavek, sestávající z klíče API obsaženého v konfiguračním souboru aplikace, a zeměpisných souřadnic aktuálně zjišťovaného zařízení. Je-li požadavek validní, server odpovídá vygenerovaným JSON souborem obsahujícím aktuální data o počasí pro požadovanou lokalitu dle zadaných souřadnic, který je aplikací stažen a zpracován.

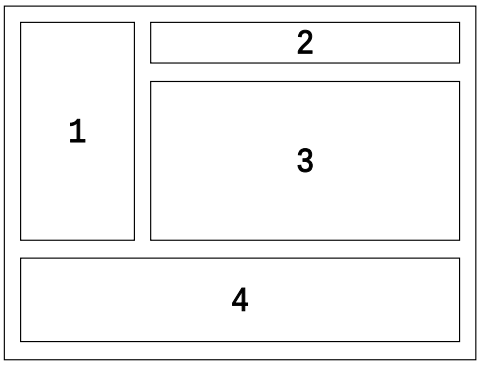
Kolektor zapisuje pro každé zařízení do databáze:

- teplotu vzduchu,
- ID počasí (viz tabulka 7.3),
- rychlost větru.

[6] Tzv. round-trip time.

6 Uživatelské rozhraní aplikace

Hlavní okno aplikace je sestaveno z funkčních sektorů znázorněných na obrázku 6.1:



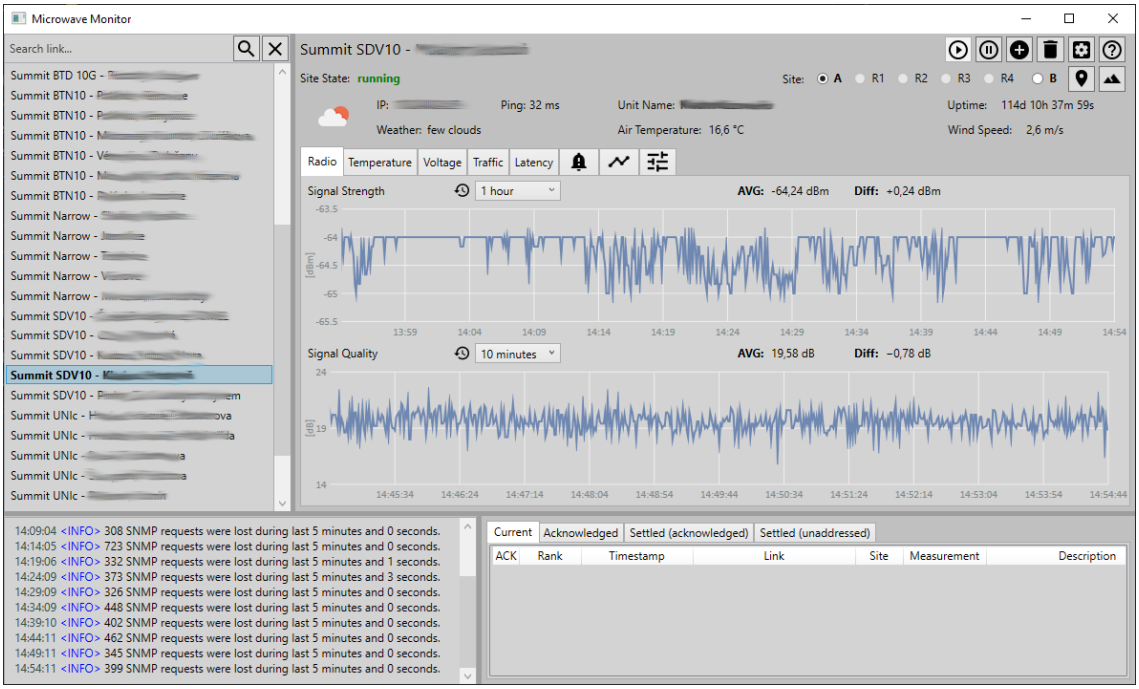
Obr. 6.1: Rozložení panelů uživatelského rozhraní.

- **1** – levý panel, obsahující seznam všech linek s možností jejich přepínání a vyhledávání,
- **2** – záhlaví zobrazení mikrovlnné linky, obsahující základní informace jako název spoje, IP adresa, počet skoků spoje a přepínáním mezi nimi, a další. Dále se v záhlaví zobrazuje i aktuální stav počasí v lokalitě. V pravé části záhlaví jsou umístěna tlačítka pro ovládání aplikace (přidání nového spoje, odstranění, nastavení aplikace, atd.).
- **3** – monitorovací sektor se zobrazenými grafickými průběhy získávaných parametrů, členěný do záložek. Jednou z nich je také nastavení parametrů linky. Nachází se zde také záložka pro zobrazení všech historických alarmů vygenerovaných pro dané zařízení.
- **4** – notifikační oblast, v levé časti se nachází okno s posledními hlášeními, v pravé části pak panely se seznamy alarmů.

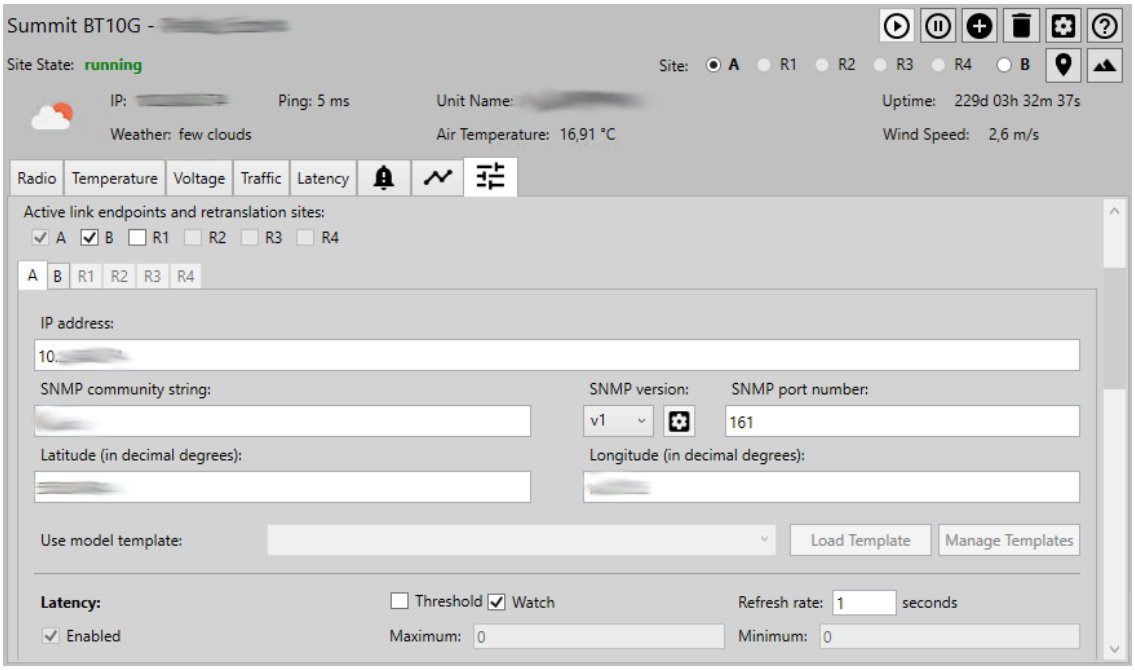
Toto rozvrhnutí aplikace by mělo být uživatelsky pohodlné a přehledné i pro větší množství monitorovaných spojů. Snímek výsledného hlavního okna aplikace je na obrázku 6.2.

6.1 Monitorovací sektor

Monitorovací sektor je ústřední částí celého hlavního okna. Ploše dominuji grafické průběhy vyčítaných veličin, vykreslované pomocí knihovny LiveCharts. Nad každým průběhem je zobrazen název veličiny, výběr pro přepnutí časového okna, průměrná hodnota veličiny za zobrazené období, a odchylka poslední vyčtené hodnoty od tohoto průměru.



Obr. 6.2: Snímek hlavního okna aplikace. Citlivé údaje jsou skryty.

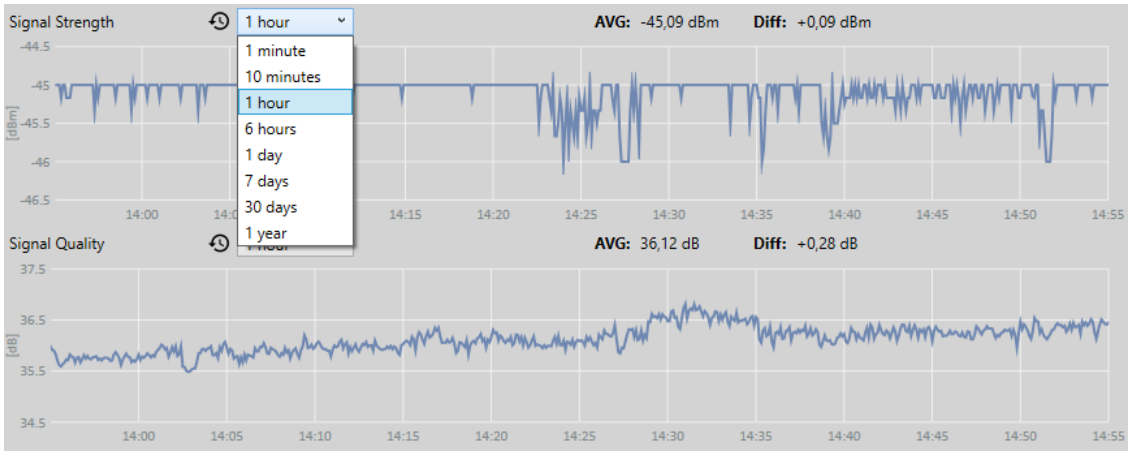


Obr. 6.3: Snímek nastavení linky. Citlivé údaje jsou skryty.

Každý zaznamenaný průběh veličiny lze přepínat mezi 8 časovými okny:

- 1 minuta, 10 minut, 1 hodina, 6 hodin,
- 1 den, 7 dní, 30 dní, 1 rok.

Přepínání mezi rozsahy je rychlé a plynulé, a to i pro nejdelší časové rozsahy (měsíc, rok). Daná období jsou načítána z odpovídajících retenčních politik InfluxDB. Ukázka výběru je na obrázku 6.4.



Obr. 6.4: Snímek přepínání časových oken průběhů veličin.

6.2 Nastavení linky

Snímek panelu s nastavením linky lze vidět na obrázku 6.3. Na obrázku je vidět záhlaví panelu s nastaveními jednotlivých zařízení. Počet namapovaných zařízení na jedné lince lze měnit zaškrtávacími políčky, mezi samotnými nastaveními zařízení se pak lze přepínat pomocí panelů pod nimi. Na obrázku naopak není vidět nejvýše položená část s poli pro název linky a vyplnění poznámky, a také dolní část s nastavením OID a parametrů vyčítání veličin.

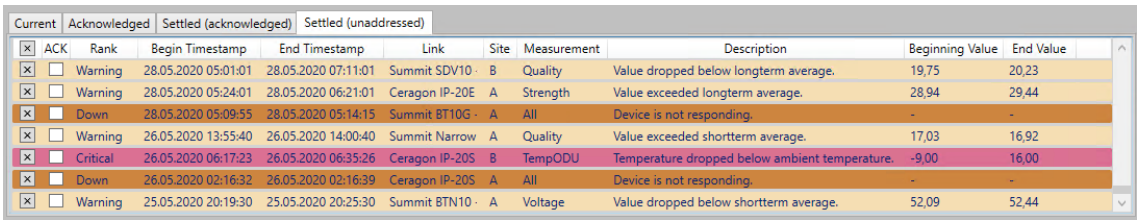
6.3 Seznamy alarmů

Na obrázku 6.5 je zobrazen snímek pravé části notifikační oblasti, se zobrazenými panely alarmů. Na snímku lze vidět několik skutečných alarmů, které byly vytvořeny. Nachází se zde čtyři záložky:

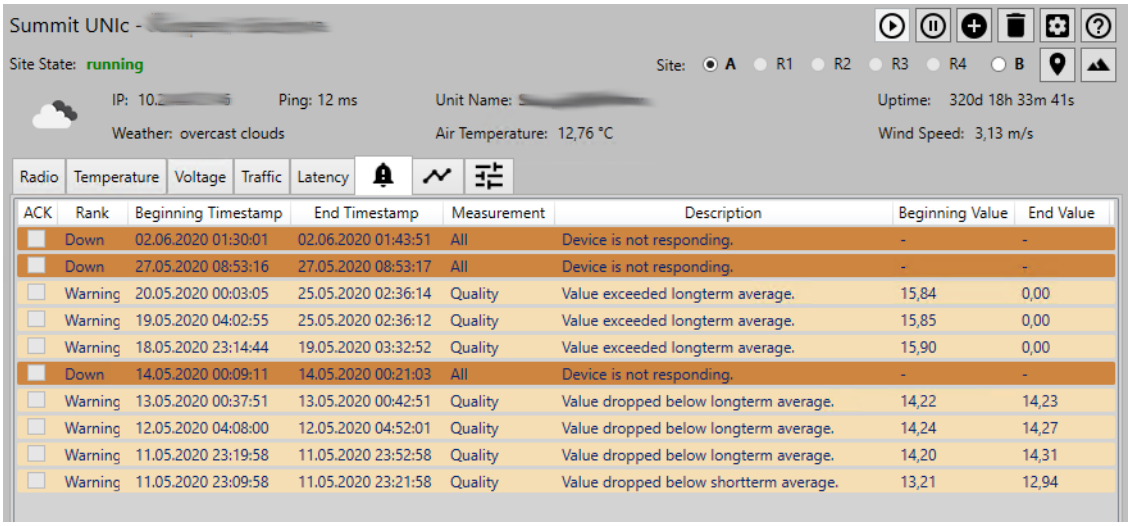
- `Current` – aktuální nepotvrzené alarmy,
- `Acknowledged` – aktuální potvrzené alarmy,
- `Settled (acknowledged)` – ukončené potvrzené (deaktivované) alarmy.
- `Settled (unaddressed)` – ukončené nepotvrzené (deaktivované) alarmy.

Alarmy lze do záložek `Acknowledged` přesouvat zakliknutím volby `ACK`. Platí to i pro zpětný přesun. Jakmile však jednou byl některý alarm přesunut na záložku `Settled`, znamená to, že jeho hodnota se již vrátila do povolených mezí, a alarm již není aktivní. Přesun do záložky `Settled` tedy řídí aplikace a nelze jej ovlivnit.

Na záložkách `Settled` se též nachází tlačítko `Hide` pro skrytí alarmu z výpisu. Alarm však bude stále zpětně dohledatelný, a to na záložce `Device Alarms` daného zařízení v monitorovacím sektoru. Tento výpis, zobrazený na obrázku 6.6, obsahuje seznam až 500 posledních vytvořených alarmů pro dané zařízení.



Obr. 6.5: Snímek panelů alarmů. Citlivé údaje jsou skryty.



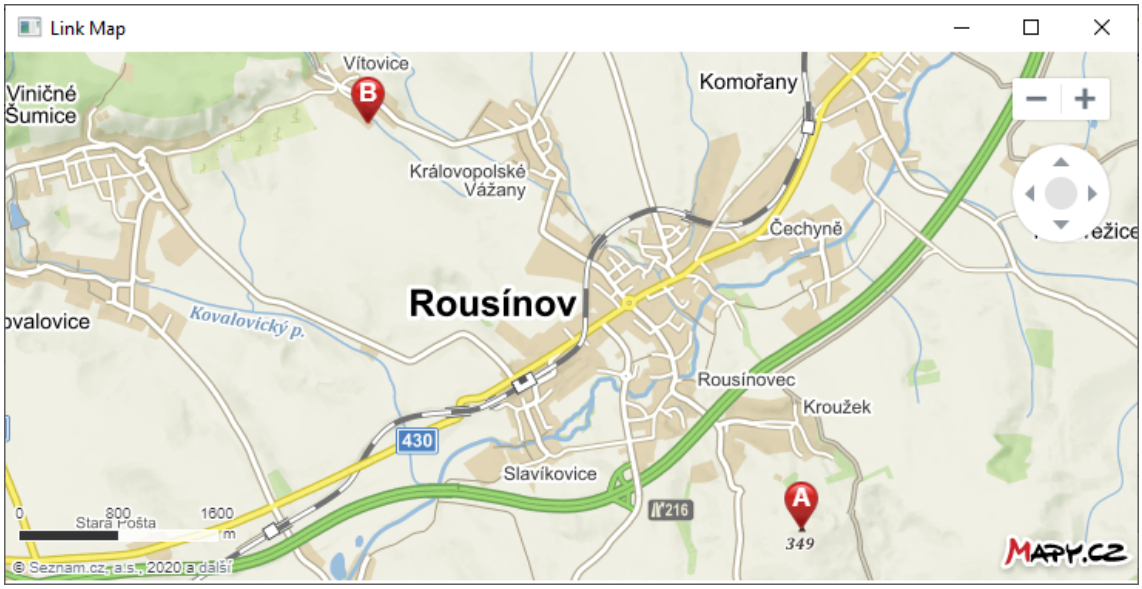
Obr. 6.6: Snímek panelu alarmů zařízení. Citlivé údaje jsou skryty.

6.4 Notifikační okno

Notifikační okno se nachází v levé části notifikační oblasti. V tomto okně jsou vypisovány textové statistické informace, varování, chyby, události vytvoření a deaktivování alarmů, a také výjimky v chronologickém pořadí.

6.5 Mapa jednotek

Aplikace disponuje funkcí zobrazení umístění jednotek linky na mapě. Mapový poklad je stahován pomocí API Mapy.cz. Ukázka pro smyšlené lokality je na obrázku 6.7. Lokality jsou označovány píseny na základě skutečného označení uvnitř aplikace.



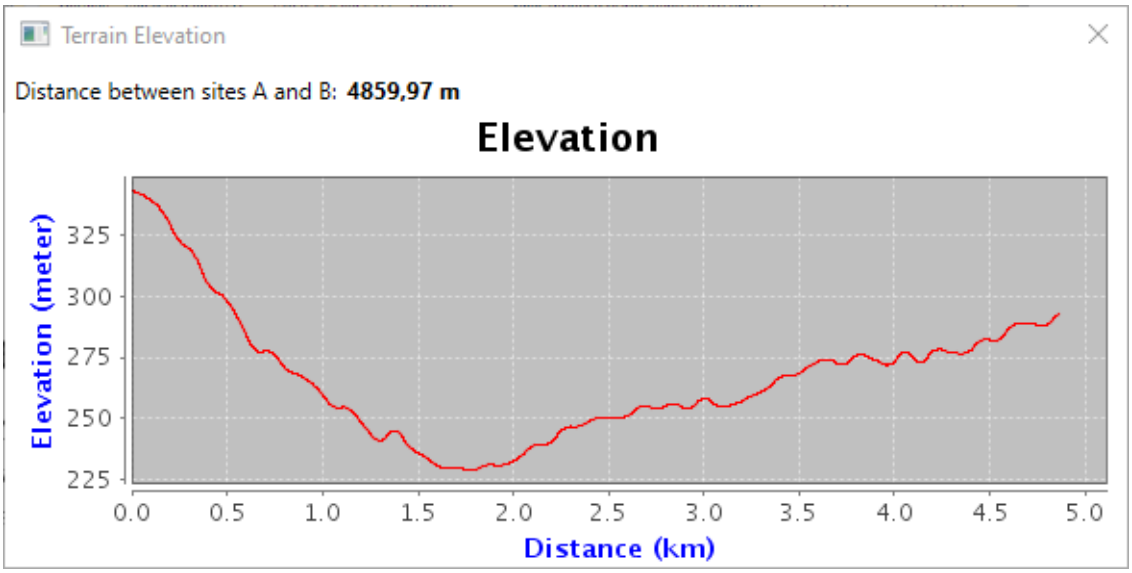
Obr. 6.7: Snímek okna s mapou linky.

6.6 Vykreslení terénního profilu linky

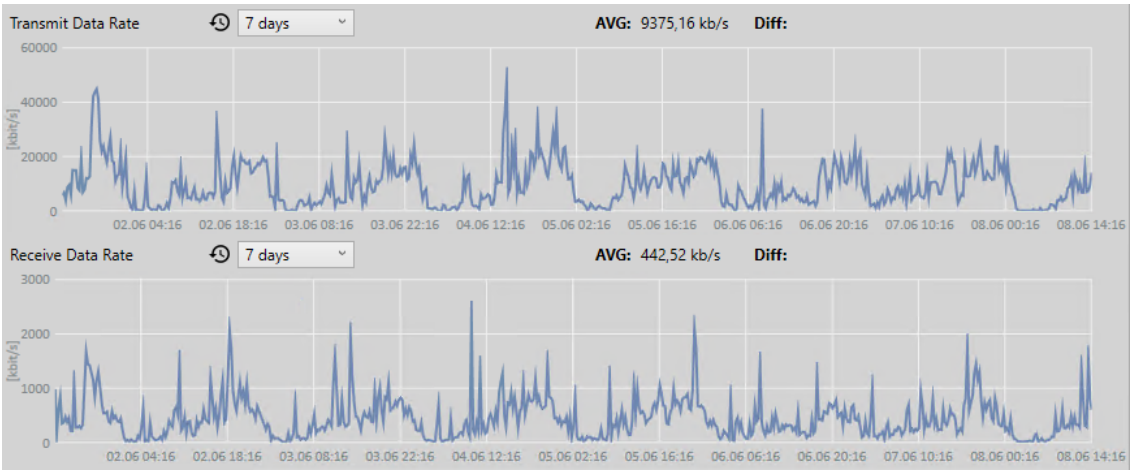
Funkce terénního profilu linky vykresluje terénní profil mezi zařízeními za pomoci MapQuest API. Aplikace musí dopočítávat všechny vykreslované body na spojnici mezi dvěma koncovými zařízeními linky. Počet těchto bodů ovlivňuje výsledné rozlišení terénního průběhu. Počet vzorků je nastaven na 60. Ukázka je na obrázku 6.8, jedná se o stejné lokality jako na předchozím obrázku 6.7 s mapou linky.

6.7 Snímky okna s nastavením aplikace

V příloze A jsou zobrazeny snímky okna nastavení aplikace, obsahující tři záložky pro analyzátory dat, blíže popsané v následující kapitole 7.



Obr. 6.8: Snímek okna s terénním profilem linky.



Obr. 6.9: Ukázka zachycených datových přenosů na lince.

7 Analýza dat a vytváření alarmů

Metody analýzy a principy uvedené v této kapitole slouží k odhalení potenciálně negativních jevů ve vyčítaných průbězích veličin, případně k notifikaci mimořádných stavů. Jsou do aplikace integrovány buďto formou samostatných tříd analyzátorů (sledování odklonu od dlouhodobého průměru; analýza periodičnosti průběhů; analýza teplotní korelace) provázaných s objektem `AlarmManager` či přímo formou třídních metod tohoto objektu (statické notifikace).

Třídy analyzátorů mohou dědit z abstraktní třídy `Analyser`, které je navrženo pro výpočty analyzátorů probíhající v samostatných vláknech nezávisle na změně (zisku) nových dat. Výjimkou je analýza teplotní korelace, kde jsou akce výpočtů vázané na změnu dat (získání nových dat z kolektorů). V tom případě jsou výpočty prováděny na vlákně, které tuto změnu vyvolalo, tj. vlákno kolektoru počasí, resp. jedno z vláken kolektorů teplotních veličin.

V případě pozitivní detekce anomálie analyzátory volají metodu `GenerateAlarm` objektu `AlarmManager`, pakliže anomálie ustála a došlo k návratu do stavu mimo kritické hodnoty, je volána metoda `SettleAlarm`. `AlarmManager` naopak volá metodu `DeviceStopped` tříd analyzátorů v případě, že došlo k pozastavení monitoringu či odstranění zařízení ze strany uživatele, a metodu `LoadSettings` pro hromadné načtení uživatelsky nakonfigurovaných hodnot vlastností tříd analyzátorů. Metoda sledování odklonu od dlouhodobého průměru dále ještě provádí individuální komunikaci s objektem `AlarmManager` z důvodu získání informací o historických výpadcích. Abstraktní třída `Analyser` obsahuje kolekce `ids` a statické kolekce `watch`, sdílené mezi všemi analyzátory. Pro každou vyčítanou veličinu je vytvořena unikátní kolekce. První jmenované uchovávají vazbu ID vygenerovaných alarmů na ID zařízení, zatímco druhé obsahují ID všech zařízení, u kterých je daná veličina monitorována, na základě čehož analyzátory zahrnují zařízení do sledování.

7.1 Sledování odklonu od dlouhodobého průměru

Tato metoda je základním způsobem detekce anomálií, integrovaným v aplikaci. Metoda v pravidelných intervalech provádí výpočet aritmetického průměru posledních vyčtených hodnot monitorovaných veličin, a to za dlouhý a za krátkých časový úsek, načež provede jejich porovnání. Pakliže je překročen limit rozdílu, daný uživatelsky nastavitelnou procentuální částí průměru za dlouhý časový úsek, je vytvořen alarm.

7.1.1 Princip

Metoda vychází z předpokladu, že monitorované veličiny, jako jsou např. síla signálu, odstup signálu od šumu, či napětí na jednotce, by v ideálním případě měli mít statický charakter. Ke změnám by mělo docházet pouze v důsledku přirozených venkovních vlivů, nejčastěji vlivu povětrnostních podmínek.

V reálném nasazení je samozřejmě nemožné dosáhnout neměnné hodnoty monitorovaných veličin, už jen z důvodu přirozeného elektromagnetického šumu a užití reálných nedokonalých součástek v mikrovlnných jednotkách. Proto je počítáno s určitým tolerančním polem, k jehož překročení musí nejprve dojít, aby případný vytvořený alarm měl pro uživatele vyšší informační hodnotu. Toleranční pole lze určit manuálním nastavením statických prahů – tzv. thresholdů[1], či dynamickým výpočtem na základě vyčtených historických hodnot, čemuž se věnuje tato metoda.

Metodu nelze použít pro všechny vyčítané veličiny, zejména pro ty, které jsou soustavně ovlivňovány povětrnostními podmínkami. Jedinými aplikací monitorovanými veličinami, spadajícími do této skupiny, jsou teploty venkovní (ODU) ale i vnitřní jednotky (IDU, viz dále). Monitorování výchylek těchto teplotních hodnot zajišťuje metoda analýzy teplotní korelace, o níž je dále pojednáváno v kapitole 7.3.

Pro metodu lze užít dva různé přístupy:

- výpočet dlouhodobého průměru, jeho porovnávání s každou novou hodnotou;
- výpočet dlouhodobého a krátkodobého průměru, porovnávání mezi nimi.

Druhý jmenovaný přístup snižuje vliv krátkodobých lokálních zakolísání, čímž snižuje zátěž uživatele planými alarmy. Naproti tomu volba příliš širokého časového okna krátkodobého průměru může zapříčinit zanedbání výchylek, které již mohou mít význam pro ohrožení korektního chodu jednotky. V této aplikaci bylo implementováno porovnání dlouhodobého a krátkodobého průměru.

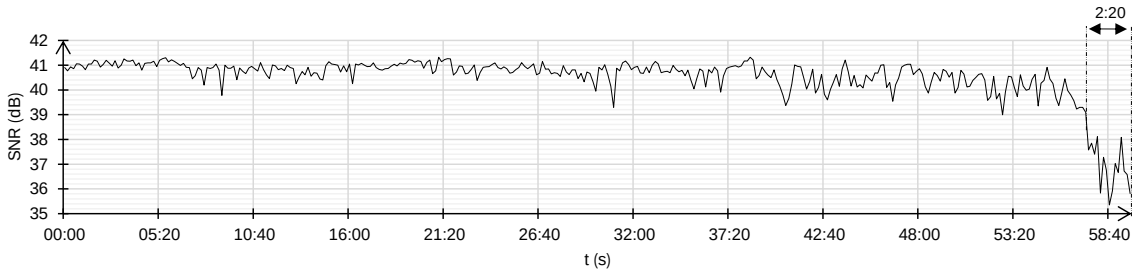
[1] Implementaci této funkce je věnována kapitola 7.4.2.

Délky časových oken a velikosti procentuálních prahů

Primárními parametry metody, které určují její citlivost, jsou časového okna krátkodobého průměru procentuální část hodnoty dlouhodobého průměru, při niž překročení má dojít k vytvoření alarmu. Sekundárním parametrem je délka časového okna dlouhodobého průměru.

Časová okna dlouhodobého průměru a krátkodobého průměru se vzájemně nepřekrývají, nýbrž sdílí společnou časovou hranici. U dlouhodobého průměru je to horní hranice s vyšší časovou hodnotou, u krátkodobého dolní hranice s nižší časovou hodnotou.

Na obrázku 7.1 je zobrazena ukázka vyčteného průběhu hodnoty odstupu signálu od šumu (SNR) na jednom z reálných testovaných zařízení. Jedná se o časový interval jedné hodiny. SNR postupně mírně nabývá na rozkmitu, přičemž přibližně v posledních dvou minutách (v čase 57:40) nastává strmý pokles.



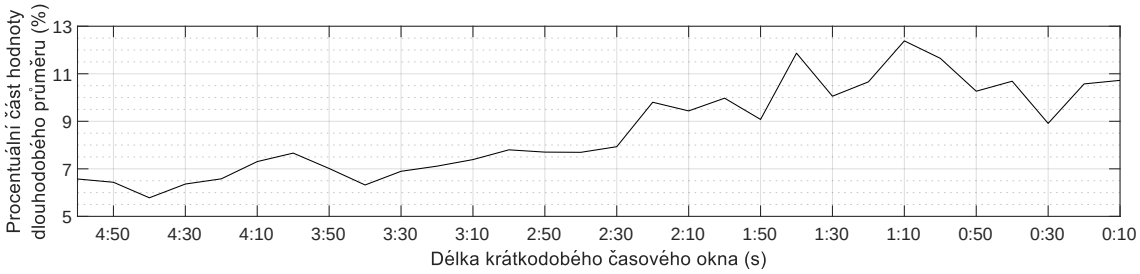
Obr. 7.1: Ukázka zachyceného hodinového průběhu SNR se strmým poklesem v posledních dvou minutách.

Byl proveden výpočet dlouhodobého a krátkodobého průměru pro časová okna s různou délkou:

- Spodní hranice dlouhodobého časového okna odpovídá zobrazenému intervalu, tedy $-1\,\mathrm{h}$.
- Horní hranice, společně s dolní hranicí krátkodobého časového okna, byla postupně zvyšována, a to od času $-5\,\mathrm{min}$ do $-0\,\mathrm{s}$, s krokem $10\,\mathrm{s}$, čímž byla zároveň zmenšována velikost krátkodobého časového okna.
- Pro takto vypočtené průměry byla následně vypočítána procentuální část, kterou tvoří hodnota krátkodobého průměru z hodnoty dlouhodobého průměru.

Vypočítané procentuální hodnoty částí byly vyneseny na graf 7.2, v závislosti na délce krátkodobého časového okna.

Není snadné určit, jaké konkrétní hodnoty procentuálního limitu a velikosti časových oken jsou ideálními nastaveními. Konkrétní konfiguraci je vhodné přizpůsobit podmínkám na daných monitorovaných zařízeních, tak, aby byl pro uživatele vyvážen poměr mezi efektivními varováními a planými alarmy.



Obr. 7.2: Ukázka závislosti procentuální části velikosti krátkodobého průměru ve dlouhodobém průměru, na délce časového okna krátkodobého průměru. Vychází z průběhu zobrazeného na obr. 7.1.

Ovlivnění veličin totálními výpadky protější jednotky

Během testování aplikace na skutečných zařízeních bylo zjištěno, že totální výpadky (doby, po které je sledované zařízení zcela nedostupné) jednoho ze zařízení na lince, mohou negativně ovlivnit i zbývající, resp. protější, jednotky.

Ukázalo se, že u některých typů jednotek, v případě výpadku protějšího zařízení, tedy ztráty mikrovlnného spojení, hodnoty veličin síly signálu a SNR začnou vykazovat neobvyklé hodnoty mimo normální rozsah veličin. To má za následek:

- planý alarm (uživateli je ohlášen totální výpadek, tudíž informovat o dalších přirozeně navázaných jevech není žádoucí),
- vysoké zkreslení hodnot dlouhodobých průměrů.

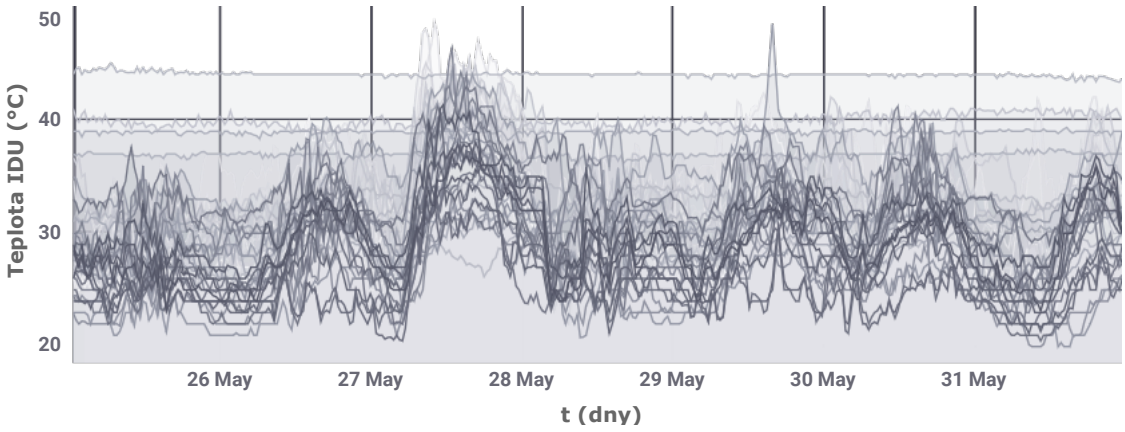
Tento jev je vhodné v aplikaci ošetřit. Toho lze docílit vynecháním všech časových intervalů, ve kterých došlo k totálnímu výpadku na jednom ze zařízeních linky, z výpočtu dlouhodobých a krátkodobých průměrů hodnot všech ovlivněných zařízení, namapovaných k dané lince.

Rozdílný charakter teplot IDU

Při testování aplikace bylo dále zjištěno, že původní domněnka, odhadující stálost teplot vnitřních jednotek (IDU), vzhledem k jejich předpokládanému umístění ve vnitřních prostorách, nebyla zcela správná. Některé modely totiž umožňují vyčítat teplotu IDU, která však může být fyzicky integrována s venkovní jednotkou do jednoho kompaktního celku (tzv. all-in-one zařízení), případně může být IDU sice externí jednotkou, její umístění však stále může být ve venkovních prostorách, např. na anténní konstrukci apod.

Při sledování této veličiny je tedy ještě třeba dodatečně určit, zda se jedná o:

- IDU se stálým charakterem teploty – vhodnou ke sledování pomocí této metody,
- IDU s vazbou na venkovní prostředí – vhodnou ke sledování pomocí metody teplotní korelace (viz kapitola 7.3).



Obr. 7.3: Ukázka zachycených teplot IDU za období jednoho týdne. Zobrazeno je 44 zařízení. Převažují průběhy vázané na venkovní teplotu prostředí.

7.1.2 Implementace

Procentuální limit, při kterém má být vytvořen alarm, je uživatelsky konfigurovatelný, stejně tak velikost časových oken. Pro vyšší přizpůsobitelnost, se v aplikace nachází **dva na sobě nezávislé analyzátory**, provádějící sledování odklonu od dlouhodobého průměru.

V terminologii aplikace se tyto analyzátory nazývají:

- `Long-term` (*dlouhodobý*),
- `Short-term` (*krátkodobý*).

Jak vyplývá z jejich označení, jeden je určený pro sledování delšího období, zatímco pro druhý se předpokládá nastavení období kratšího. Konfigurace je nicméně plně v roli uživatele. Oba analyzátory sdílí stejnou třídní strukturu, existují ve formě samostatných instancí objektu. Činnost obou z nich může být nezávisle zakázána/-povolena.

V tabulce 7.1 jsou uvedeny výchozí hodnoty uživatelských nastavení obou instancí analyzátorů.

Tab. 7.1: Výchozí konfigurační parametry analyzátoru sledování odklonu od dlouhodobého průměru.

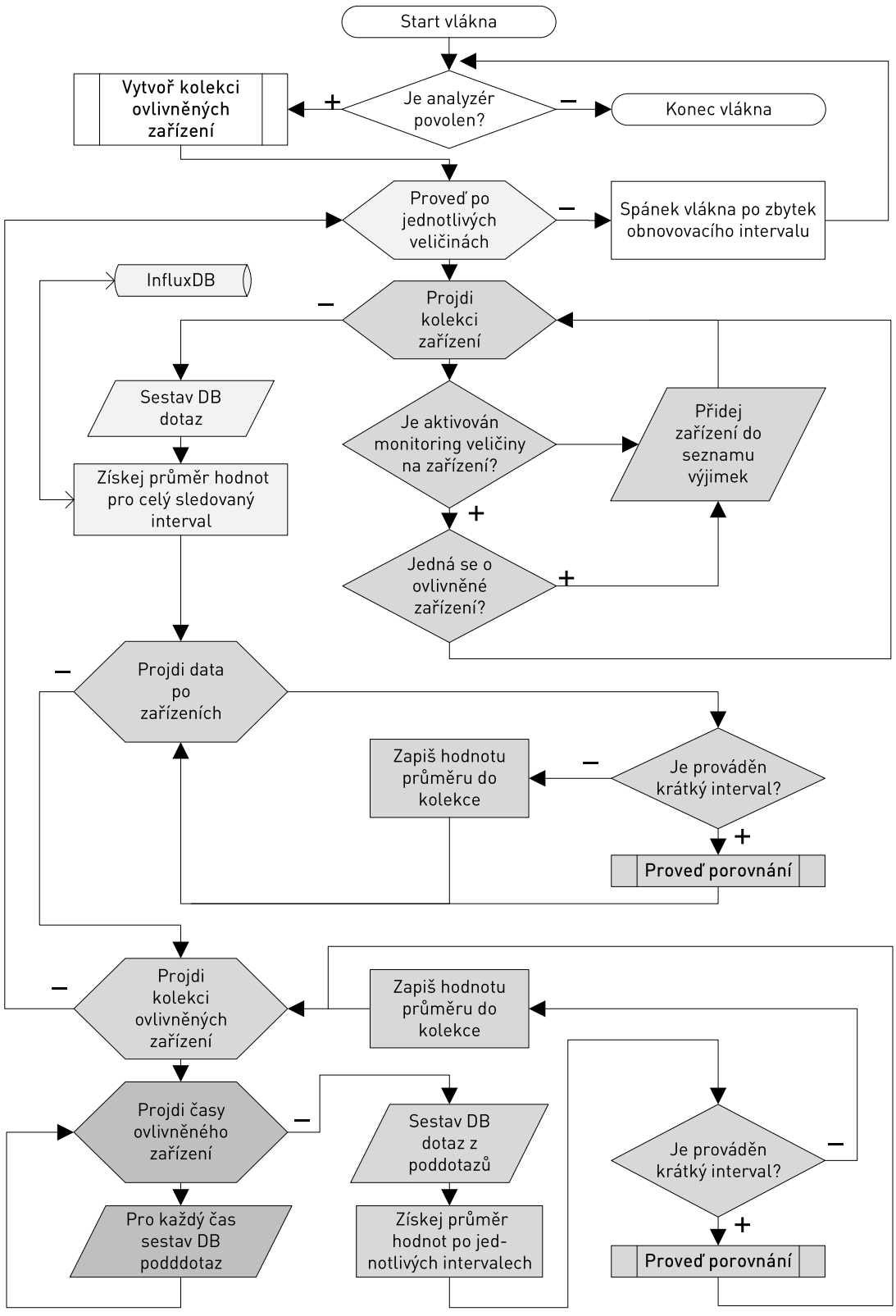
Položka konfigurace		Dlouhodobý analyzátor	Krátkodobý analyzátor
Časový interval obnovy dlouhodobého průměru		30 min	5 min
Časový interval obnovy krátkodobého průměru (a porovnání hodnot)		1 min	1 min
Dolní hranice dlouhodobého čas. okna		1 týden	1 h
Horní hranice dlouhodobého čas. okna a dolní hranice krátkodobého čas. okna		30 min	5 min
Minimální procento dlouhodobého průměru nutné pro vytvoření alarmu	síla signálu	10,0 %	15,0 %
	kvalita signálu	10,0 %	15,0 %
	teplota IDU	11,0 %	11,0 %
	napětí	3,1 %	3,1 %
	latence	250,0 %	300 %

Třída analyzátorů provádějících metodu sledování odklonu od dlouhodobého průměru nese název `AverageAnalyser`, dědí ze třídy `Analyser`, ve které je definováno vlákno `tQueryer`, na kterém se v pravidelných intervalech, definovaných v uživatelském nastavení, provádí výpočet dlouhodobých průměrů hodnot (prostřednictvím dotazování se InfluxDB). `AverageAnalyser` dále definuje ještě jedno vlastní vlákno, označené jako `tComparator`, na kterém běží výpočet krátkodobých průměrů hodnot, s následným porovnáním a vytvářením alarmů.

Vývojový diagram běhu vlákna analyzátoru je zobrazen na obrázku 7.4. Obě vlákna `tQueryer` i `tComparator` mají mají podobnou strukturu (metody), liší se v parametrech sestavovaného InfluxQL dotazu a v posledním kroku, kdy `tQueryer` ukládá získanou hodnotu dlouhodobého průměru do kolekce `data<veličina>`, zatímco `tComparator` provede porovnání získané hodnoty krátkodobého průměru s uloženou hodnotou dlouhodobého průměru v kolekci `data<veličina>`.

Po startu vlákna metodou `Start` ze strany objektu `AlarmManager` proběhne kontrola, zda je splněna podmínka pravdivosti vlastnosti `IsRunning`. Následuje metoda `GetDownAffectedDevices`, vracející kolekci časových intervalů `affectedDevices` indexovaných dle zařízení `Dictionary<int, TimePeriodCollection>`, pro taková zařízení, která byla ovlivněna výpadky protějších jednotek na lince.

Poté je sestaven seznam zařízení, která mají být vynechána z databázového dotazu, z důvodu nepovolení jejich monitorování. Následně je sestaven databázový dotaz. Vynechána jsou i ovlivněná zařízení, nacházející se v kolekci `affectedDevices`.



Obr. 7.4: Vývojový diagram běhu vlákna pro metodu sledování odklonu od dlouhodobého průměru.

Výpis 7.1: Struktura InfluxQL dotazu sledování odklonu od dlouhodobého průměru.

```
1 SELECT mean("název_hodnot")
2 FROM "název_DB"."retenční_doba"."veličina"
3 WHERE time > dolní_hranice_časového_okna
4 AND time < horní_hranice_časového_okna
5 AND podmínka_obsahující_vynechaná_zařízení
6 GROUP BY "device"
7 FILL(none)
```

Výpočet samotného aritmetického průměru provádí InfluxDB, na základě parametrů zaslaného dotazu. Ukázka dotazu je zobrazena ve výpisu 7.1. Funkce `mean` provede výpočet aritmetického průměru všech hodnot nashromážděných za časové období, definované klauzulí `WHERE` s parametrem `time`. Klauzule `GROUP BY "device"` pak zajistí rozdělení navrácených hodnot po jednotlivých zařízeních.

Získaná data z InfluxDB jsou v cyklu, po jednotlivých zařízeních, uložena do kolekce `data<veličina>`, nebo je v případě vlákna `tComparer` zavolána metoda `Compare` pro porovnání hodnot.

7.1.3 Sestavení dotazů pro ovlivněná zařízení

Následuje získání průměrů pro ovlivněná zařízení v kolekci `affectedDevices`. Kolekce obsahuje časové intervaly, pro které je možné provést výpočet průměru. Posloupnost kroků je stejná jako u nezasažených zařízení, odlišný je způsob sestavení databázového dotazu. `affectedDevices` je kolekce obsahující podkolekce s časovými intervaly, vázanými na ID zařízení. Tyto intervaly je pro příslušné zařízení nutno vyjmout z doby, dle které jsou vypočítávány hodnoty průměrů.

InfluxQL bohužel v klauzuli `WHERE` nepodporuje vyjmutí specifických časových intervalů prostřednictvím logických operátorů. Problém je několik let nahlášen v issue tracking systému vývojářů [43]. Situaci je však možno řešit sestavením poddotazu pro každé zařízení, a následným vytvořením hlavního dotazu.

Každý poddotaz reprezentuje jeden časový interval z kolekce `affectedDevices`, každý hlavní dotaz reprezentuje získání průměru pro jedno zařízení. Poddotazy mají podobnou strukturu jako ve výpisu 7.1. Za časové ohraničení v parametru `time` klauzule `WHERE` jsou dosazeny hranice časového intervalu. Klauzule `WHERE` poddotazu obsahuje také parametr `"device"=ID`, který zajišťuje navrácení hodnot pouze pro dané zařízení. Dosazovat naopak není nutné klauzuli `GROUP BY` s parametrem `device`.

Struktura hlavního dotazu je prostá:

```
SELECT mean(*) FROM <poddotazy> FILL(none)
```

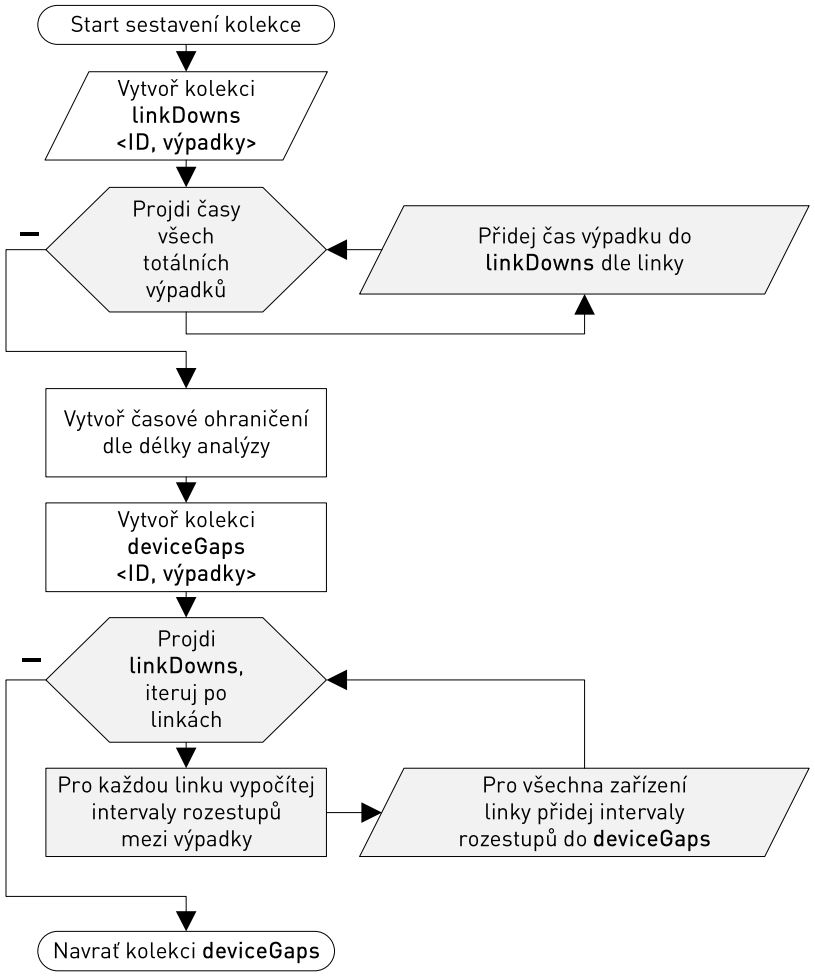
kde jednotlivé poddotazy jsou uvedeny v závorkách a odděleny čárkou.

Zbytek procedury pro ovlivněná zařízení probíhá shodně, získaná data z databáze jsou uložena do kolekce, nebo je zavolána metoda `Compare`.

Po provedení aktualizace průměrů a jejich porovnání u všech monitorovaných veličin, je vlákno po zbytek obnovovacího intervalu uspáno, a po probuzení cyklem while vráceno na počátek.

7.1.4 Sestavení kolekce ovlivněných zařízení

Následující část textu popisuje činnost metody `GetDownAffectedDevices`. Její vývojový diagram je zobrazen na obrázku 7.5.



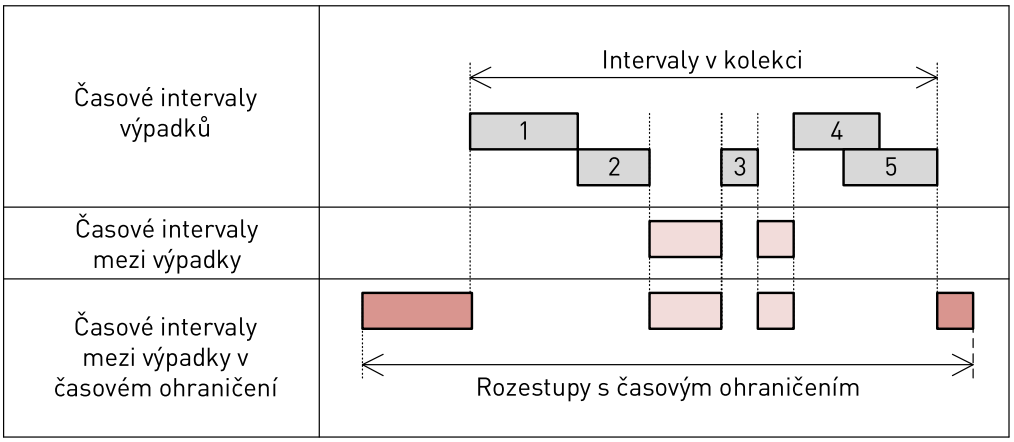
Obr. 7.5: Vývojový diagram metody `GetDownAffectedDevices`.

Metoda `GetDownAffectedDevices` z kolekce `DownTimes` objektu `AlarmManager` získává v cyklu délku trvání všech totálních výpadku a příslušná ID zařízení.[2]

Pro uložení intervalů a určení inverzních časových oken, je využita knihovna Time Period Library for .NET (TPL). Autorem knihovny je Jani Giannoudis, publikována je pod licencí CPOL [44].

Pro každou linku je vytvořena kolekce typu `TimePeriodCollection`, obsaženým v knihovně TPL, do níž jsou přidávány časy všech totálních výpadků všech zařízení přiřazených k dané lince. Následně je v cyklu pro každou linku s takto vytvořenou kolekcí, vypočítána inverzní kolekce `linkCorrectTimes` prostřednictvím objektu `TimeGapCalculator` z knihovny TPL, resp. jeho metody `GetGaps`.

Časové intervaly této inverzní kolekce představují doby, ve kterých nenastal výpadek na žádném zařízení linky. Parametrem metody `GetGaps` je také objekt typu `CalendarTimeRange`, umožňující provedení výpočtu inverzních intervalů na ohraničeném časovém úseku. Hranice tohoto časového úseku jsou nastaveny na časy, ohraničující aktuální zkoumané časové okno vlákna analyzátoru. Grafické vyjádření funkce metody `GetGaps` je zobrazeno na obrázku 7.6.



Obr. 7.6: Určení inverzních časových intervalů metodou `GetGaps` knihovny TPL [44].

Vytvořené inverzní intervaly jsou následně v cyklu přiřazeny ke všem zařízením dané linky a uloženy do kolekce `deviceCorrectTimes`, která je zároveň návratovou hodnotou celé metody `GetDownAffectedDevices`.

[2]Objekt `AlarmManager` do kolekce přidává výpadky s maximálním stářím, dle dolní hranice dlouhodobého časového okna dlouhodobého analyzátoru, tj. 1 týden ve výchozí konfiguraci.

7.1.5 Porovnání

Metoda `Compare` zajišťuje porovnávání dlouhodobého a krátkodobého průměru a vytváření/deaktivaci alarmů. Její vývojový diagram je zobrazen na obrázku 7.7.

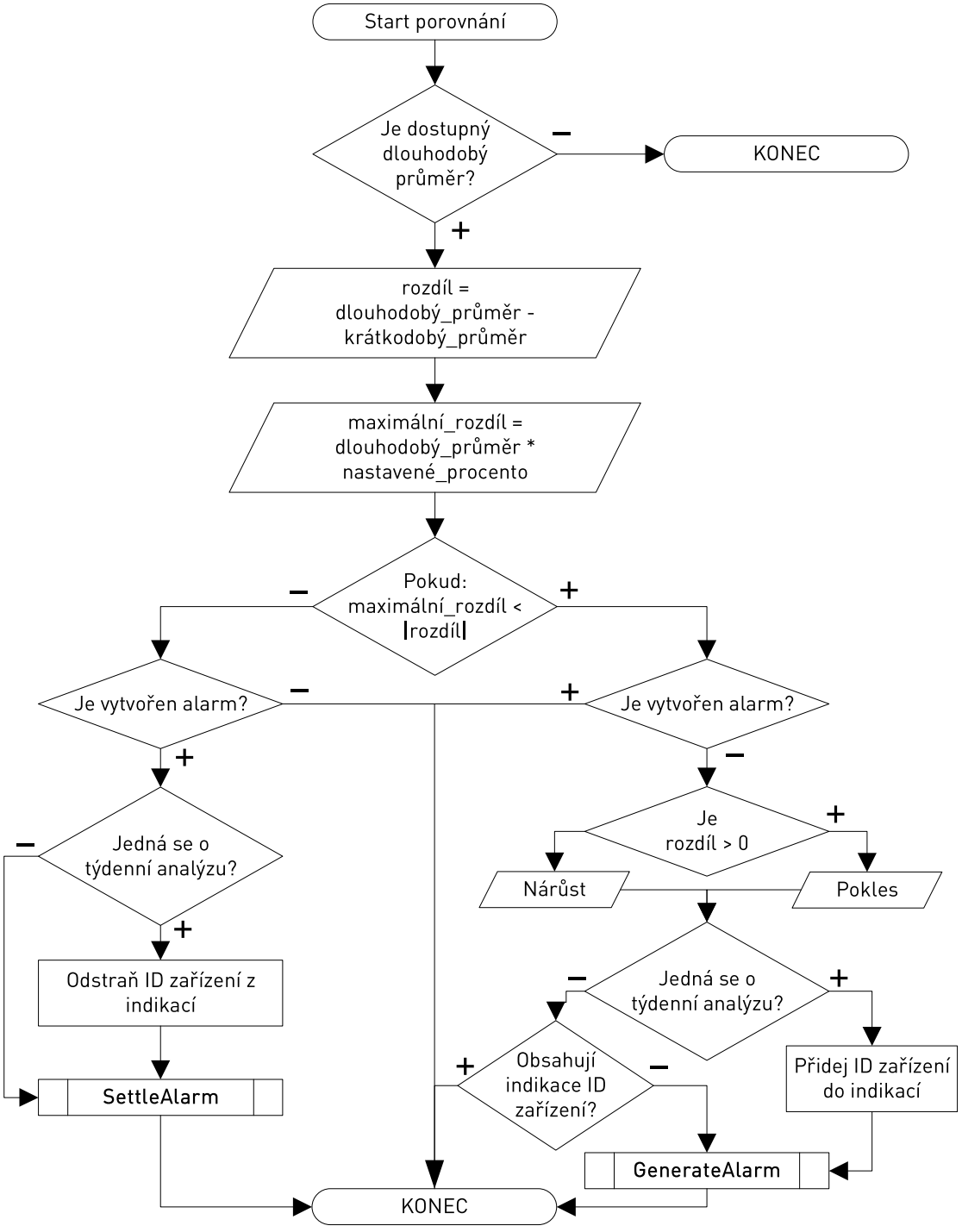
Nejprve je provedena kontrola dostupnosti dlouhodobého průměru, tj. zda je s čím porovnávat. Pokud ano, je vypočítán rozdíl `diff` dlouhodobého a krátkodobého průměru. Naproti tomu hodnotu maximálního rozdílu `maxDiff`, jíž překročení jest impulsem k vytvoření alarmu, činí uživatelsky nastavitelná procentuální část dlouhodobého průměru.

Pakliže je `diff` **větší než** `maxDiff`, je provedena kontrola, zda již není k tomuto zařízení vytvořen alarm z minulých analýz. Pokud ano, nepokračuje se dále.

Následné chování se liší v závislosti na tom, zda je metoda `Compare` volána v dlouhodobém analyzátoru, či v krátkodobém. Jedná-li se o dlouhodobý analyzátor, je nejprve uložena informace o vytvoření alarmu do kolekce `indication` a až poté je zavolána metoda `GenerateAlarm` objektu `AlarmManager`. Jedná-li se o analyzátor krátkodobý, je kolekce `indication` zkontrolována na existenci alarmu, pokud zde tato informace není, je zavolána metoda `GenerateAlarm`.

Toto chování je zaneseno z důvodu upřednostnění alarmů z dlouhodobého analyzátoru a zabránění vytváření alarmů s duplicitní informační hodnotou. Pokud je k dané veličině a danému zařízení již vytvořen alarm z dlouhodobého analyzátoru, v krátkodobém analyzátoru již další alarm vytvořen nebude.

Pakliže je `diff` **menší než** `maxDiff`, je postup analogický, v závěru je volána metoda `SettleAlarm` objektu `AlarmManager` pro deaktivaci alarmu.



Obr. 7.7: Vývojový diagram metody `Compare`.

7.2 Analýza periodičnosti průběhů

Tato metoda formou alarmu informuje uživatele o případném periodickém charakteru některého ze sledovaných průběhů veličin. Metoda je aktivní pro data síly signálu, kvality signálu a napájecího napětí. Vzrůsty a poklesy těchto veličin v pravidelných intervalech mohou značit například soustavné externí rušení či ovlivnění jinými vnějšími vlivy. Uživatel je v případě pozitivní detekce informován o nálezu detekovaného jevu společně s jeho periodou.

Mezi základní způsoby detekce periodičnosti signálu v časové doméně, za který lze časovou řadu považovat, patří *Fourierova transformace* (FT) a metoda *autokorelace*. FT převádí signál z časové domény do frekvenční domény, zatímco metoda autokorelace porovnává jednotlivé úseky signálu se sebou samým, tj. jinými úseky stejného signálu [47].

Fourierova transformace pracuje se spojitým signálem, v oblasti digitální techniky se však nejčastěji setkáváme se signálem diskrétním, charakterizovaným konečným počtem signálových vzorků s určitou vzorkovací frekvencí. Pro tyto signály je nutno použít diskrétní Fourierovu transformaci (DFT).

DFT je slabší při detekci periodičností na dlouhých časových úsecích, jelikož periodičnosti, jejichž frekvence nejsou celočíselným násobkem frekvenčního kroku výsledného DFT periodogramu, mohou být rozprostřeny do vícero frekvenčních složek periodogramu. Naproti tomu autokorelace je nespolehlivá při výskytu mnoha frekvenčních složek v signálu [47].

Pro prvotní implementaci do aplikace bylo zvoleno provádění DFT, s možností pozdější kombinace s dalšími metodami (např. autokorelace) pro větší zpřesnění výsledků.

7.2.1 Princip

Diskrétní Fourierova transformace převádí diskrétní signál z časové domény do frekvenční domény. DFT je v literatuře definována vztahem 7.1 [48]:

$$S[k] = \sum_{n=0}^{N-1} s[n] e^{-jk\frac{2\pi}{N}n}, \quad k = 0,\ 1,\ \cdots,\ N-1, \tag{7.1}$$

kde N označuje počet vzorků vstupního signálu a k je pořadí frekvenční složky na výstupu.

DFT má kvadratickou výpočetní náročnost $\mathcal{O}(n^2)$ (tj. počet kroků algoritmu bude úměrný druhé mocnině počtu vzorků [49], každý takovýto krok DFT sestává z náročné operace komplexního součinu), je proto obvykle vhodnější použít rychlou Fourierovu transformaci (*Fast Fourier Transform*, FFT), což je výpočetně optimalizovaný algoritmus pro výpočet DFT. FFT má linearitmickou časovou náročnost

$\mathcal{O}(n \log n)$, čímž je její čas výpočtu podstatně kratší[3]. Možnost provedení FFT je však podmíněna tím, že počet vzorků vstupního signálu musí být roven mocnině dvou, tj. $N = 2^m$, kde $m \in \mathbb{N}$ [48, 49, 50].

Algoritmus FFT je užíván v mnoha variantách, rozdělených do dvou skupin:

- první skupina, označována jako DIT (*Decimation in Time*, redukování v čase), rozděluje vstupní data na dvě posloupnosti, přičemž jedna má členy se sudými indexy, druhá pak s indexy lichými. Tyto dílčí posloupnosti jsou dále děleny stejným způsobem až do stavu rozdělení po dvoubodových základních posloupnostech. Na tyto dvoubodové posloupnosti je poté aplikována DFT.
- Druhá skupina, označována jako DIF (*Decimation in Frequency*, redukování na frekvenci), funguje obdobně jako DIT, vstupní data však nejsou rozdělována na základě indexů, nýbrž přímo na dvě poloviny. Dále se porkačuje stejně, dílčí posloupnosti jsou opět rozděleny na dvě poloviny atd., až do základních dvoubodových posloupností.

Výstupní data

Výstupem DFT/FFT je periodogram obsahující amplitudové spektrum (resp. modulové spektrum, pakliže je započítávána i imaginární pravá část spektra, případně jsou-li na vstupu komplexní hodnoty, viz dále) frekvenčních složek vstupního signálu. Počet výstupních frekvenčních složek je roven počtu vzorků signálu na vstupu.

Pro správné škálování amplitudových/modulových hodnot spektra je nutné tyto hodnoty na výstupu podělit počtem vzorků N vstupního signálu, resp. $N/2$ pro případ hodnot na vstupu pouze v oboru reálných čísel [51].

Spektrum je rozděleno na reálnou část v levé polovině spektra a imaginární část v pravé polovině spektra. Pakliže jsou hodnoty vstupního signálu v oboru reálných čísel (což je případ i této práce), imaginární část neobsahuje žádná užitečná data, jedná se pouze o imaginární obraz reálné části, tudíž s ní není nutno dále pracovat [50]. Poslední prakticky využitelná frekvenční složka je na pozici $k = N/2 - 1$, kde N je počet vzorků na vstupu.

Frekvenční složka na pozici $k = N/2$ je tzv. Nyquistova frekvence, která je rovna $f_{\text{vz}}/2$, kde f_{vz} je vzorkovací frekvence vstupního signálu. Ze vzorkovacího Nyquistova teorému dle vztahu 7.2:

$$f_{\text{vz}} > 2f_{\max} \tag{7.2}$$

vyplývá, že u této frekvenční složky již může dojít k překryvu frekvenčních spekter [48].

[3] Autor knihovny DSPLib pro signál o počtu vzorků $N = 8192$, s použitím procesoru Intel Core i7, uvádí pro DFT s použitím jeho knihovny výpočetní čas 360 milisekund, zatímco při aplikované FFT je to pouze 1 milisekunda [50].

Frekvenční složka na pozici $k = 0$ je takzvaná stejnosměrná složka signálu, která odpovídá aritmetickému průměru vstupních hodnot signálu.

Vzhledem k výše uvedenému, je hodnota frekvence první frekvenční složky na pozici $k = 1$ rovna f_{vz}/N, přičemž tato hodnota představuje i frekvenční krok mezi jednotlivými frekvenčními složkami, neboli rozlišení spektra. Každá frekvenční složka spektra nese informaci o energii celého přilehlého frekvenčního rozsahu s šířkou odpovídající rozlišení spektra, což je pro praktické užití v této aplikaci velice důležité, jelikož žádná frekvence potencionální periodičnosti ve sledovaném průběhu nikdy přesně neodpovídá hodnotám frekvencí frekvenčních složek spektra.

Ukázka

Na obrázku 7.8 lze vidět data, reprezentující odstup signál–šum (SNR) na jedné z reálných mikrovlnných jednotek, zachycená pomocí aplikace za časový interval jednoho týdne (viz část 8.2 věnující se testování a získaným datům). Pro názornost ukázky byla vybrána data se zřetelným periodickým charakterem, odpovídajícím periodě jednoho dne. Data byla pro tento účel předpřipravena:

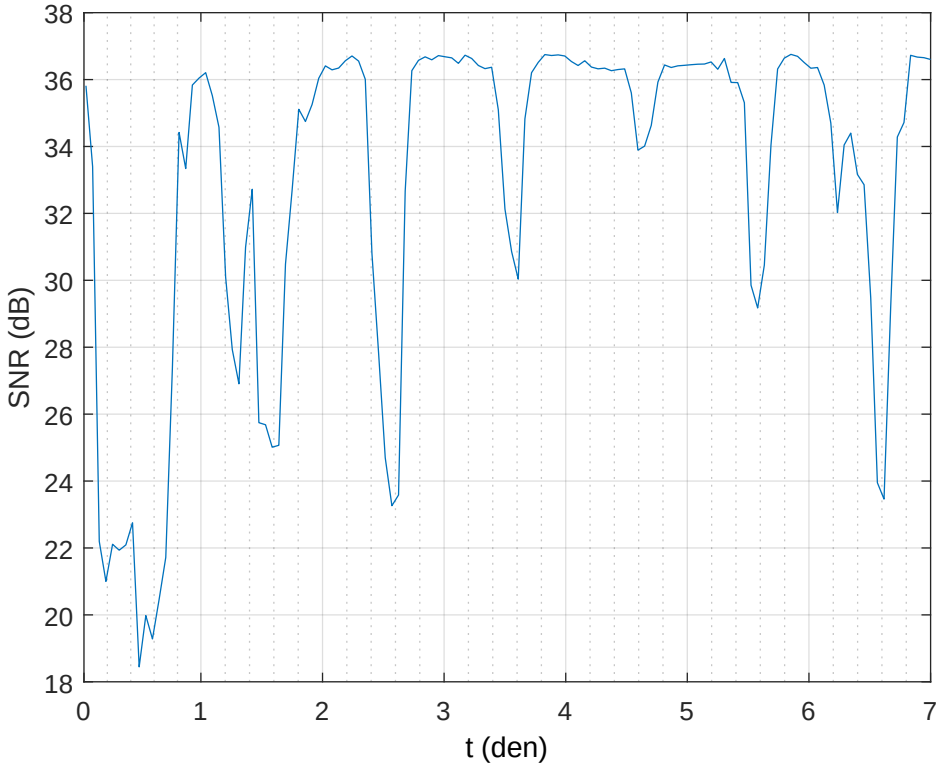
- jedná se o 128 vzorků (2^7, tak, aby byla splněna podmínka mocniny dvou),
- délka intervalu je přesně 1 týden, tj. 604 800 s,
- z výše uvedeného vyplývá délka jedné vzorkové periody 4 725 s, tj. 78,75 min, což udává vzorkovací frekvenci $211,640\,\mu\text{Hz}$.

Na obrázku 7.9 jsou zobrazena výstupní data po průchodu algoritmem FFT. Jedná se o periodogram zobrazující 1. až 64. frekvenční složku (vstupní data měla velikost 128 vzorků), imaginární pravá polovina je vynechána. Velikost amplitud je přizpůsobena jejich vydělením hodnotou $N/2$. Odpovídající frekvenční krok, neboli rozlišení osy `x`, je $1,654\,\mu\text{Hz}$, určený na základě vzorkovací frekvence. Tato hodnota je zároveň i hodnotou první frekvenční složky.

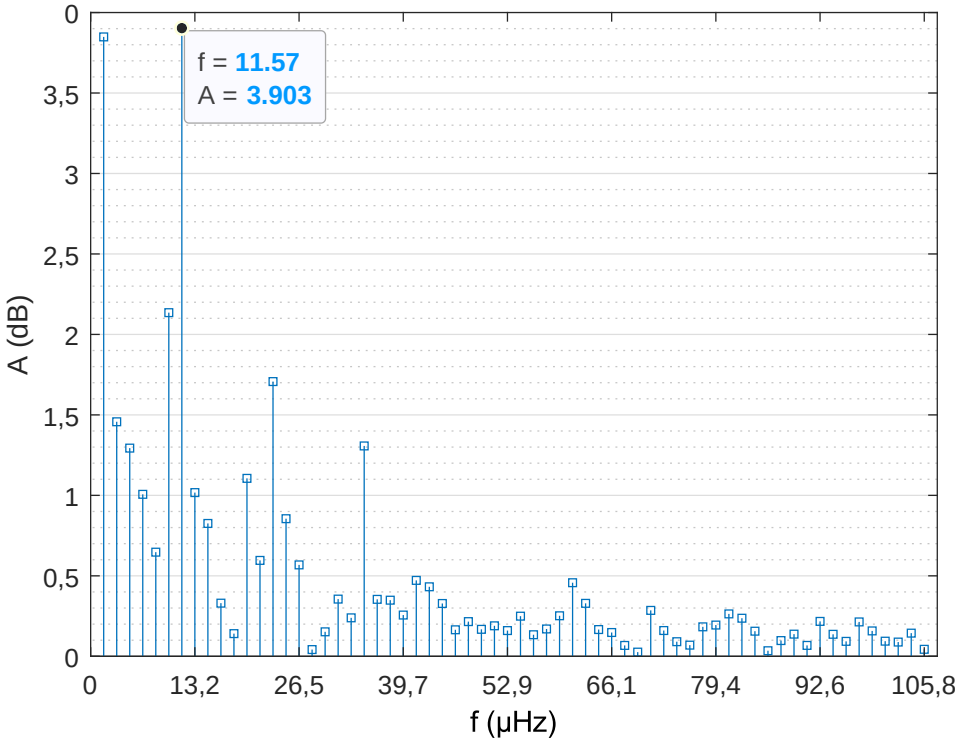
Nultá frekvenční složka (tzv. stejnosměrná složka) je v zobrazeném periodogramu vynechána, jelikož svou velikostí poměrově značně překonává všechny ostatní složky. Její hodnota amplitudy činí $32,9\,\text{dB}$, zatímco u druhé nejvyšší složky je to hodnota $3,9\,\text{dB}$.[4]

Z výstupu je patrné, že frekvenčními složkami s nejvyššími hodnotami jsou složka č. 7, odpovídající frekvenci $11,574062\,\mu\text{Hz}$, s amplitudou $3,903\,\text{dB}$, a složka č. 1, odpovídající frekvenci $1,653438\,\mu\text{Hz}$, s amplitudou $3,849\,\text{dB}$.

[4]Decibely jsou pro osu `y` uváděny z důvodu původního signálu měřeného v této jednotce, nikoli z důvodu logaritmické škály na ose `y` periodogramu.



Obr. 7.8: Ukázka vstupních dat pro FFT. Časový interval je jeden týden.



Obr. 7.9: Ukázka výstupního periodogramu FFT pro data z obr. 7.8.

Z frekvence lze určit délku periody, jež pro 11, 574062 µHz odpovídá 86 400 sekundám, tedy jednomu dni, což koresponduje s trendem ve vstupních datech. Délka periody druhé jmenované frekvence pak odpovídá celému týdennímu úseku, což je projev měnící se hloubky denních poklesů hodnot, kdy tyto poklesy zde mají znatelný harmonický charakter (maximální pokles nastává první den, poté lze poklesy aproximovat jako postupně klesající, přičemž 4. den je pokles hodnot na minimální hodnotě, následně však opět roste).

Aplikace například může vybrat frekvenční složku s nejvyšší hodnotou amplitudy, porovnat ji s nastaveným limitem, a při překročení tohoto limitu informovat uživatele.

7.2.2 Implementace

V implementaci analyzátoru byla pro výpočet FFT zvolena knihovna DSPLib s licencí MIT, jíž autor je Steve Hageman. Knihovna obsahuje metody pro výpočet DFT i FFT [50].

Třída analyzátoru `PeriodicityAnalyser` dědí ze třídy `Analyser`, vytváří tedy vlastní vlákno, na kterém v pravidelných intervalech spouští analýzu, v nečinný čas je vlákno uspáno.

Analyzátor provádí FFT pro tři vyčítané veličiny:

- sílu signálu,
- kvalitu signálu (SNR),
- napájecí napětí,

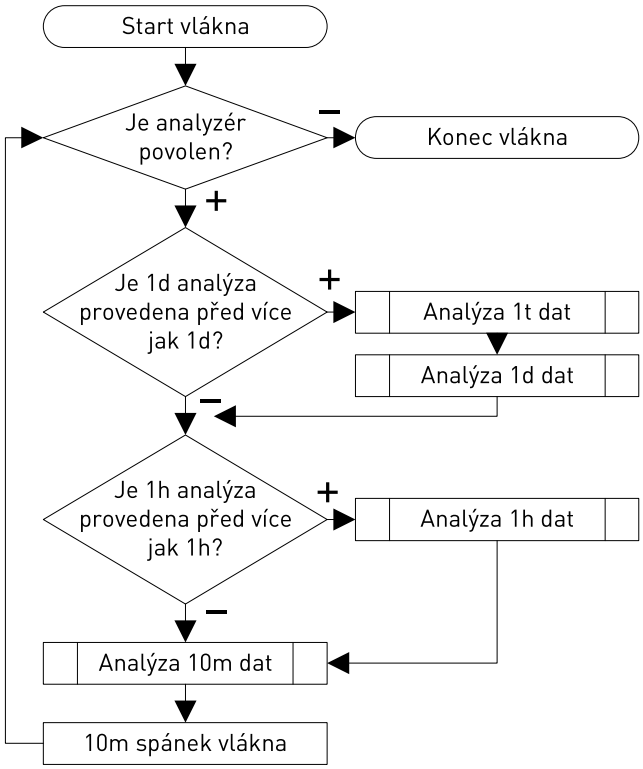
a to na čtyřech časových intervalech, každém o počtu vzorků $N = 128$:

- 10 minut, perioda vzorkovací frekvence $T = 4,6875\,s$,
- 1 hodina, perioda vzorkovací frekvence $T = 28,1250\,s$,
- 1 den, perioda vzorkovací frekvence $T = 675\,s$,
- 7 dní, perioda vzorkovací frekvence $T = 4\,725\,s$.

7.2.3 Plánování a analýza

Na obrázku 7.10 lze vidět vývojový diagram pro plánování jednotlivých analýz. Vlákno analyzátoru je vytvářeno prostřednictvím metody `Start`, setterem třídní vlastnosti `IsRunning`, ovládané objektem `AlarmManager`. Na vytvořeném vlákně je spuštěna metoda `Run`, obsahující while cyklus s jedinou vstupní podmínkou `IsRunning`.

Cyklus postupně synchronně volá metodu `Analyse` pro jednotlivé časové intervaly a vyčítané veličiny. Porovnává aktuální čas s posledním časem analýzy, přičemž analýza je znovu spuštěna až po uplynutí celého daného intervalu.



Obr. 7.10: Vývojový diagram vlákna analyzátoru periodičnosti.

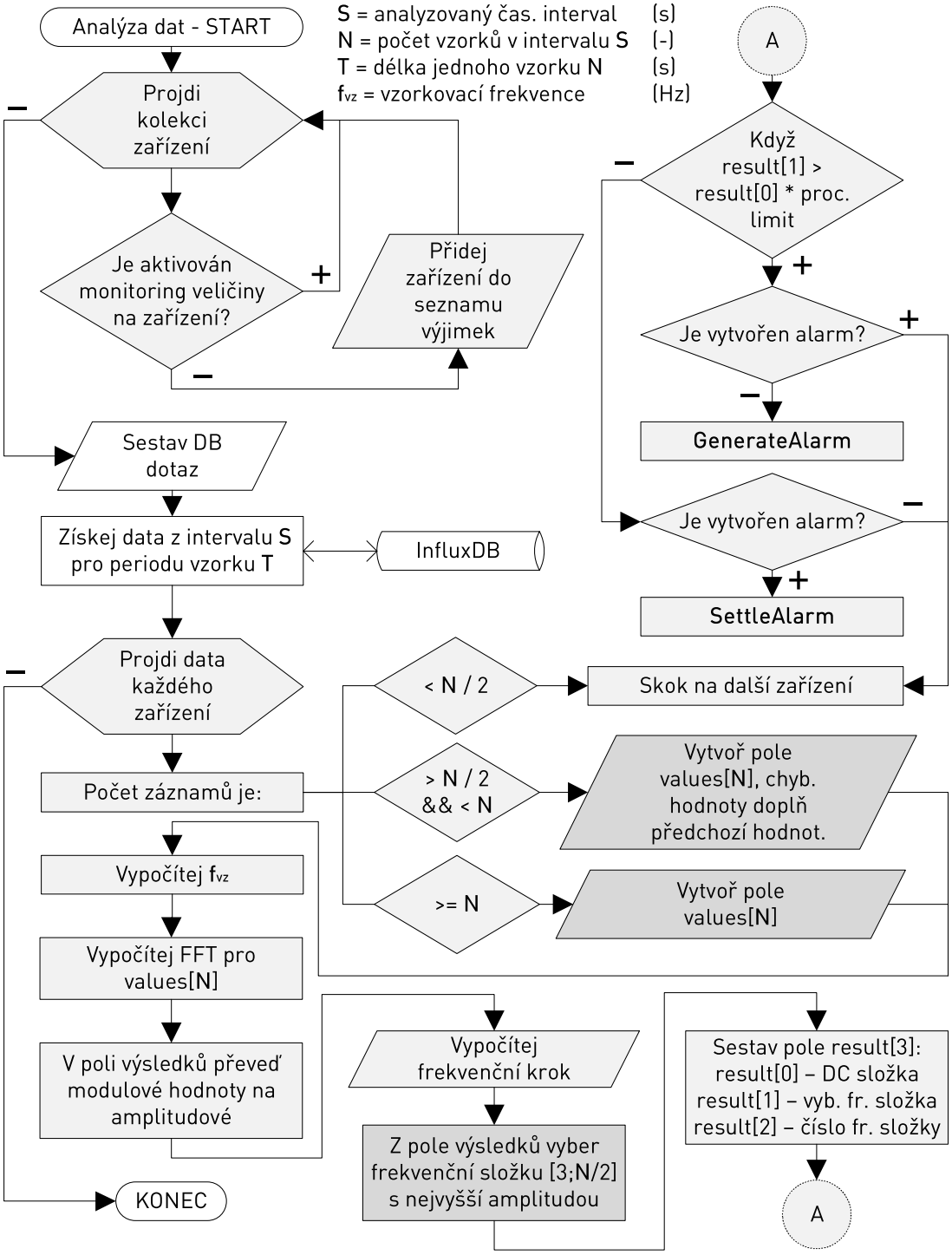
Analýza je prováděna nejprve od nejdelšího týdenního intervalu, po poslední desetiminutový interval. Je to z důvodu upřednostnění delší periody u případné nalezené periodičnosti před periodami kratšími, při informování uživatele prostřednictvím alarmu. Analyzátor nevytváří více jak jeden alarm tohoto typu pro jedno zařízení.

Analýza

Na obrázku 7.11 je zobrazen vývojový diagram samotné analýzy. Jedná se o asynchronní metodu, provádějící:

- sestavení databázového dotazu,
- získání databázových dat,
- kontrolu získaných dat a jejich přizpůsobení,
- volání metody `Calc` pro výpočet FFT,
- vyhodnocení výsledku a vytvoření/deaktivování alarmu.

Po zavolání metody je nejprve proveden cyklus, vytvářející řetězec s InfluxQL klauzulí, obsahující seznam zařízení, která mají být z databázového dotazu vynechána (tj. zařízení na kterých monitoring není povolen). Následně je sestaven řetězec s databázovým dotazem, jehož tvar je uveden ve výpisu 7.2.



Obr. 7.11: Vývojový diagram analýzy periodičnosti.

Výpis 7.2: Struktura InfluxQL dotazu analyzátoru periodičnosti.

```
1 SELECT mean("název_hodnot")
2 FROM "název_DB"."retenční_doba"."veličina"
3 WHERE time > velikost_časového_intervalu AND time < now()
4 AND podmínka_obsahující_vynechaná_zařízení
5 GROUP BY time(délka_vzorku), "device"
6 FILL(none)
```

Podstatná je klauzule `GROUP BY`, specifikující, že navrácená data mají být rozdělena po jednotlivých zařízeních, a také po časových úsecích uvedených ve výpisu 7.2 v parametru „délka vzorku“. Jeho hodnota koresponduje s periodou vzorkovací frekvence T.

Po získání dat následuje cyklus, zpracovávající data po jednotlivých zařízeních. Proběhne kontrola dat a jejich uložení do nového pole hodnot `values[N]`. V databázi nemusí být z různých důvodu k dispozici data pro každý časový vzorek. Analýza pro zařízení není proveden, pakliže počet získaných vzorků pro dané zařízení je menší než $N/2$, tedy 64. Pokud je vzorků více, ale stále méně než N, jsou chybějící vzorky doplněny předchozí hodnotou. V případě že je počet vzorků N (128) nebo více, je zpracováno prvních N vzorků.

Následuje určení vzorkovací frekvence a zavolání metody `Calc`. Argumenty metody jsou pole hodnot `values[N]` a vzorkovací frekvence, návratový typ je pole `results[3]`.

V metodě `Calc` proběhne inicializace třídy `FFT` z knihovny DSPLib. Poté je zavolána metoda `Execute` s argumentem `values[N]`, což provede výpočet FFT a navrátí pole komplexních frekvenčních složek. Tyto komplexní frekvenční složky jsou následně metodou `ToMagnitude` převedeny na jejich absolutní hodnoty, čímž jsou, vzhledem k reálným hodnotám v poli `values[N]`, získány amplitudy těchto složek. Metodou `FrequencySpan` je vypočítáno frekvenční spektrum. Následně je vybrána frekvenční složka s největší hodnotou (amplitudou).

Výběr začíná až od 3. frekvenční složky (včetně). Nultá, stejnosměrná složka, je vynechána vzhledem k charakteru analyzovaných dat, kdy bude prakticky vždy složkou nejvyšší (pracuje se s ní při vyhodnocení překročení limitu, viz dále). První a druhá frekvenční složka je vynechána z důvodu vysokého zkreslení a nepřesností, které se na těchto složkách při testování objevovalo. Frekvence odpovídající těmto složkám mohou být zachyceny ve vyšších časových intervalech. Z této skutečnosti, a délky nejdelšího časového intervalu (jeden týden), vyplývá, že nejdelší periodičnost, kterou je této konfiguraci možno zachytit, má délku 2,3 dne.

Návratem metody `Calc` je pole tři reálných hodnot:

- `results[0]` – nultá frekvenční složka,
- `results[1]` – fr. složka s nejvyšší amplitudou,
- `results[2]` – frekvence fr. složky s nejvyšší amplitudou.

Pro každou veličinu je v uživatelském nastavení definován procentuální práh, při kterém má dojít k vytvoření alarmu. Tato procentuální hodnota je vynásobena nultou frekvenční složkou, takto získaná hodnota je porovnána s hodnotou fr. složky s nejvyšší amplitudou. Pakliže hodnota nejvyšší amplitudy přesahuje nastavenou procentuální část nulté frekvenční složky, jedná se o pozitivní detekci vedoucí k vytvoření alarmu. V opačném případě, je detekce negativní.

Alarm je vytvořen pouze v případě, že žádný jiný alarm vzešlý z tohoto analyzátoru pro dané zařízení není aktivní.

7.2.4 Ukázka výpisu v módu ladění

Ve výpisu 7.3 je uvedena ukázka ladícího logu analyzátoru (režim ladění lze povolit v uživatelském nastavení) získaná ze skutečných dat. Význam položek:

- `FFT` – typ analyzátoru,
- `meas` – měřená veličina,
- `time` – časový interval (`1` = 10 min, `2` = 1 h, `3` = 1 den, `4` = 1 týden),
- `rows` – počet získaných vzorků z databáze,
- `dev` – ID zařízení,
- nultá frekvenční složka / nejvyšší amplituda / frekvence.

Výpis 7.3: Ladící výpis analyzátoru periodičnosti v aplikaci.

```
18:26:24 <DEBUG> Begin FFT meas: signalQ time: 2
query rows: 128
FFT dev: 100 meas: signalQ time: 2 42,509/0,012/0,00333 Hz
FFT dev: 101 meas: signalQ time: 2 43,346/0,012/0,00639 Hz
FFT dev: 102 meas: signalQ time: 2 39,917/0,028/0,00194 Hz
FFT dev: 103 meas: signalQ time: 2 40,853/0,025/0,00111 Hz
FFT dev: 104 meas: signalQ time: 2 20,682/0,045/0,00750 Hz
FFT dev: 106 meas: signalQ time: 2 18,644/0,022/0,01389 Hz
FFT dev: 107 meas: signalQ time: 2 18,611/0,028/0,00333 Hz
FFT dev: 108 meas: signalQ time: 2 17,808/0,018/0,00083 Hz
FFT dev: 109 meas: signalQ time: 2 18,684/0,028/0,00083 Hz
```

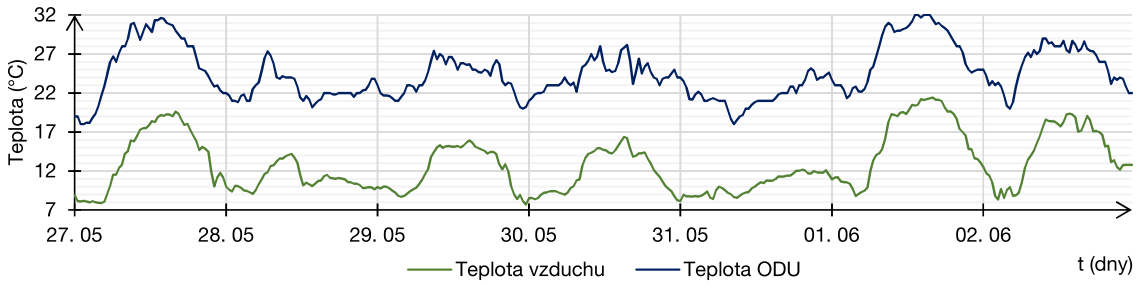
Všechny amplitudy uvedených zařízení ve výpisu 7.3 jsou pod hraničním limitem, nebyl vytvořen žádný alarm.

7.3 Analýza teplotní korelace

Tento analyzátor slouží ke stanovení dynamických limitů teplotních veličin, vázaných na projevy venkovního prostředí, a k jejich kontrole. Tyto teplotní limity jsou stanoveny na základě historických hodnot teplotní veličiny na konkrétním zařízení, s ohledem na aktuální stav počasí v dané lokalitě – pro každé zařízení je prováděn výpočet vlastního limitu. Analyzátorem je monitorována teplota venkovní jednotky (ODU), a teplota vnitřní jednotky (IDU) s nastavenou vazbou na venkovní prostředí (viz část 7.1.1, *Rozdílný charakter teplot IDU*).

7.3.1 Princip

Teploty zařízení mají přímou vazbu na teplotu prostředí, ve kterém se nacházejí. U venkovních jednotek se dále může projevovat i míra vystavení slunečnímu záření či síla větru. Bezpečné teplotní limity je tedy vhodné stanovit na základě těchto povětrnostních podmínek, pakliže jsou informace o aktuálním stavu v lokalitě k dispozici. Na obrázku 7.12 je ukázka zachycené korelace mezi teplotou vzduchu[5] a teplotou venkovní jednotky (ODU).



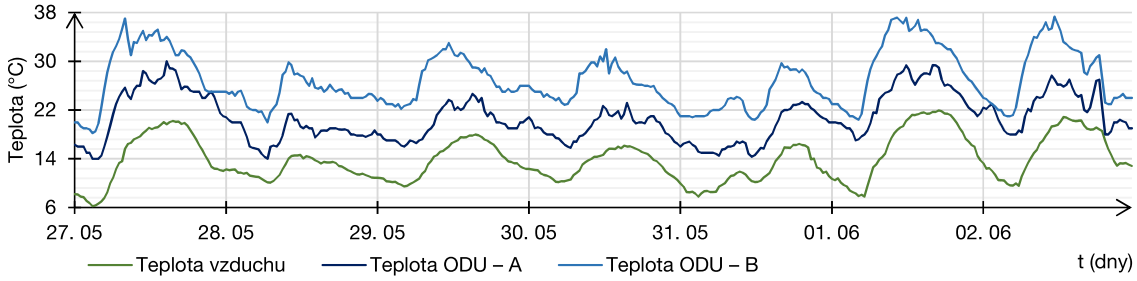
Obr. 7.12: Korelace mezi teplotou vzduchu a teplotou venkovní jednotky (ODU).

Z výše uvedeného vyplývá, že ke stanovení bezpečných teplotních rozsahů by mohlo být postačující specifikování určitého teplotního intervalu pro každý typ (model) mikrovlnné jednotky, který by byl přičítán k aktuální teplotě vzduchu. Tato funkce byla v aplikaci taktéž implementována, viz. 7.4.2.

Během testování se však ukázalo, že celá řada mikrovlnných jednotek, byť stejného modelového typu, vykazuje vůči sobě odlišné teplotní hladiny. Specifikování jediného teplotního intervalu pro daný typ jednotky se tedy ukázalo neefektivním, neboť práh maximální teploty by musel korespondovat s teplotním prahem jednotky vykazující nejvýše položenou teplotní charakteristiku, přičemž jiná zařízení stejného typu s níže položenou charakteristikou by již nebyla efektivně monitorována.

[5]Data počasí pochází ze služby OpenWeatherAPI.

Na obrázku 7.13 je ukázka teplotních charakteristik ODU dvou protějších mikrovlnných jednotek jedné linky. Zařízení jsou od sebe vzdálena 3 km a jsou umístěna přibližně ve stejné nadmořské výšce, vzhledem k jejich vzájemné vzdáleností je také okolní teplota vzduchu v obou případech téměř totožná. Typ jednotek je Summit BT10G. V některých okamžicích činí teplotní rozdíl mezi dvěma jednotkami až 10 °C.



Obr. 7.13: Rozdíly v teplotních charakteristikách dvou ODU. Model zařízení je Summit BT10G.

Tento jev může být způsobem vícero faktory, např. individuálními odlišnostmi kusů z výroby, degradací částí pro odvod tepla (teplovodivé pasty apod.), nebo v případě, kdy nejvýraznější rozdíly nastávají ve dne, i různou mírou slunečního záření dopadajícího na zařízení, kdy jedna jednotka může být vystavena slunci, zatímco druhá je částečně, nebo zcela ve stínu.

Z výše uvedeného vyplývá, že pro dosažení nejpřesnějšího fungování je výhodné teplotní limity určovat individuálně pro každé zařízení, na základě jeho historických hodnot.

7.3.2 Implementace

Tento analyzátor je implementován odlišně nežli předchozí dva analyzátory. Ty si vytvářely svá vlastní vlákna, a analýzu spouštěly v určitých časových intervalech, nehledě na změny ve zdrojových datech. Analyzátor teplotní korelace je však vázán na změnu, resp. zisk nově vyčtených dat v kolektorech, a žádná vlastní vlákna nevytváří. Vzhledem k tomu ani třída tohoto analyzátoru `TemperatureAnalyser` neimplementuje třídu `Analyser`, na rozdíl od ostatních analyzátorů.

Analyzátor využívá získaných informací o počasí z API OpenWeatherMap prostřednictvím kolektoru `WeatherColletor` (viz kapitola 5.4.3). Při každé aktualizaci stavu počasí tímto kolektorem je zároveň proveden přepočet teplotní odchylky pro daný čas a z toho odvozeného teplotního limitu.

Stejně tak je analyzátor navázán na aktualizace samotných teplotních veličin, konkrétně teploty ODU a teploty IDU s nastavenou vazbou na venkovní prostředí, jež vyčítání dat zajišťují kolektory `SnmpTempOdu` a `SnmpTempIdu`. Pro každou z těchto dvou teplot je vytvořena samostatná instance třídy `TemperatureAnalyser`, obě však sdílí stejná uživatelská nastavení. Při každé nově vyčtené hodnotě teploty je volána metoda `Compare` tohoto analyzátoru.

Navázání akcí analyzátoru na aktualizovaná data je zajištěno pomocí objektu `DeviceDisplay`, který slouží jako jednosměrný komunikační prostředek mezi kolektory a analyzátory či uživatelským prostředím aplikace. Pro každé zařízení, u něhož je aktivováno monitorování, je vytvořena jedna instance tohoto objektu, uchovávaná v kolekci, k níž mají přístup všechny třídy, vyžadující aktualizovaná data. Kolektory do těchto objektů ukládají vždy poslední hodnotu vyčtenou ze zařízení.

`DeviceDisplay` implementuje rozhraní `INotifyPropertyChanged`, což znamená že při změně jakékoli vlastnosti této třídy, je vytvořena událost `OnPropertyChanged`. K odběru těchto událostí je přihlášen mimo jiné i objekt `AlarmManager`. Ten zajišťuje při změnách:

- vlastnosti `WeatherId`, volání metody `WeatherChanged` analyzátoru;
- vlastností `DataTempOdu`, případně `DataTempIdu`, volání metody `Compare`.

Parametry obou metod jsou ID počasí, síla větru a zeměpisné souřadnice, pro metodu `Compare` jsou to ještě teplota vzduchu a vyčtená teplotě (předmět porovnání).

Metoda `WeatherChanged` je pouze prostředníkem zajišťujícím kontrolu, zda je aktivován monitoring teploty na daném zařízení. Hlavní metodou provádějící výpočet teplotní odchylky je metoda `WeatherUpdate` (viz dále).

Veškeré parametry analyzátoru jsou stejně jako u ostatních metod uživatelsky konfigurovatelné. Přehled první části nastavení a jejich výchozích hodnot je uveden v tabulce 7.2. Zbytek konfigurovatelného nastavení představuje substituční koeficienty počasí, které jsou uvedeny v tabulce 7.4.

Tab. 7.2: Výchozí konfigurační parametry analyzátoru teplotní korelace.

Maximální procentuální překročení teplotního limitu	30 %
°C na m/s rychlosti větru	0,3 °C
Maximální časová odchylka vyhledávaných hodnot v jednom dni	45 min
Maximální stáří vyhledávaných hodnot	25 dní
Počet nejnovějších dní, které mají být přeskočeny	1 dní
Počet denních hodnot tvořících průměrnou teplotní odchylku	7 dní
Minimální velikost teplotní odchylky	10 °C

7.3.3 ID počasí

ID počasí je trojmístný kód, který vrací API OpenWeatherMap (viz kapitola 5.4.3). Jedná se o klasifikaci počasí v lokalitě pomocí jediného identifikátoru.

- První číslice označuje skupinu počasí, kdy každá skupina reprezentuje jiné meteorologické jevy. Těchto skupin je sedm.
- Následující dvě číslice popisují intenzitu daného meteorologického jevu, počet těchto stavů je 9–11, v závislosti na skupině.
  - Výjimkou jsou skupiny 800 a 8xx.
  - Skupinu 800 reprezentuje jediný možný stav – jasno.
  - Skupina 8xx má čtyři stavy: skoro jasno, polojasno, oblačno a zataženo.
- Zvláštní je skupina 7xx, která obsahuje různé nezařaditelné atmosférické jevy.
  - Pro podnebí v ČR lze v této skupině reálně uvažovat pouze mlhu a opar.
  - Zbytek skupiny tvoří v našem podnebí velice nestandardní jevy (spad sopečného popela, tornádo, písečná bouře, atd.) [45].

Souhrnný přehled všech skupin je uveden v tabulce 7.3.

Tab. 7.3: Tabulka hodnot ID počasí OpenWeatherMap API [45].

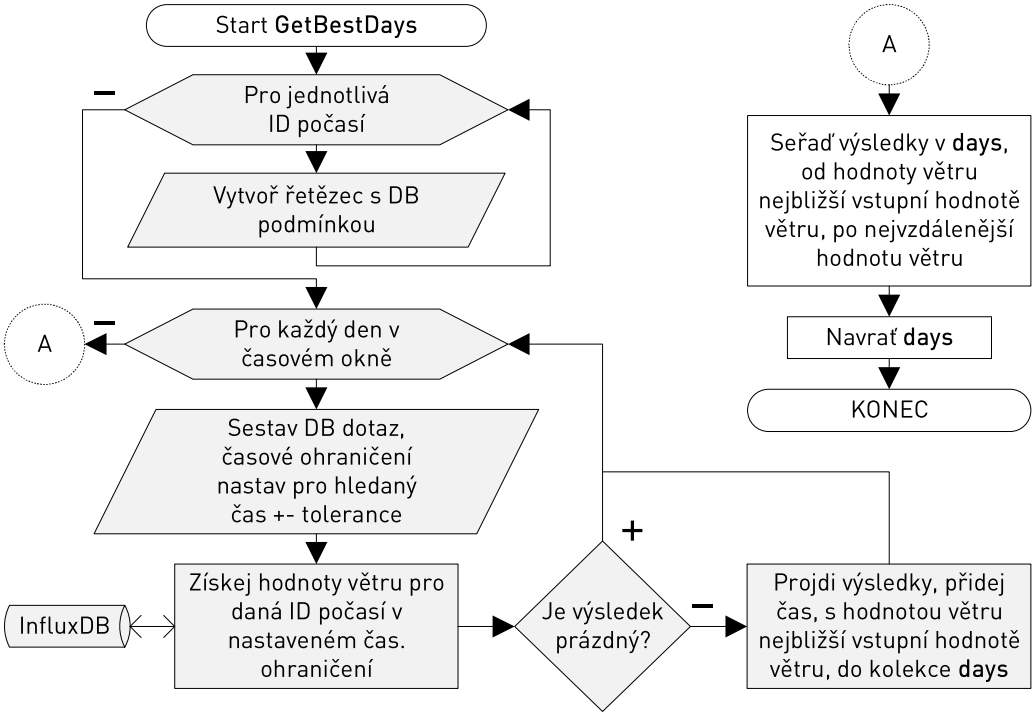
Skupina	Popis	Hodnoty stavů	Počet stavů
2xx	Bouřka	200 – 232	10
3xx	Mrholení	300 – 321	9
5xx	Déšť	500 – 531	10
6xx	Sněžení	600 – 622	11
7xx	Atmosférické jevy	701 – 781	10
8xx	Jasno	800	1
80x	Oblačnost	801 – 804	4

7.3.4 Metoda `GetBestDays`

Metoda `GetBestDays` je důležitou privátní metodou užívanou při výpočtu teplotní odchylky. Návratovou hodnotu je kolekce struktur `TimeWind`, obsahující časovou hodnotu a údaj o síle větru. Jejími parametry jsou:

- ID zařízení `devId`,
- pole hledaných ID počasí `weatherIds`,
- hledaná rychlost větru `wind`,
- hledaný čas `searchedTimeOfDay`.

Metoda zjišťuje, zda se v daný čas `searchedTimeOfDay` (s tolerancí ± časového intervalu z uživatelského nastavení), v některém z minulých dní (počet dní, ve kterých se zpětně vyhledává – taktéž konfigurováno v nastavení), na daném zařízení `devId`, vyskytlo některé z počasí uvedených v poli `weatherIds`. Pokud je nalezen více než jeden výsledek v jenom dni, je vybrán ten s rychlostí větru daného počasí nejbližší hledané rychlosti větru. Pokud je nalezeno několik výsledků z více dnů, jsou seřazeny od výsledku s nejmenším rozdílem rychlosti větru, po výsledek s největším rozdílem rychlosti větru. Takto jsou výsledky vloženy do struktur `TimeWind` a navráceny. Vývojový diagram metody je na obrázku 7.14.



Obr. 7.14: Vývojový diagram metody `GetBestDays` analyzátoru teplotní korelace.

7.3.5 Metoda `GetTemperaturesDiff`

Metoda `GetTempreraturesDiff` je druhou privátní metodou, užívanou ve výpočtu teplotní odchylky. Návratovou hodnotou je typ `double`. Jejími parametry jsou:

- ID zařízení `devId`,
- čas `searchedTime`,
- hodnota `measTempNegativeOffset` udávající posun návratové hodnoty.

Tato metoda zajišťuje samotné zjištění a výpočet odchylky – rozdílu mezi analyzovanou teplotou a teplotou vzduchu, pro jediný okamžik stanovený parametrem `searchedTime`. Od zjištěného rozdílu je dále ještě odečten posuv z parametru `measTempNegativeOffset`, jehož argumentem je v analyzátoru vždy použita návratová hodnota privátní metody `WindTempCorrection`.

$$t = t_{\text{ODU/IDU}} - t_{\text{vzduch}} - x_{\text{komp}} \quad (^{\circ}\text{C}), \quad (7.3)$$

`WindTempCorrection` vrací teplotní hodnotu, mající za cíl kompenzovat rozdíly v rychlosti větru. Je předpokládáno, že metodou `GetBestDays` nikdy nebude nalezen výsledek s rychlostí větru přesně odpovídající aktuální vyhledávané hodnotě. Kompenzace je počítána dle vztahu 7.4 jako:

$$x_{\text{komp}} = t_{\text{koeficient}}(v_{\text{hledana}} - v_{\text{nalezena}}) \quad (^{\circ}\text{C}), \quad (7.4)$$

kde x_{komp} je výsledná teplotní kompenzace, v_{hledana} je hledaná rychlost větru, v_{nalezena} je nalezená rychlost větru, a $t_{\text{koeficient}}$ je teplotní koeficient konfigurovatelný v uživatelském nastavení, udávající kolik (°C) je vykompenzováno jedním m/s. Výchozí hodnota koeficientu určená na základě pozorování během testování je $0,3\,^{\circ}\text{C}$.

Pokud je hledaná rychlost větru nižší než rychlost nalezená, znamená to, že v okamžiku zaznamenání nalezené rychlosti větru byla mikrovlnná jednotka více ochlazována. Kompenzace má zápornou hodnotu, přičemž při výpočtu teplotního rozdílu (odchylky) je odečítána, rozsah povolených teplot pro hledanou nižší rychlost větru je tedy větší. Stejný princip, ale s opačnými hodnotami, je aplikován, pokud je hledaná rychlost větru větší než nalezená.

7.3.6 Aktualizace teplotní odchylky metodou `WeatherUpdate`

Metoda `WeatherUpdate` provádí výpočet teplotní odchylky – rozdílu mezi sledovanou teplotou (IDU/ODU) a teplotou vzduchu. Snaží se přitom tuto odchylku učinit co nejpřesnější, vzhledem k aktuálnímu stavu počasí. Vývojový diagram metody je znázorněn na obrázcích 7.15 a 7.16.

Po zavolání metody je nejprve zkoušeno získat časy vhodných historických hodnot metodou `GetBestDays`, pro vstupní (aktualizované kolektorem) ID počasí a vstupní rychlost větru. ID počasí je však celá řada a je nemalá pravděpodobnost, že dané počasí ještě v minulých dnech v danou denní dobu nenastalo.

Pokud bylo první volání `GetBestDays` neúspěšné, nastává volání druhé. Tentokrát je vstupní ID počasí substituováno všemi ID počasí dané skupiny. Příklad: pakliže se nepodaří najít vhodné časy, např. pro ID 212, označující silnou bouřku, argument `GetBestDays` je namísto tohoto ID nahrazen polem ID bouřek o všech intenzitách, tj. 10 ID v rozsahu 200 – 232.

Pakliže není ani toto volání úspěšné, pole ID počasí je opět substituováno, tentokrát počasími, která nastávají nejčastěji a tím pádem je u nich největší předpoklad nálezu – jedná se o počasí skupiny 8xx, tj. jasná obloha, a různé úrovně oblačnosti (bez dalších meteorologických jevů). Tato substituce je provedena pouze v případě, pokud samotné vstupní ID počasí nepochází ze skupiny 8xx (takový postup by neměl význam). Nejprve je proveden pokus volání `GetBestDays` s ID oblačností (803, 804 – z důvodu vyšší pravděpodobnosti teplotní korespondence se vstupním stavem počasí), pokud je toto volání neúspěšné, následuje další pokus s ID jasna, skoro jasna a polojasna (800, 801, 802). Pokud se některý z těchto dvou pokusů s ID 8xx zdařil, nastane výpočet substitučního koeficientu. Tomu se věnuje samostatný úsek níže, a také část vývojového diagramu zobrazená na obrázku 7.16.

Může se stát, že se nepodaří získat časy ani jednou substitucí. V tom případě nastane poslední pokus volání `GetBestDays`, a to s nulovou hodnotou ID počasí. V takovém případě se `GetBestDays` pokusí získat jakákoli data, bez ohledu na počasí. Vzhledem k tomu, že takto získaná data nejsou nijak vázaná na aktuální vstupní ID počasí, je substituční koeficient nastaven na hodnotu pro neznámá data, která činí 1,3. Pokud ani v tomto případě nejsou navráceny žádné výsledky, neexistují pro aktuální nastavení srovnatelná data, a metoda `WeatherUpdate` je ukončena.

Pokud se jednomu z volání `GetBestDays` podařilo získat časy historických hodnot spolu s rychlostí větru, algoritmus pokračuje. Pro každý takto získaný čas je v cyklu zavolána metoda `GetTemperaturesDiff` s vykompenzováním rychlosti větru, přičemž získané teplotní odchylky jsou uloženy do kolekce `diffs`, a příslušné rychlosti větru do kolekce `winds`.

Pakliže je počet prvků v kolekci `diffs` větší, než polovina hodnoty uživatelského nastavení „počet denních hodnot tvořících průměrnou teplotní odchylku“, je vypočítána průměrná hodnota odchylek dle vztahu 7.5:

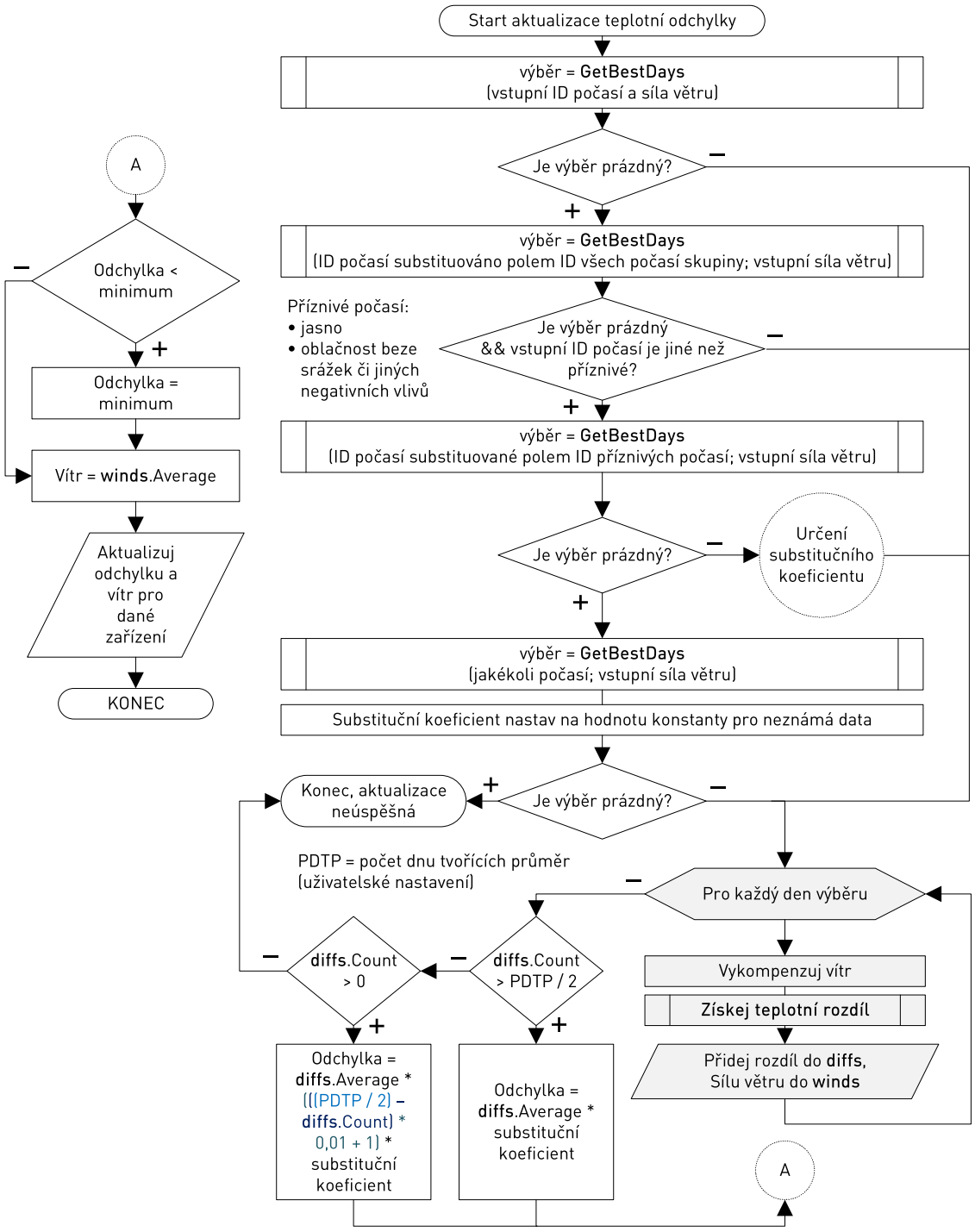
$$t_{\text{odchyl}} = s \left(\frac{1}{n} \sum_{i=1}^{n} t_i \right) \quad (^{\circ}\text{C}), \quad (7.5)$$

kde s je substituční koeficient, n počet prvků v kolekci `diffs`, a t_i prvek této kolekce (teplotní odchylka).

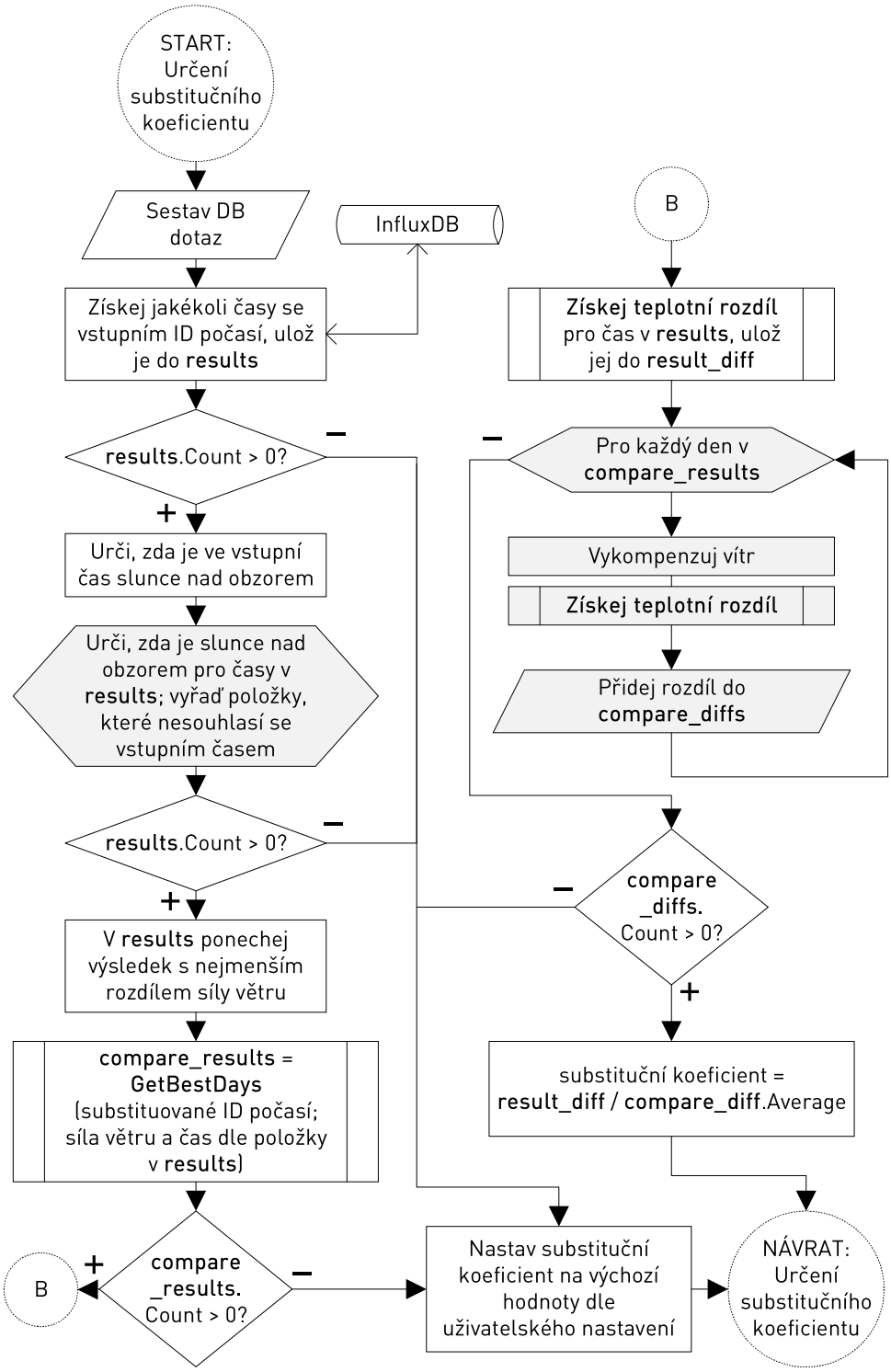
V opačném případě, kdy je počet prvků v kolekci `diffs` menší, než polovina „počtu denních hodnot tvořících průměrnou teplotní odchylku“, je průměrná hodnota odchylek vypočítána dle vztahu 7.6:

$$t_{\text{odchyl}} = s \left(\frac{1}{n} \sum_{i=1}^{n} t_i \right) \left(0,01 \left(\frac{N}{2} - n \right) + 1 \right) \quad (^{\circ}\text{C}), \quad (7.6)$$

kde s je substituční koeficient, n počet prvků v kolekci `diffs`, t_i prvek této kolekce (teplotní odchylka) a N hodnota nastavení „počtu denních hodnot tvořících průměrnou teplotní odchylku“. Tento výpočet kompenzuje každou chybějící denní hodnotu minimálního počtu zvýšením výsledné odchylky o jedno procento její hodnoty.



Obr. 7.15: Vývojový diagram aktualizace teplotní odchylky dle aktuálního počasí, 1. část, pokračování na obrázku 7.16.



Obr. 7.16: Vývojový diagram aktualizace teplotní odchylky dle aktuálního počasí, 2. část, navazující na obrázek 7.15.

Vypočítaná průměrná hodnota odchylek je následně zkontrolována, zda dosahuje minimální hodnoty definované v uživatelském nastavení. Ve výchozím nastavení je to hodnota 10 °C. Nakonec jsou zprůměrovány i hodnoty rychlostí větru v kolekci `winds`.

Pokud vše proběhlo úspěšně, je průměrná hodnota odchylek považována za finální hodnotu odchylky a uložena do kolekce `deviceDifferences`. Stejně tak zprůměrovaná rychlost větru je uložena do kolekce `deviceWinds` a čas provedení této aktualizace do kolekce `lastUpdate`. Kolekce jsou indexovány dle ID zařízení (`devId`).

Výpočet substitučního koeficientu

Substituční koeficient představuje snahu analyzátoru vyrovnat rozdíly mezi vstupním počasím a nalezeným substituovaným počasím. Funguje na principu prohledání databáze a srovnání původního a substituovaného počasí v jiný, než vstupní čas. Je vypočítáván a aplikován pouze v případě, že proběhla úspěšná substituce ID počasí skupinou 8xx (v ostatních případech má neutrální hodnotu 1). Postup algoritmu jeho výpočtu je následující:

1. Při výpočtu substitučního koeficientu je nejprve proveden databázový dotaz, snažící se získat veškeré časy teplotních záznamů s příslušnými rychlostmi větru, v nastaveném pracovním časovém okně (určeném počtem přeskočených dní a maximálním stářím ve dnech), ve kterých se vyskytuje původní vstupní ID počasí. Získané časy a rychlosti větru jsou uloženy v kolekci `results`.
2. To však znamená, že takto získané časy mohou mít oproti zjišťovanému času různý posun, a mohou se nacházet v opačné části dne (v noci). Z toho důvodu jsou metodě `WeatherUpdate` předávány i parametry `latitude` a `longitude`, se zeměpisnými souřadnicemi daného zařízení. Na základě souřadnic je pro všechny časy v kolekci `results`, určeno, zda je v daný čas nad obzorem slunce, tj. zda je den, či noc. Časy, u kterých tento údaj nekoresponduje s částí dne původního zjišťovaného času, jsou z kolekce odstraněny.
3. V dalším kroku výpočtu substitučního koeficientu, je ze zbylých časů v kolekci `results` ponechán pouze takový, jehož hodnota rychlosti větru je nejbližší vstupní rychlosti větru. Je zavolána metoda `GetBestDays`, jejímiž argumenty jsou substituované ID počasí, a zbylý čas s rychlostí větru v `results`. Získaná data jsou uložena do kolekce `compare_results`.

4. Předposledním krokem výpočtu je získání teplotní odchylky `result_diff` metodou `GetTemperaturesDiff` pro čas v `results`, přičemž rychlost větru není kompenzována (argumentem `GetTemperaturesDiff` je nula). Následně jsou získány teplotní odchylky i pro všechny časy v kolekce `compare_results`, s kompenzovanou rychlostí větru. Tyto odchylky jsou zprůměrovány a uloženy do proměnné `compare_diff`.
5. Substituční koeficient odpovídá podílu: `result_diff` / `compare_diff`.

Pokud během výpočtu některý databázový dotaz nevrátí žádné výsledky, výpočet nelze provést a koeficient je v takovém případě určen staticky, na základě uživatelsky nastavených hodnot. Výchozí hodnoty jsou uvedeny v tabulce 7.4.

Tab. 7.4: Výchozí substituční koeficienty počasí analyzátoru teplotní korelace.

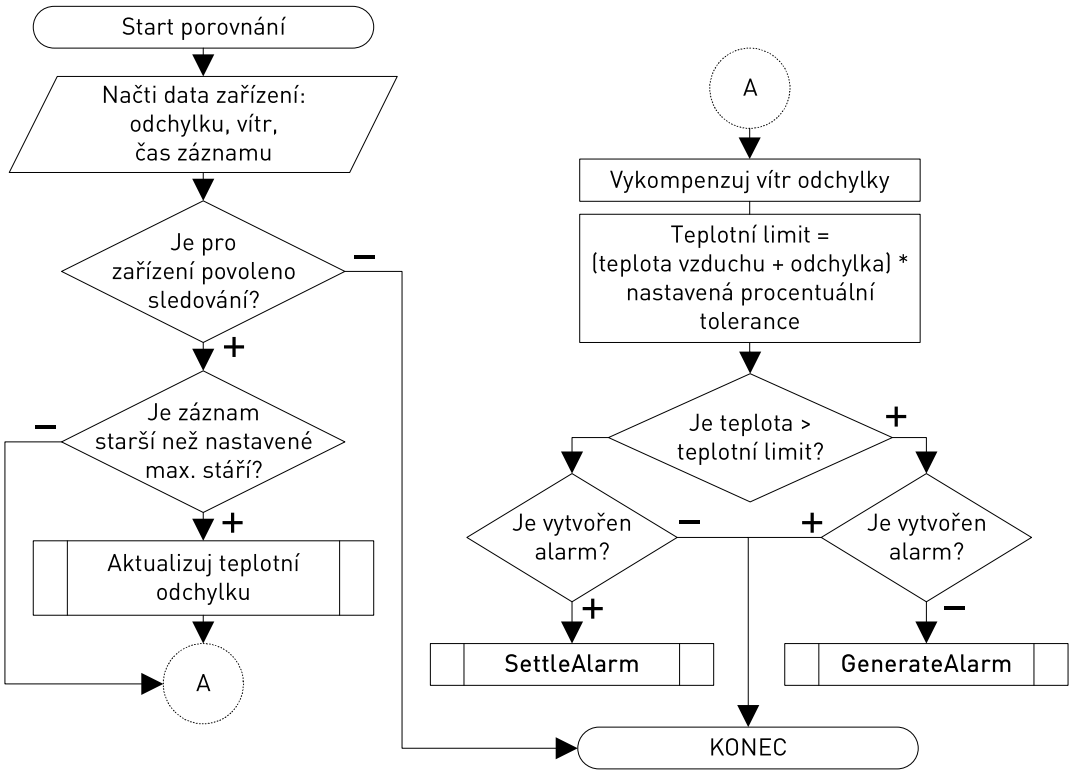
Vyhledávané počasí	Substituce jasnou oblohou (%)	Substituce zataženou oblohou (%)
Jasno	–	102
Zataženo	98	–
Mlha a ostatní atmosferické jevy	98	99
Sníh	94	95
Déšť	96	98
Mrholení	97	99
Bouřka	96	98

Určení denní doby na základě souřadnic zajišťuje knihovna CoordinateSharp od společnosti Signature Group, LLC., zveřejněná pod licencí AGPL 3.0. Využita je její třída reprezentující zeměpisnou souřadnici a metoda `isSunUp`, vracející hodnotu typu `boolean` – zda je slunce nad obzorem [46].

7.3.7 Porovnání teplot a vytvoření alarmu

Metoda `Compare` porovnává nově vyčtenou teplotní hodnotu s vypočítaným teplotním limitem, na základě hodnoty teplotní odchylky určené metodou `WeatherUpdate`. Vývojový diagram metody je na obrázku 7.17.

Nejprve je v kolekci `isWatched` zjištěno, zda je aktivní monitoring zařízení. Následně proběhne kontrola, v jaký čas byla naposledy provedena aktualizace teplotní odchylky metodou `WeatherUpdate`. Pokud staří hodnot překračuje uživatelsky nastavenou hodnotu „max. časová odchylka vyhledávaných hodnot v jednom dni“ (ve výchozím nastavení 45 minut), je vynucena okamžitá aktualizace teplotní odchylky.



Obr. 7.17: Vývojový diagram metody `Compare` analyzátoru teplotní korelace.

Poté je proveden výpočet maximální bezpečné teploty dle vztahu 7.7:

$$t_{\text{limit}} = k\ (t_{\text{vzduchu}} + t_{\text{odchyl}}) \qquad (^{\circ}\text{C}), \quad (7.7)$$

kde t_{limit} je výsledná bezpečná teplota, t_{vzduchu} teplota vzduchu, t_{odchyl} teplotní odchylka získaná metodou `WeatherUpdate`, a k je procentuální uživatelsky nastavitelný koeficient určující míru tolerance. Výchozí hodnota k je 30 %.

Nově vyčtená teplotní hodnota je porovnána s vypočítanou maximální bezpečnou teplotou t_{limit}. Je-li hodnota přesažena a neexistuje-li již kvůli tomu alarm, je alarm vytvořen. Není-li hodnota přesažena, je opět provedena kontrola, zda již není vytvořen alarm, pokud ano, je deaktivován.

7.3.8 Ukázka výpisu v módu ladění

I pro tento analyzátor je možné v nastavení aplikace povolit režim ladění. Ve výpisu 7.4 je uvedena ukázka ladícího logu analyzátoru. Význam položek je následující:

- `TA` – typ analyzátoru (**T**eplotní **A**nalýza),
- `tempOdu` – zkoumaná teplota (ODU/IDU),
- `dev` – ID zařízení,
- `diff` – získaná teplotní odchylka (průměr nalezených hodnot),
- `cnt` – počet hodnot (prvků průměru), tvořících odchylku `diff`,
- `wind` – získaná rychlost větru (průměr nalezených hodnot),
- `alter` – substituční koeficient.

Výpis 7.4: Ladící výpis analyzátoru teplotní korelace.

```
TA tempOdu dev: 33 diff: 35,196 cnt: 7 wind: 3,24 alter: 1,000
TA tempIdu dev: 33 diff: 19,784 cnt: 5 wind: 3,40 alter: 1,000
TA tempOdu dev: 34 diff: 40,362 cnt: 5 wind: 3,08 alter: 1,000
TA tempIdu dev: 34 diff: 24,039 cnt: 3 wind: 2,92 alter: 1,000
TA tempOdu dev: 43 diff: 10,998 cnt: 1 wind: 1,50 alter: 1,000
TA tempOdu dev: 44 diff: 10,692 cnt: 5 wind: 3,08 alter: 0,980
TA tempOdu dev: 47 diff: 18,443 cnt: 1 wind: 1,50 alter: 1,000
TA tempOdu dev: 51 diff: 13,611 cnt: 4 wind: 3,09 alter: 1,000
TA tempOdu dev: 52 diff: 10,422 cnt: 6 wind: 3,27 alter: 0,980
```

7.4 Statické notifikace

Princip vytváření následujících alarmů je odlišný, než v analyzátorech popsaných v předešlých kapitolách. Zatímco ty fungovaly na bázi vlastních objektových tříd, obsahujících veškerou obslužnou logiku analyzátoru, hlášení totálního výpadku a překročení hodnotového prahu je integrováno přímo ve třídách kolektorů (viz 5.4). Z kolektorů jsou pak volány metody třídy `AlarmManager`, v kódu nazývané jako „triggery“.

7.4.1 Totální výpadek

Totální výpadek je stav, při kterém zařízení neodpovídá na zasílané dotazy a jeví se jako nedostupné. Může být způsoben poruchou zařízení, či závadou na přístupové trase k němu.

Kolektory

Kolektory v případě obdržení i neobdržení odpovědi volají metodu `HasResponded` abstraktní třídy `Collector`, kterou implementují. Argumentem je hodnota `boolean` vyjadřující, zda byla obdržena odpověď, či časový limit pro odpověď vypršel (nastal tzv. timeout). Hodnota časového limitu je pro:

- SNMP kolektory – dvojnásobek obnovovacího intervalu, maximálně však 5 s,
- ICMP ping kolektory – hodnota obnovovacího intervalu.

S každou neobdrženou odpovědí metoda `HasResponded` zvyšuje hodnotu čítače `timeoutCount` o jedna. Pakliže hodnota čítače nabude hodnoty konstanty `MaxTimeoutCount` (10), je volána metoda `DeviceDownTrigger` třídy `AlarmManager` s parametrem ID zařízení, přičemž zařízení je od tohoto momentu považováno za nedostupné. Zároveň je dvakrát prodloužen obnovovací interval zasílání dalších dotazů, aby se předešlo zbytečné zátěži. Pokud čítač `timeoutCount` dosáhne dvojnásobné hodnoty konstanty `MaxTimeoutCount` (20), je obnovovací interval prodloužen na hodnotu konstanty `MaxDownRefresh` (30 s).

S každou obdrženou odpovědí metoda `HasResponded` zkontroluje hodnotu čítače `timeoutCount`. Pokud je rovna nebo větší hodnotě konstanty `MaxTimeoutCount` (10), znamená to, že kolektor v minulosti označil zařízení jako nedostupné, a je volána metoda `DeviceUpTrigger` třídy `AlarmManager`. Čítač `timeoutCount` je resetován na nulovou hodnotu a obnovovací interval nastaven na původní hodnotu

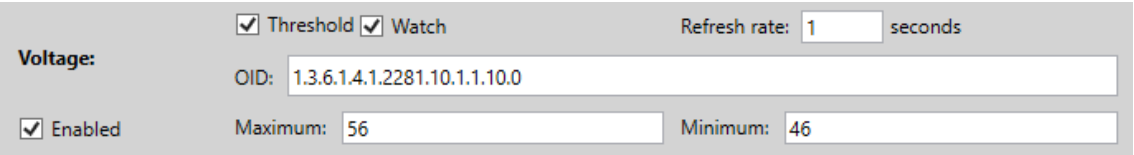
Metody ve třídě `AlarmManager`

Metoda `DeviceDownTrigger` má vytvořen čítač v kolekci `downTriggers` pro každé ID zařízení. Je zkontrolována hodnota čítače, pokud má nulovou hodnotu, je zvýšena o jedna a zavolána metoda `GenerateAlarm`. Pokud má čítač vyšší než nulovou hodnotu, je tato hodnota pouze zvýšena o jedna bez další akce. Tímto je zajištěno, že další kolektory stejného zařízení, volající `DeviceDownTrigger`, již nevytváří další duplicitní alarmy.

Metoda `DeviceUpTrigger` nejprve sníží hodnotu čítače v kolekci `downTriggers` pro dané ID zařízení o jedna. Má-li po této změně čítač nulovou hodnotu, je volána metoda `SettleAlarm`.

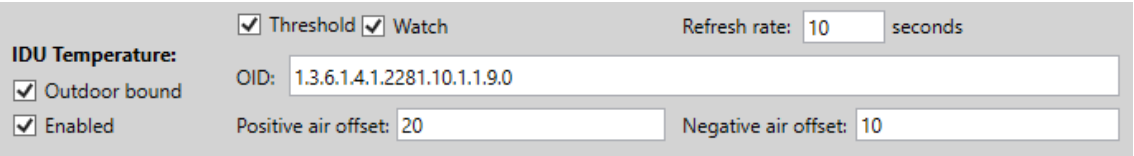
7.4.2 Překročení statických uživatelských prahů

Nastavitelné hodnoty prahů jsou v aplikaci a jejím kódu nazývané jako „thresholdy“. Uživatel má možnost definice vlastního konstantního prahu pro všechny monitorované veličiny, a to u každého zařízení v jeho nastavení. Při překročení tohoto prahu je vytvořen alarm. Je možno definovat jak horní hranici, tak i dolní hranici hodnoty. Ukázka této položky v nastavení zařízení je na obrázku 7.18.



Obr. 7.18: Prahy veličin v nastavení monitorování zařízení v aplikaci.

U teplotních veličin s vazbou na venkovní teplotu vzduchu, je princip obdobný, nenastavují se však konstantní limity, nýbrž maximální posun veličiny vůči aktuální teplotě vzduchu. Ukázka položky v nastavení zařízení je na obrázku 7.19.



Obr. 7.19: Práh teplotní veličiny s vazbou na teplotu vzduchu v nastavení monitorování zařízení v aplikaci.

Kolektory

Kolektory veličin s nastavitelnými prahy volají po úspěšném vyčtení hodnoty metodu `ThresholdCheck` abstraktní třídy `Collector`, kterou implementují. Parametrem této metody je vyčtená hodnota. Metoda `ThresholdCheck` porovnává vyčtenou hodnotu s hodnotami prahů. Při jejich překročení volá metodu `TreshExcTrigger` třídy `AlarmManager`, a nastavuje proměnnou `threshActive` na `true`.

Pakliže je vyčtená hodnota v bezpečném rozmezí a hodnota `threshActive` je `true`, je zavolána metoda `TreshSettTrigger` třídy `AlarmManager` pro deaktivaci alarmu.

Metody ve třídě `AlarmManager`

Metody `TreshExcTrigger` a `TreshSettTrigger` krom volání `GenerateAlarm`, resp. `SettleAlarm`, obsahují také logiku pro uchování a rozřazení ID vytvořených alarmů ve strukturách `AlarmIdAssign`, jejíchž instance jsou uloženy ve slovníkové kolekci `thresholdIds`, indexované dle ID zařízení.

8 Testování a výsledky aplikace

Funkčnost aplikace byla testována pomocí simulačního softwaru i reálných mikrovlnných spojů. Simulací bylo možno ověřit korektní vyčítání SNMP kolektorů, a také základní chování detekčních analyzátorů pomocí jednoduchých změn průběhů veličin. Větší důraz byl kladen na testovací nasazení aplikace na dohled reálné skupiny mikrovlnných jednotek, zatížených produkčním provozem. Tuto možnost poskytl a technicky zajistil poskytovatel služeb mikrovlnných spojů, společnost CBL Communication by light s.r.o.

8.1 Testování pomocí simulace SNMP agentů

Pro simulační zařízení podporujících protokol SNMP byl využit nástroj SNMP Agent Simulator od společnosti iReasoning [52]. Aplikace má dostupnou zkušební edici zdarma ke stažení na stránkách výrobce, její nevýhodou je však časové omezení 30 minut, po kterých se aplikace vypne. Je tak nutno při déle trvajících testech provádět její opětovný start.

Tato aplikace disponuje grafickým uživatelským rozhraním pro správu namodelovaných agentů. Podrobná nastavení, včetně nastavení samotných simulovaných hodnot, jsou potom prováděna pomocí příslušných XML souborů. Aplikace podporuje protokol SNMP ve všech verzích.

Virtuální agenti jsou vytvářeni na adrese vybraného síťového rozhraní přítomného v systému. Namísto standardního SNMP portu 161 jim lze přidělit k naslouchání jakýkoli jiný volný port. Tímto způsobem lze vytvořit libovolné množství agentů (omezené pouze systémovými prostředky daného stroje), naslouchajících na stejné IP, avšak různých portech.

XML soubor se simulačními daty lze vygenerovat buďto z MIB souboru pro dané zařízení, anebo přímým vyčtením OID i s aktuálními hodnotami ze „vzorového“ zařízení pomocí SNMP dotazů `GetNextRequest` (viz kapitola 2.3.1). Této možnosti bylo využito i při simulacích v této práci.

Ukázka struktury tohoto souboru i s konfiguračními daty je ve výpisu 8.1. Uvedený výpis obsahuje definice dvou OID:

- `iso.org.dod.internet.mgmt.mib-2.system.sysDescr`
  - vracející výrobcův popisek zařízení,
- `iso.org.dod.internet.private.enterprises.summit-development.summit-Products.uni-a.rssi`
  - vracející sílu signálu RSSI.

Výpis 8.1: Konfigurační data virtuálního SNMP agenta jednotky Summit Narrow.

```
<?xml version="1.0"?>
<SnmpSimulatorData>
    <Instances readCommunity="public">
        <Instance
         oid=".1.3.6.1.2.1.1.1.0"
         valueType="OctetString">
            <Value>
                <![CDATA[NARROW PtP Microwave radio
                man. SUMMIT DEVELOPMENT]]>
            </Value>
        </Instance>
        <Instance
         oid=".1.3.6.1.4.1.23688.1.1.5.0"
         valueType="UnsignedInteger">
            <Value>
                <![CDATA[45]]>
            </Value>
        </Instance>
    </Instances>
</SnmpSimulatorData>
```

XML tagy `Instance` označují jednotlivé položky v MIB stromu, tedy samotné veličiny a informace, které lze vyčítat. Parametry `oid` a `valueType` specifikují číslo OID a datový typ. Samotná hodnota, kterou virtuální agent simuluje, se nachází uvnitř tagu `Value` (sekce `CDATA` v XML označuje textový úsek, který nemá být interpretován a může tak obsahovat jakékoli textové řetězce).

Pomocí změn hodnot „instancí“ v tomto definičním XML souboru a následného provedení znovunačtení definic v menu aplikace Agent Simulator, lze ovlivňovat průběhy vyčítaných charakteristik.

8.2 Testování na reálných spojích

Přehled všech typů spojů, které byly monitorovány aplikací, je uveden v tabulce 8.1. Pro monitoring byl vyhrazen samostatný virtuální server v managementové síti poskytovatele, na kterém byla aplikace spuštěna. Na tomto serveru běžela vždy stabilní verze aplikace, která byla průběžně aktualizována, čímž byl zajištěn kontinuální chod a vyčítání dat. Pro vývojové účely byl poskytnut přístup do managementové sítě skrze VPN.

Tab. 8.1: Seznam mikrovlnných jednotek monitorovaných aplikací během testování.

Typ jednotky	Počet zařízení
Ceragon IP-20E	10
Ceragon IP-20G	8
Ceragon IP-20S	26
Ceragon IP-10	2
Summit BT10G	14
Summit BTD10G	6
Summit BTN10	14
Summit Narrow	11
Summit QAM	14
Summit SDV10	10
Summit UNic	12
Celkem	**127**

Celkem bylo monitorováno 127 jednotek, po dobu 51 dní. Všechny jednotky, s výjimkou jediné, tvořili linky o dvou uzlech, tudíž bez retranslačních stanic. Zbývající jednotka byla namapována v samostatné lince bez protistrany. Za celou dobu sledování bylo vytvořeno 1465 alarmů[1], z toho:

- 848 totálních výpadků, přičemž 45 % výpadků bylo kratších něž 5 minut,
- 339 alarmů z analyzátoru dlouhodobých průměrů,
- 233 alarmů z analyzátoru krátkodobých průměrů,
- 36 alarmů z analyzátoru periodičnosti,
- 9 alarmů teplotní korelace.

Většina vytvořených alarmů byla způsobena zhoršeným počasím. Pro 44 % (56) jednotek dokonce za celou dobu nebyl vytvořen jediný alarm (bez započítání totálních výpadků). Podařilo se zachytit několik skutečných neočekávaných anomálií, o kterých pojednává další kapitola 8.3.

[1]Nastavení analyzátorů bylo na výchozích hodnotách.

8.3 Odhalené anomálie reálných spojů

U následujících pěti zařízení bylo zachyceno nestandardní chování. Všechny jevy byly oznámeny a konzultovány s dohledovým centrem poskytovatele služeb.

8.3.1 Teplotně nestále jednotky

U dvou jednotek typu Ceragon IP-20S došlo k výskytu níže popsaného jevu. Jednotky jsou na sobě nezávislé – pracující na jiných linkách, v rozdílných lokalitách.

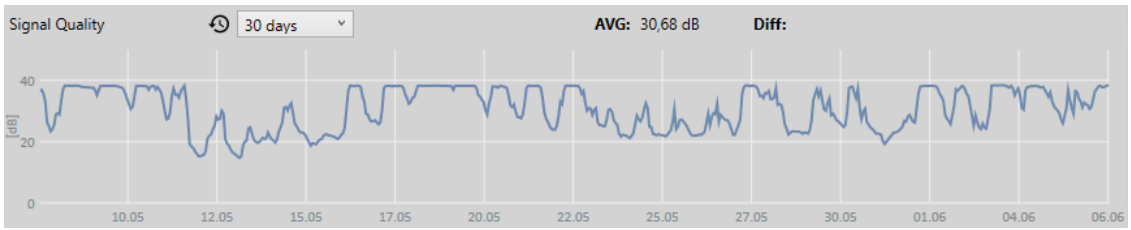
Aplikace k oběma jednotkám opakovaně vytvářela varovné alarmy jak z analýzy sledováním odklonu od průměru, tak z analyzátoru periodičnosti, kterým byla detekována denní perioda změn. Na obrázku 8.1 je snímek příslušných alarmů v aplikaci, na obrázku 8.2 pak měsíční průběh SNR jedné z jednotek zobrazený aplikací.

Na jednotkách byla zjištěna teplotní nestálost. Jak je patrné z průběhů veličin na obrázcích 8.3 a 8.4, úroveň odstupu signálu od šumu (SNR) jednotek má přímou vazbu na teplotu venkovní jednotky (ODU), korelující s teplotou vzduchu. U obou jednotek je průběh velice obdobný. Kolísání SNR nastává v pravidelných denních intervalech, se změnami až o 20 dB, v závislosti na teplotních propadech, především v nočních dobách. Modulace linky byla mimo nepříznivá počasí neměnná. Dle vyjádření dohledového centra poskytovatele, je toto chování nestandardní a značí pravděpodobnou závadu těchto jednotek.

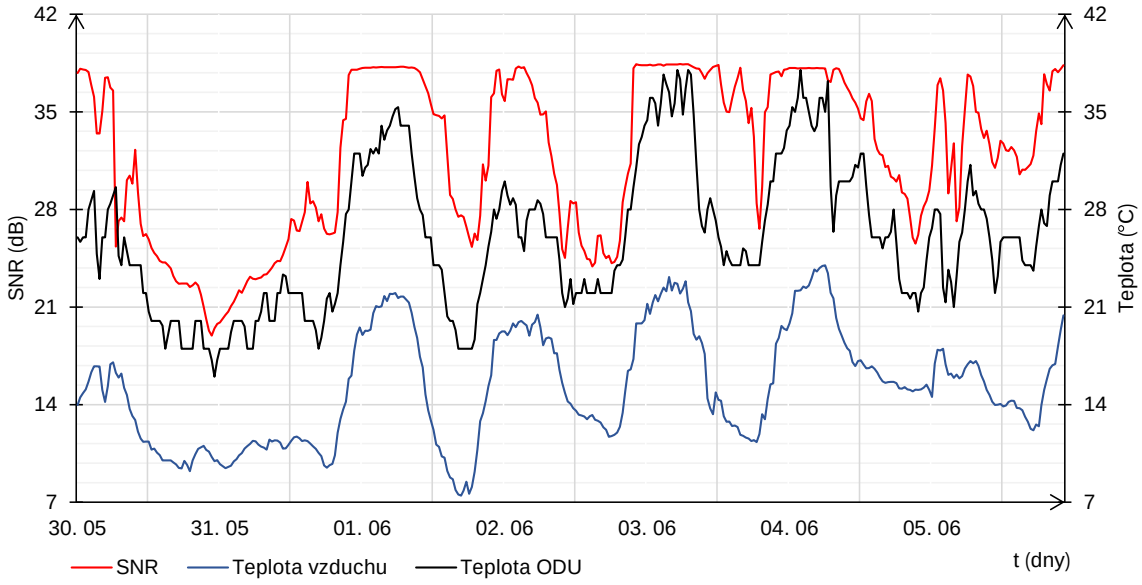
Current | Acknowledged | Settled (acknowledged) | Settled (unaddressed)

ACK	Rank	Timestamp	Link	Site	Measurement	Description	Value
☐	Warning	06.06.2020 11:19:02	Ceragon IP-20S	B	Quality	Value exceeded longterm average.	34,63
☐	Warning	06.06.2020 07:21:00	Ceragon IP-20S	B	Quality	Value exceeded longterm average.	35,08
☐	Info	04.06.2020 20:38:43	Ceragon IP-20S	B	Quality	Values have repetitive character with period of 1440,00 minutes.	-
☐	Info	04.06.2020 20:38:43	Ceragon IP-20S	B	Quality	Values have repetitive character with period of 1440,00 minutes.	-

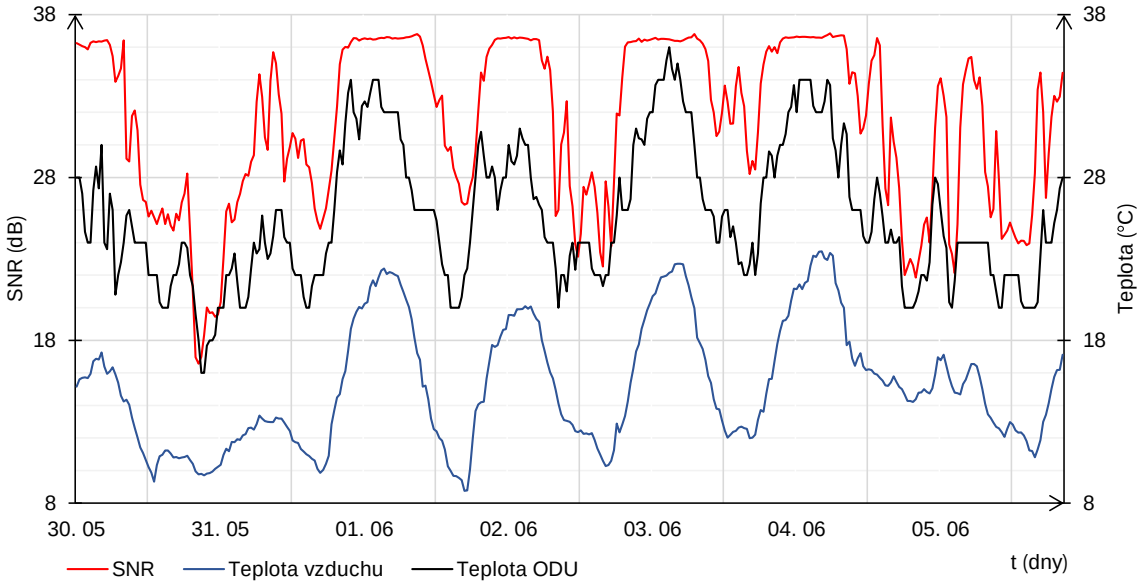
Obr. 8.1: Snímek vytvořených alarmů teplotně nestálých jednotek v aplikaci. Identifikační názvy linek byly skryty z důvodu citlivých informací.



Obr. 8.2: Snímek z aplikace s měsíčním průběhem SNR teplotně nestálé jednotky.



Obr. 8.3: Průběh SNR, teploty ODU a venkovní teploty, na první z teplotně nestálých jednotek.

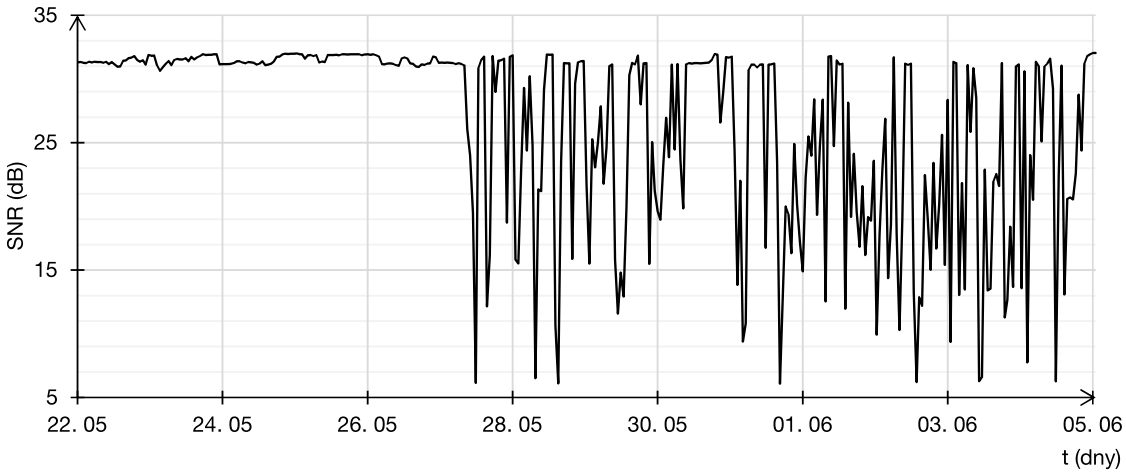


Obr. 8.4: Průběh SNR, teploty ODU a venkovní teploty, na druhé z teplotně nestálých jednotek.

8.3.2 Časté skokové změny SNR na jednotce

Další objevená anomálie se týká jednotky Ceragon IP-20E. Bylo zjištěno, že od určitého momentu dochází k častému kolísání úrovně SNR, pouze však na jedné straně linky, čímž se nejedná o změny modulace.[2] Na jednotce dochází několikrát denně k náhlému přechodu mezi hladinami SNR cca 31 dB, a cca 6 dB. Protější strana spoje má trvale úroveň SNR o hladině 6 dB. Modulace spoje je dlouhodobě základní – BPSK (viz kapitola 1.3.1). Situace byla konzultována s dohledovým centrem poskytovatele, a bylo potvrzeno, že se jedná skutečně o závadný stav. Může se jednat o závadu zařízení i externí rušení. Průběh hodnoty SNR, včetně několika dní před počátkem jevu, je zobrazen na obrázku 8.5.[3]

Aplikace k danému zařízení vytvářela alarmy na základě analýzy odklonu od průměru (dlouhodobého i krátkodobého) i analyzátoru periodičnosti.



Obr. 8.5: Dvoutýdenní průběh SNR na jednotce s častými změnami modulace.

[2]Pokud by změny byly oboustranné, mohlo by se jednat o projev funkce ACM (viz kapitola 1.8.1), ovšem za předpokladu její korektní funkčnosti, pouze v případě zhoršeného počasí, přičemž ani to by u tohoto zařízení neplatilo.

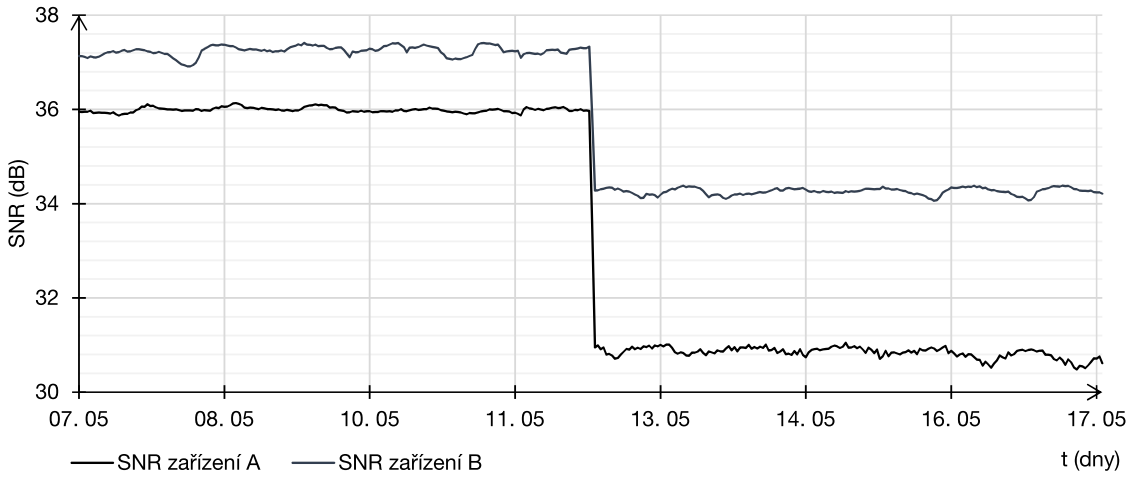
[3]Jelikož se jedná o dvoutýdenní časový úsek, období s častými změnami jsou vzhledem k rozlišení grafu průměrována a může se tak zdát, že úroveň SNR nabývala vícero hladin. Ve skutečnosti se jedná výhradně o skoky mezi hladinami, zmíněnými výše.

8.3.3 Nesprávná funkčnost ACM

Na dvou protějších jednotkách (stejné linky) Summit BT10G byla zaznamenána náhlá změna SNR, související se změnou modulace na lince. Jednalo se o skokový pokles hodnoty SNR cca o 4 dB, tudíž přechod na výkonnější typ modulace s větším počtem stavů. Stalo se tak však v době relativně dobrého počasí (v daný den), kdy ke změně v daný okamžik nebyl důvod. Je tedy pravděpodobné, že ještě před začátkem sledování pomocí aplikace, došlo nejspíše v důsledku zhoršeného počasí, k přepnutí modulace funkcí ACM (viz kapitola 1.8.1) na méně výkonný typ modulace s menším počtem stavů, avšak s nižšími nároky na úroveň SNR. Funkce ACM však pravděpodobně nefungovala korektně, jelikož zařízení v této modulaci setrvala minimálně několik týdnů, než došlo k přepnutí zpět na původní úroveň.

Situace byla konzultována s dohledovým centrem poskytovatele, bylo potvrzeno, že prahy aktivace jednotlivých úrovní ACM již zde nemusí fungovat korektně.

Zachycené průběhy SNR obou zařízení linky jsou na obrázku 8.6.



Obr. 8.6: Skoková změna SNR na zařízení z důvodu přepnutí modulace.

9 Měření datové zátěže

Měření v této kapitole bylo provedeno s namapovanými reálnými jednotkami (viz kapitola 8.2. Celkový počet monitorovaných zařízení tedy odpovídá počtu 127, v době měření však jedna linka trpěla totálním výpadkem, což znamená, že bude dále uvažován počet 125 zařízení.

Zařízení však neměla jednotnou konfiguraci (resp. stejný počet aktivních kolektorů), nýbrž konfiguraci lišící se dle typu jednotky, jelikož monitorování některých veličin nebylo na některých typech zařízení podporováno. Dále je třeba uvažovat rozdílný obnovovací interval veličin, který navíc pro každé zařízení může být nastaven individuálně. Během testování však bylo postačující a vhodné, mít všechny obnovovací intervaly dané veličiny nastaveny na shodné hodnoty. Souhrnná tabulka 9.1 uvádí počet všech aktivních kolektorů pro 125 zařízení a jejich obnovovacích intervalů.[1]

Tab. 9.1: Počet aktivních kolektorů dle typu během testování aplikace.

Typ kolektoru	Veličina	Počet kolektorů	Obnovovací interval (s)
SNMP	Síla signálu	125	1
	Kvalita signálu (SNR)	125	1
	Datová rychlost Tx	48	10
	Datová rychlost Rx	48	10
	Teplota ODU	123	300
	Teplota IDU	44	300
	Napětí	65	5
	Uptime	125	10
	Název zařízení	125	600
ICMP	Ping	125	1
HTTP API	Počasí	1	300

Na základě tabulky 9.1 by aplikace dle teoretického výpočtu měla za časový interval jedné minuty (první minuty po spuštění) zaslat:

- 17 398 SNMP PDU,
- 7 500 ICMP datagramů,
- 125 HTTP GET požadavků.

a přijmout odpovídající množství paketů.

[1] Data přenášená mezi aplikací a InfluxDB v této kapitole nejsou uvažována, předpokládá se provoz databáze na stejném stroji kde je aplikace spuštěna, a komunikace v rámci místního `loopback` rozhraní.

9.1 Síťové přenosy

Pro měření skutečného síťového provozu byl použit software Wireshark. Měření bylo provedeno pro prvních 5 minut běhu aplikace, tak, aby se stihla stáhnout data počasí pro všechna zařízení. Ze zachyceného provozu byla pomocí filtru ponechána pouze data protokolů, souvisejících s aplikací, viz podoba filtru níže.

```
snmp or icmp or json or http
```

Je nutno uvažovat nepřesnost měření v rámci jednotek sekund, jelikož vlákna kolektorů startují a začínají vyčítat na pozadí, ještě předtím, než se objeví hlavní uživatelské rozhraní. Na serveru provozujícím aplikaci trval start aplikace, od spuštění `.exe` souboru, po promítnutí hlavní okna, průměrně 5 sekund. Dále je nutno je nutno též uvažovat ztracené a zahozené SNMP PDU a ICMP datagramy.

Statistiku počtu ztracených SNMP požadavků, ke kterým nebyla obdržena odpověď, vypisuje sama aplikace, a to v intervalu každých 5 minut. Při testování činila obvyklá hladina za tento časový úsek cca 600 ztracených SNMP požadavků, což při teoretickém počtu odeslaných 86 990 PDU za 5 minut, odpovídá ztrátě cca 0,7 %.[2]

Za časový interval 5 minut bylo **odesláno**:

- 119 090 ethernetových rámců, o velikosti 9,70 MiB[3], s rychlostí 271 kb/s,
  - 119 090 IPv4 paketů, o velikosti 8,11 MiB, s rychlostí 227 kb/s,
    - 82 179 UDP datagramů, o velikosti 4,41 MiB, s rychlostí 123 kb/s,
      - 82 179 SNMP PDU, o velikosti 3,79 MiB, s rychlostí 106 kb/s,
    - 126 TCP segmentů, o velikosti 37,85 KiB,
      - 126 HTTP požadavků, o velikosti 17,70 KiB,
    - 36 785 ICMP datagramů, o velikosti 1,40 MiB, s rychlostí 39 kb/s.

Za stejný časový interval 5 minut bylo **přijato**:

- 117 799 ethernetových rámců, o velikosti 9,92 MiB, s rychlostí 277 kb/s,
  - 117 799 IPv4 paketů, o velikosti 8,34 MiB, s rychlostí 233 kb/s,
    - 81 253 UDP datagramů, o velikosti 4,60 MiB, s rychlostí 128 kb/s,
      - 81 253 SNMP PDU, o velikosti 3,99 MiB, s rychlostí 111 kb/s,
    - 125 TCP segmentů, o velikosti 258,70 KiB,
      - 125 HTTP odpovědí, o velikosti 156,80 KiB,
        - 125 JSON objektů, o velikosti 57,36 KiB,
    - 36 421 ICMP datagramů, o velikosti 1,39 MiB, s rychlostí 38 kb/s.

[2] Tato ztráta je běžná, SNMP je enkapsulováno nespojovaným protokolem UDP, viz kapitola 2.
[3] 1 MiB (mebibajt) = 1024 KiB (kibibajt) = 1 048 576 bajtů.

Celkově bylo za časový interval 5 minut **přeneseno**:

- 236 889 ethernetových rámců, o velikosti 19,62 MiB, s rychlostí 549 kb/s,
  - 236 889 IPv4 paketů, o velikosti 16,45 MiB, s rychlostí 461 kb/s,
    - 163 432 UDP datagramů, o velikosti 9,02 MiB, s rychlostí 251 kb/s,
      - 163 432 SNMP PDU, o velikosti 7,78 MiB, s rychlostí 217 kb/s,
    - 251 TCP segmentů, o velikosti 296,55 KiB,
      - 251 HTTP požadavků, o velikosti 174,50 KiB,
        - 125 JSON objektů, o velikosti 57,36 KiB,
    - 73 206 ICMP datagramů, o velikosti 2,79 MiB, s rychlostí 78 kb/s.

Mezi aplikací a jediným zařízením, s aktivním vyčítáním všech veličin, je za časový interval 5 minut na fyzické vrstvě obousměrně přeneseno 163,19 KiB, což odpovídá rychlosti 4,463 kb/s, neboli 0,54 KiB/s (aktualizace informací o počasí není v tomto případě uvažována, nejedná se o přenos mezi aplikací a zařízením).

Je-li uvažován neměnný počet monitorovaných zařízení 125, konstantní rychlost na fyzické vrstvě (ethernetových rámců) 549 kb/s (67,02 KiB/s) a přenos v obou směrech, je aplikací přeneseno:

- za 1 minutu 3,93 MiB;
- za 1 hodinu 235,62 MiB;
- za 1 den 5,52 GiB;
- za 1 týden 38,65 GiB;
- za 1 měsíc 165,67 GiB;
- za 1 rok 2 015,63 GiB.

9.2 InfluxDB

Úložiště InfluxDB je v souborovém systém členěno dle tzv. retenčních politik (RP – viz kapitola 3.2. Po naplnění časové kapacity úložiště RP již zůstává objem uložených dat v daném úložišti přibližně stejný, pakliže je zachován časový interval zápisu dat. Parametry vytvořených RP v databázi aplikace jsou přiblíženy v kapitole 5.2.3.

Po uplynutí testovací doby 51 dní bylo možné týdenní a měsíční RP považovat za naplněné, zatímco roční RP byla naplněna ze 14 %.

Dosažené velikosti úložiště za časový interval 51 dnů, pro stejnou konfiguraci aplikace jako v předchozí kapitole, byly následující:

- týdenní RP – 736 MiB,
- měsíční RP – 180 MiB,
- roční RP – 207 MiB (naplněno ze 14 %).

9.3 SQLite

Databázové soubory SQLite jsou aplikací využívány pro ukládání definic zařízení, linek, a vytvořených alarmů. Data jsou v souboru strukturována po stránkách, kdy jedna stránka může nabývat velikosti mocniny dvou, od 512 bajtů po 65 536 bajtů. Maximální počet stránek je $2^{31} - 2$, tj. 2 147 483 646 bajtů, což při největší použité velikosti stránky činí maximální velikost databáze 128 TiB [53].

Velikost databází po testovacím období činila:

- pro databázi `DeviceData.db`,
  - definováno 129 zařízení a 65 linek,
  - 20 stránek po 4 096 bajtech,
  - o celkové velikosti 80 KiB;
- pro databázi `AlarmData.db`,
  - 1 465 alarmů,
  - 30 stránek po 4 096 bajtech,
  - o celkové velikosti 120 KiB.

Závěr

Byla vytvořena aplikace, schopná vyčítání různých veličin ze vzdálených mikrovlnných jednotek na základě nastavených SNMP OID, samozřejmostí je také měření latence mezi aplikací a sledovanými zařízeními. Dále jsou stahovány informace o počasí pro každou jednotlivou lokalitu, přičemž tyto informace jsou následně využity při analýze vyčtených dat a stanovení teplotních limitů. Vyčtené hodnoty veličin jsou uživateli zobrazovány v přehledných grafických průbězích s možností přepínání zobrazovaného časového úseku. Aktuální stav počasí je zobrazován v příslušném informačním panelu. Pro zvýšení uživatelského pohodlí je též obsažena funkce zobrazení lokalit s body umístění mikrovlnných jednotek na mapě, a funkce vykreslení terénního profilu mezi mikrovlnnými jednotkami. Vyčtená data aplikace jsou ukládána do databáze InfluxDB, a konfigurační data do databází SQLite, což umožňuje snadné prohlížení a editaci dat pomocí externích databázových nástrojů.

Nad vyčtenými daty je prováděna automatická analýza pomocí třech různých typů analyzátorů, bez nutnosti konfigurace ze strany uživatele. Každý z těchto analyzátorů funguje na odlišném principu, jedná se o průměrování hodnot v různých časových oknech, analýzu periodičností a dynamický výpočet bezpečných rozsahů teplotních veličin. Obsažena je také možnost definice vlastních prahů vyčítaných veličin. Nepostradatelnou vestavěnou funkcí je oznamování totálních výpadků. Pro detekované anomálie a kritické události jsou pro účel indikace uživateli vytvářeny „alarmy“, zobrazované v příslušném panelu hlavního okna aplikace. Alarmy je možno třídit jejich potvrzováním, či je zpětně dohledat pro každé zařízení.

Pomocí automatických analýz bylo na testovaných zařízeních odhaleno několik anomálií, které byly nahlášeny a konzultovány s dohledovým centrem společnosti CBL Communication by light s.r.o., se kterou bylo během tvorby aplikace spolupracováno. Aplikace obdržela kladný ohlas ze strany operátorů dohledového centra společnosti.

Aplikace byla testována na počtu 127 sledovaných mikrovlnných jednotek. Během tohoto testování byla odezva aplikace plynulá a bez významných prodlev. Při sledování ještě většího počtu zařízení však může být naraženo na systémový limit počtu vytvořených vláken na jeden proces, či zpoždění způsobené systémovou režií těchto vláken, a to z důvodu implementace základního modelu správy vláken v aplikaci. Výhodou tohoto řešení je však možnost pracovat i s velice krátkými obnovovacími intervaly pro vyčítání veličin. Je nicméně vhodné při dalším vývoji aplikace implementovat model sdílení omezeného počtu vytvořených vláken pro zpracování vícero úloh.

Další vhodnou úpravou je vložení nové vrstvy mezi analyzátory dat a správce alarmů, která by regulovala přicházející podněty k vytváření alarmů, na základě aktuálního počasí a změn modulace, čímž by se omezila zátěž uživatele planými či duplicitními alarmy. Tato „mezivrstva“ by informovala uživatele až v případě, kdy by došlo k návratu počasí do příznivého stavu a některá ze sledovaných veličin by se nevracela do původních hodnot.

Samotných nových funkcí vhodných k další implementaci je celá řada. Lze jmenovat například stahování radarových meteorologických snímků, ze kterých by na základě souřadnic zařízení bylo možné vyčítat přesnou intenzitu srážek v lokalitě, dále funkce hromadného importu a exportu definic zařízení, například ze souborů typu CSV. Pro analýzu dat lze pak například uvažovat použití metod strojového učení.

Literatura

[1] KASAL, Miroslav. *Směrové a družicové spoje: přednášky*. Vyd. 2. V Brně: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2003. ISBN 80-214-2496-6.

[2] Principy digitální mikrovlnné komunikace. In: *Alcoma* [online]. Praha: ALCOMA, 2012 [cit. 2019-10-21]. Dostupné z: `https://www.alcoma.cz/media/document/2-manual-cz-principy-digitalni-mikrovlnne-komunikace-2.1.pdf`

[3] Strategie správy rádiového spektra. *Český telekomunikační úřad* [online]. Praha: Český telekomunikační úřad, 2015 [cit. 2019-11-25]. Dostupné z: `https://www.ctu.cz/sites/default/files/obsah/stranky/49264/soubory/strategiesprspektra2015.pdf`

[4] VODRÁŽKA, Jiří a Zdeněk BRABEC. *Technologie k připojení základnových stanic mobilní sítě k páteřní síti* [online]. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra telekomunikační techniky, 2015 [cit. 2019-11-24]. Dostupné z: `https://www.ctu.cz/sites/default/files/obsah/o-ctu/66411/soubory/cvut-ctubackhaultechnologie70def-tiskkorekce01002-vodafone-4-4-2016.pdf`

[5] Alcoma Radio Relay Link. In: *Wikimedia Commons* [online]. San Francisco: Wikimedia Foundation, 2010 [cit. 2019-10-29]. Dostupné z: `https://commons.wikimedia.org/wiki/File:Alcoma.bg_Radio_relay_link.jpg`

[6] HANUS, Stanislav. *Bezdrátové a mobilní komunikace*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2001. ISBN 80-214-1833-8.

[7] PRAVDA, Ivan. *Moderní modulační metody a jejich aplikace* [online]. Praha: České vysoké učení technické v Praze, 2015 [cit. 2019-11-09]. Dostupné z: `https://publi.cz/books/234/01.html`

[8] VOLPATO, Paolo. *An introduction to packet microwave systems and technologies*. Boston: Artech House, 2017. Artech House microwave library. ISBN 978-1-63081-331-4.

[9] NAGATA, Kengo, Yasuyoshi KOJIMA, Takefumi HIRAGURI a Yasushi TAKATORI. Wireless Local Area Network Standardization of IEEE 802.11 and the Wi-Fi Alliance. In: *NTT Technical Review* [online]. Yokosuka: Nippon Telegraph and Telephone, 2010 [cit. 2019-10-28]. Dostupné z: https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201004gls.html

[10] BELANGER, Phil. WiFi isn't short for "Wireless Fidelity." In: *BoingBong* [online]. [cit. 2019-10-28]. Dostupné z: https://boingboing.net/2005/11/08/wifi-isnt-short-for.html

[11] LEA, Perry. *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security*. Birmingham: Packt Publishing, 2018. ISBN 978-1788470599.

[12] MOLNÁR, Karol. *Hardware počítačových sítí*. V Brně: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. ISBN 978-80-214-4449-2.

[13] WiMAX IEEE 802.16 Technology. *Electronics Notes* [online]. [cit. 2019-11-24]. Dostupné z: https://www.electronics-notes.com/articles/connectivity/wimax/what-is-wimax-802-16-technology-basics.php

[14] ACM: Adaptive Coding and Modulation. *Microwave Link* [online]. Wireless Excellence Limited, 2019 [cit. 2019-12-23]. Dostupné z: https://www.microwave-link.com/microwave/acm-adaptive-coding-modulation/

[15] KOZIEROK, Charles M. *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. San Francisco: No Starch Press, 2005, 1539 s. ISBN 978-1-59327-047-6.

[16] MAURO, Douglas R. a Kevin J. SCHMIDT. *Essential SNMP*. 2nd ed. Sebastopol, CA: O'Reilly, 2005, 442 s. ISBN 05-960-0840-6.

[17] *RFC Index* [online]. Wilmington: Internet Engineering Task Force, 1969- [cit. 2019-12-19]. Dostupné z: https://tools.ietf.org/rfc/

[18] Cisco Global Site Selector Administration Guide. *Cisco* [online]. San Jose: Cisco, 2007 [cit. 2019-12-21]. Dostupné z: https://www.cisco.com/c/en/us/td/docs/app_ntwk_services/data_center_app_services/gss4400series/v2-0/administration/guide/gssadmgd/SNMP.html

[19] MIB Download. *Summit Development* [online]. Tachlovice: Summit Development [cit. 2019-12-21]. Dostupné z: https://www.mysummitd.cz/cz/files/314

[20] BT MIB Download. *Summit Development* [online]. Tachlovice: Summit Development [cit. 2019-12-21]. Dostupné z: https://www.mysummitd.cz/cz/files/482

[21] LibreNMS: librenms/mibs/ceraos. *GitHub* [online]. San Francisco: GitHub, 2017 [cit. 2019-12-21]. Dostupné z: https://github.com/librenms/librenms/tree/master/mibs/ceraos

[22] ALBAHARI, Joseph a Ben ALBAHARI. *C# 7.0 in a nutshell*. 7th edition. Sebastopol: O'Reilly, 2018, 1070 s. ISBN 978-1-491-98765-0.

[23] What is .NET Framework? A software development framework. *Microsoft* [online]. Redmond: Microsoft, 2019 [cit. 2019-12-21]. Dostupné z: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework

[24] RUDOLPH, Kevin. *A Comparison of NoSQL Time Series Databases*. Berlin: GRIN Publishing, 2015. ISBN 978-3656965763.

[25] InfluxDB 1.x: Open Source Time Series Platform. *InfluxData* [online]. San Francisco, CA: InfluxData, 2020 [cit. 2020-05-31]. Dostupné z: https://www.influxdata.com/time-series-platform/

[26] InfluxData Documentation: Downsample and retain data. *InfluxData* [online]. San Francisco, CA: InfluxData, 2020 [cit. 2020-05-31]. Dostupné z: https://docs.influxdata.com/influxdb/v1.8/guides/downsample_and_retain/

[27] About SQLite. *SQLite* [online]. Charlotte, NC: SQLite Consortium, 2004 [cit. 2020-06-01]. Dostupné z: https://www.sqlite.org/about.html

[28] InfluxDB Client for .NET. *GitHub* [online]. San Francisco: GitHub, 2020 [cit. 2020-06-01]. Dostupné z: https://github.com/MikaelGRA/InfluxDB.Client

[29] SQLite-net. *GitHub* [online]. San Francisco: GitHub, 2020 [cit. 2020-06-01]. Dostupné z: https://github.com/praeclarum/sqlite-net

[30] Lextudio/sharpsnmplib. *GitHub* [online]. San Francisco: GitHub, 2019 [cit. 2020-06-01]. Dostupné z: https://github.com/lextudio/sharpsnmplib

[31] LiveCharts. *GitHub* [online]. San Francisco: GitHub, 2020 [cit. 2020-06-01]. Dostupné z: https://github.com/Live-Charts/Live-Charts

[32] *OpenWeatherMap* [online]. London: OpenWeather, 2020 [cit. 2020-06-05]. Dostupné z: https://openweathermap.org/

[33] *MapQuest Developer: Open Elevation API* [online]. Denver, CO: MapQuest, 2020 [cit. 2020-06-05]. Dostupné z: https://developer.mapquest.com/documentation/open/elevation-api/elevation-chart/get/

[34] *Mapy API* [online]. Praha: Seznam, 2020 [cit. 2020-06-05]. Dostupné z: https://api.mapy.cz/

[35] *Tobi Oetiker's MRTG - The Multi Router Traffic Grapher* [online]. Olten: OETIKER+PARTNER, 2017 [cit. 2020-06-01]. Dostupné z: https://oss.oetiker.ch/mrtg/

[36] *Zabbix: Network Monitoring* [online]. Brooklyn, NY: Zabbix, 2020 [cit. 2020-06-01]. Dostupné z: https://www.zabbix.com/network_monitoring

[37] *SNMP monitoring with PRTG* [online]. Nuremberg: Paessler, 2020 [cit. 2020-06-01]. Dostupné z: https://www.paessler.com/snmp_monitor

[38] *SNMPc Enterprise* [online]. Saratoga, CA: Castle Rock Computing, 2020 [cit. 2020-06-01]. Dostupné z: https://www.castlerock.com/products/snmpc/

[39] STORIMER, Jesse. How many threads is too many? *Jesse Storimer* [online]. 2013 [cit. 2020-06-01]. Dostupné z: https://www.jstorimer.com/blogs/workingwithcode/7970125-how-many-threads-is-too-many

[40] Using My.Settings in Visual Basic 2005. *Microsoft Docs* [online]. Redmond: Microsoft, 2012 [cit. 2020-06-02]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/ms379611(v=vs.80)

[41] *Swiftyspiffy/OpenWeatherMap-API - C# Library* [online]. San Francisco: GitHub, 2020 [cit. 2020-06-03]. Dostupné z: https://github.com/swiftyspiffy/OpenWeatherMap-API-CSharp

[42] *Json.NET: Popular high-performance JSON framework for .NET* [online]. Newtonsoft, 2020 [cit. 2020-06-08]. Dostupné z: https://www.newtonsoft.com/json

[43] STERNBERG, Jonathan. Support disparate time intervals and more advanced time in WHERE clauses: influxdata/influxdb/issues. *GitHub* [online]. San Francisco: GitHub, 2016 [cit. 2020-05-31]. Dostupné z: https://github.com/influxdata/influxdb/issues/7530

[44] GIANNOUDIS, Jani. Time Period Library for .NET. *CodeProject* [online]. Toronto, 2017 [cit. 2020-31-05]. Dostupné z: https://www.codeproject.com/Articles/168662/Time-Period-Library-for-NET

[45] Weather Conditions. *OpenWeatherMap* [online]. London, 2012 [cit. 2020-06-01]. Dostupné z: https://openweathermap.org/weather-conditions

[46] CoordinateSharp: Developer Guide. *CoordinateSharp* [online]. Chattanooga, 2020 [cit. 2020-06-05]. Dostupné z: https://coordinatesharp.com/DeveloperGuide

[47] PUECH, Tom, Matthieu BOUSSARD, Anthony D-AMATO a Gaëtan MILLERAND. A Fully Automated Periodicity Detection in Time Series. In: *Advanced Analytics and Learning on Temporal Data*. Cham: Springer International Publishing, 2020, 2020-01-23, s. 43-54. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-39098-3_4. ISBN 978-3-030-39097-6. Dostupné také z: http://link.springer.com/10.1007/978-3-030-39098-3_4

[48] SMÉKAL, Zdeněk. *Analýza signálů a soustav - BASS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav komunikací, 2012. ISBN 978-80-214-4453-9.

[49] HLAVÁČ, Václav. Fourierova transformace v 1D a 2D. In: *CIIRC: Český institut informatiky, robotiky a kybernetiky* [online]. Praha: České vysoké učení technické v Praze, 2018 [cit. 2020-05-26]. Dostupné z: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/12FourierTxCz.pdf

[50] HAGEMAN, Steve. DSPLib - FFT / DFT Fourier Transform Library for .NET 4. *CodeProject* [online]. Toronto, 2018 [cit. 2020-05-26]. Dostupné z: https://www.codeproject.com/Articles/1107480/DSPLib-FFT-DFT-Fourier-Transform-Library-for-NET-6

[51] LYONS, Richard. *Understanding Digital Signal Processing: Unders Digital Signal Proces*. Third edition. Upper Saddle River: Prentice Hall, 2010. ISBN 978-0137027415.

[52] SNMP Agent Simulator. *Network Management Solutions* [online]. Markham: iReasoning, 2020 [cit. 2020-06-02]. Dostupné z: http://ireasoning.com/snmpsimulator.shtml

[53] Database File Format. *SQLite* [online]. Charlotte, NC: SQLite Consortium, 2004 [cit. 2020-06-07]. Dostupné z: https://www.sqlite.org/fileformat.html

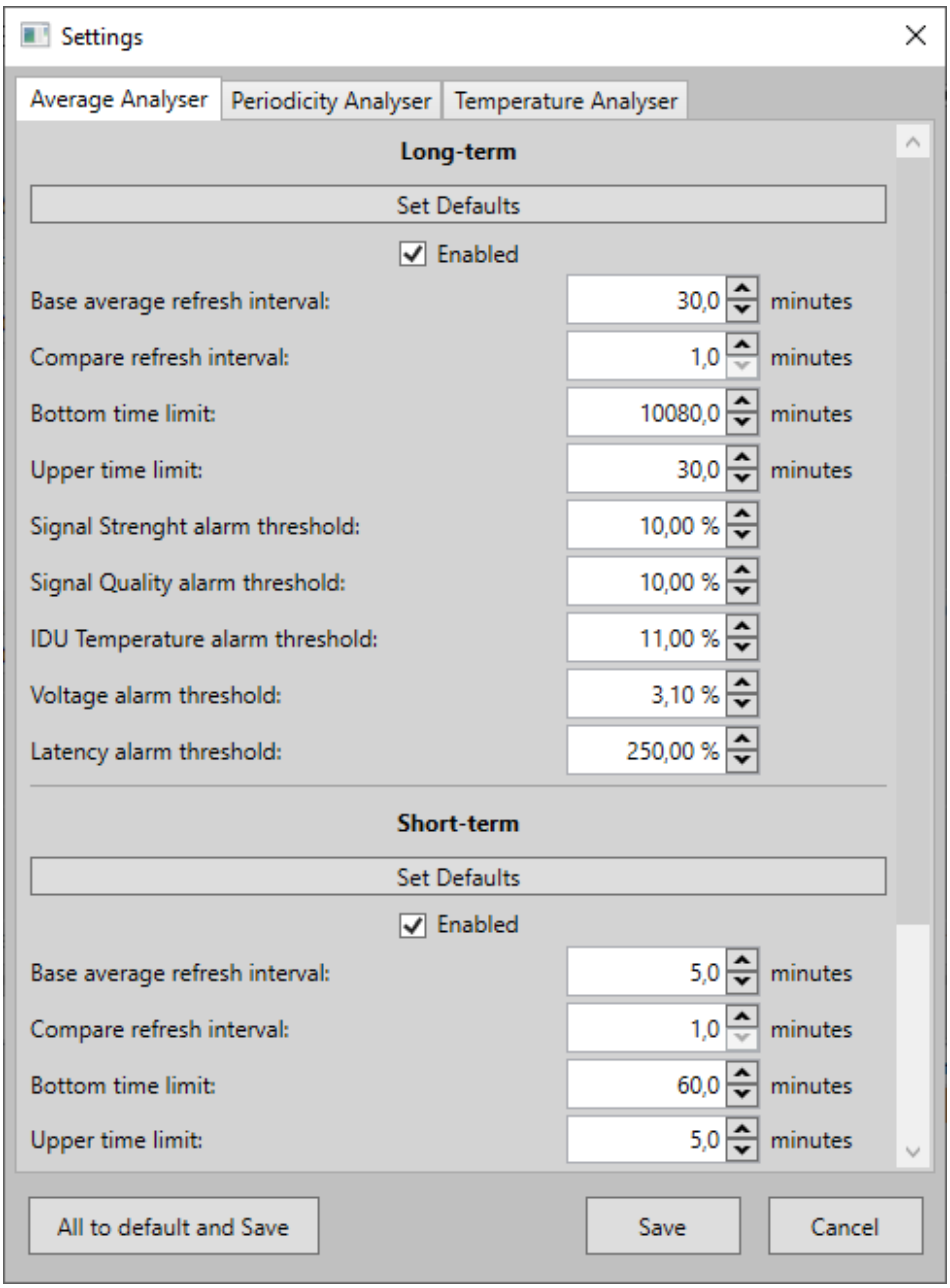
Seznam symbolů, veličin a zkratek

A	Amplituda
ACM	Adaptive Coding and Modulation
AM	Amplitudová modulace
AP	Access Poin
API	Application Programming Interface
ASK	Amplitudové klíčování
ASN.1	Abstract Syntax Notation One
Bd	Baud, jednotka modulační rychlosti
BER	Bit Error Ratio
BTS	Base Transceiver Station
BPSK	Binární klíčování fázovým posuvem
bit/s	Bit za sekundu, jednotka přenosové rychlosti
CLR	Common Language Runtime
CSMA/CA	Carrier-sense multiple access with collision avoidance
ČTÚ	Český Telekomunikační Úřad
D	Velikostní průměr
dB	Decibel, poměrová jednotka
DSSS	Direct Sequence Spread Spectrum
Eb/No	Energy per bit to noise power spectral density ratio
f	Frekvence
FHSS	Frequency Hopping Spread Spectrum
FM	Frekvenční klíčování
FM	Frekvenční modulace
φ	Úhel odchylky; fázový posuv
G	Výkonový zisk
Hz	Hertz, jednotka frekvence
ICMP	Internet Control Message Protocol
IDE	Vývojové prostředí
IDU	In-door Unit
IEEE	Institute of Electrical and Electronics Engineers
IPv6	Internet Protocol version 6
ISO/OSI	International Organization for Standardization/Open Systems Interconnection
IETF	Internet Engineering Task Force
TCP/IP	Transmission Control Protocol/Internet Protocol
ITU	International Telecommunication Union
λ	Délka vlny

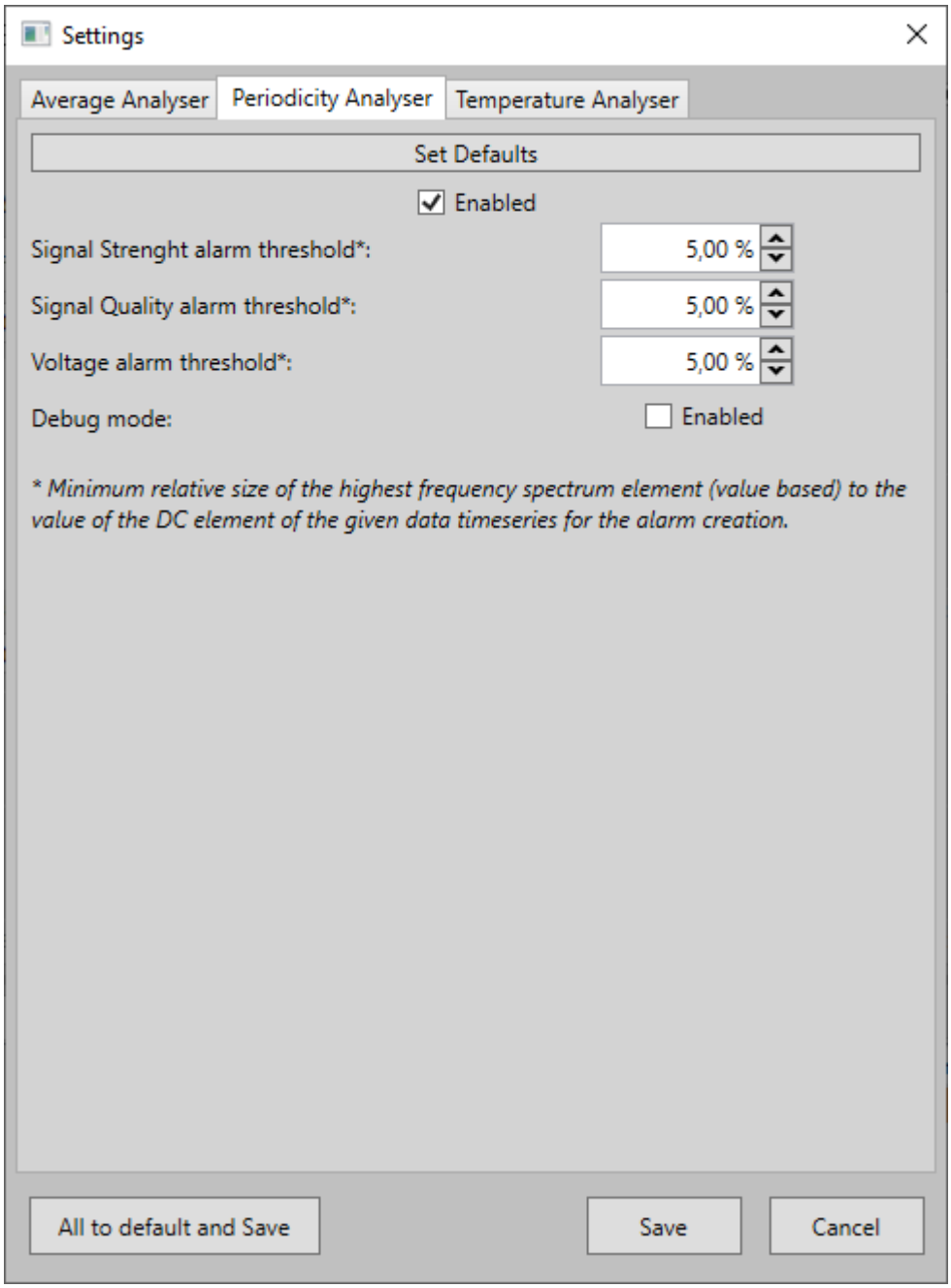
m	Metr, jednotka vzdálenosti
MIB	Management Information Base
MPLS	Multiprotocol Label Switching
NMS	Network Management Stations
ODU	Out-door Unit
OID	Object identifier
OFDM	Orthogonal Frequency Division Multiplexing
PDH	Plesiochronní digitální hierarchie
PDU	Protocol Data Unit
PM	Fázová modulace
PSK	Fázové klíčování
QAM	Kvadraturní amplitudová modulace
QoS	Quality of Service
QPSK	Kvadraturní klíčování fázovým posuvem
OSPF	Open Shortest Path First
RF	Radio frequency
RFC	Request for Comments
RIP	Routing Information Protocol
RSSI	Received Signal Strength Indication
SDH	Synchronní digitální hierarchie
SLA	Service Level Agreement
SNR	Signal-to-noise ratio
SNMP	Simple Network Management Protocol
SFP	Small form-factor pluggable transceiver
SMI	Structure of Management Information
SQL	Structured Query Language
t	Čas
T1	Transmission System 1
UDP	User Datagram Protocol
USM	User-Based Security Model
VACM	View-Based Access Control Model
VLAN	Virtual Local Area Network
W	Watt, jednotka výkonu
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Networks
WPF	Windows Presentation Foundation

Seznam příloh

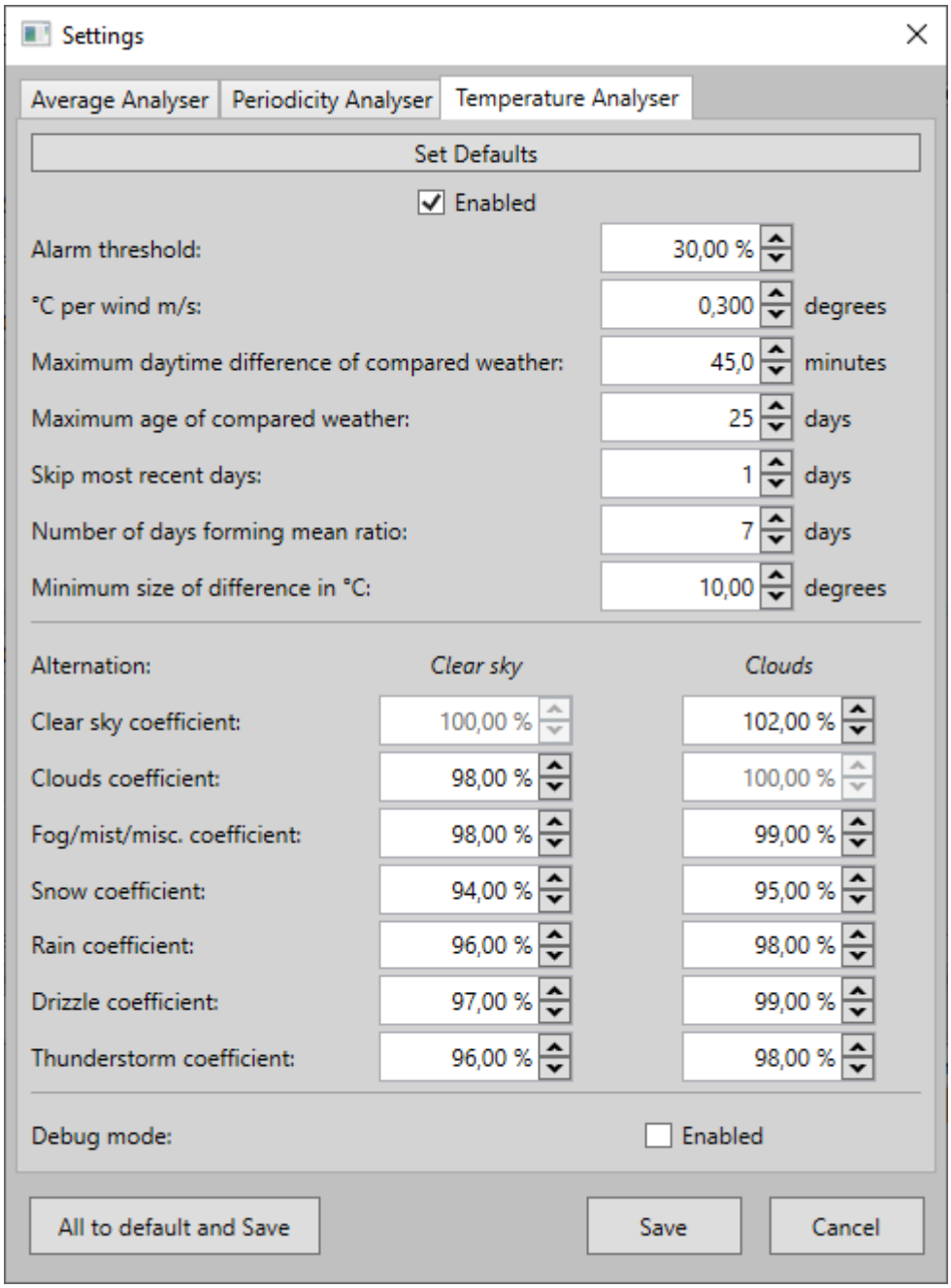
A Snímky nastavení aplikace



Obr. A.1: Snímek nastavení sledování odklonu od dlouhodobého průměru.



Obr. A.2: Snímek nastavení analyzátoru periodičností.



Obr. A.3: Snímek analyzátoru teplotní korelace.

B Vytvoření DB, RP a CQ v InfluxDB

Výpis B.1: Vytvoření retenčních politik v InfluxDB.

```
> create retention policy "week" on "MMDB"
  duration 1w replication 1 DEFAULT
> create retention policy "month" on "MMDB"
  duration 30d replication 1
> create retention policy "year" on "MMDB"
  duration 365d replication 1
```

Výpis B.2: Vytvoření kontinuálních dotazů pro měsíční retenční politiku v InfluxDB.

```
> create continuous query "cq_primary_30s" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."latency"
  from "latency" group by time(30s), "device" end
> create continuous query "cq_primary_30s_sig" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."signal"
  from "signal" group by time(30s), "device" end
> create continuous query "cq_primary_30s_sigQ" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."signalQ"
  from "signalQ" group by time(30s), "device" end
> create continuous query "cq_primary_10m_tempAir" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."tempAir"
  from "tempAir" group by time(10m), "device" end
> create continuous query "cq_primary_30s_tempIdu" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."tempIdu"
  from "tempIdu" group by time(30s), "device" end
> create continuous query "cq_primary_30s_tempOdu" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."tempOdu"
  from "tempOdu" group by time(30s), "device" end
> create continuous query "cq_primary_30s_voltage" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."voltage"
  from "voltage" group by time(30s), "device" end
> create continuous query "cq_primary_30s_tx" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."tx"
  from "tx" group by time(30s), "device" end
> create continuous query "cq_primary_30s_rx" on "MMDB"
  begin select mean("value") as "mean_value" into "month"."rx"
  from "rx" group by time(30s), "device" end
> create continuous query "cq_primary_10m_weather" on "MMDB"
  begin select median("condition")
  as "median_condition", mean("wind")
  as "mean_wind" into "month"."weather"
  from "weather" group by time(10m), "device" end
```

Výpis B.3: Vytvoření kontinuálních dotazů pro roční retenční politiku v InfluxDB.

```
> create continuous query "cq_secondary_1m" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."latency"
  from "latency" group by time(1m), "device" end
> create continuous query "cq_secondary_1m_sig" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."signal"
  from "signal" group by time(1m), "device" end
> create continuous query "cq_secondary_1m_sigQ" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."signalQ"
  from "signalQ" group by time(1m), "device" end
> create continuous query "cq_secondary_1h_tempAir" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."tempAir"
  from "tempAir" group by time(1h), "device" end
> create continuous query "cq_secondary_1m_tempIdu" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."tempIdu"
  from "tempIdu" group by time(1m), "device" end
> create continuous query "cq_secondary_1m_tempOdu" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."tempOdu"
  from "tempOdu" group by time(1m), "device" end
> create continuous query "cq_secondary_1m_voltage" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."voltage"
  from "voltage" group by time(1m), "device" end
> create continuous query "cq_secondary_1m_tx" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."tx"
  from "tx" group by time(1m), "device" end
> create continuous query "cq_secondary_1m_rx" on "MMDB"
  begin select mean("value") as "mean_value" into "year"."rx"
  from "rx" group by time(1m), "device" end
> create continuous query "cq_secondary_1h_weather" on "MMDB"
  begin select median("condition")
  as "median_condition", mean("wind")
  as "mean_wind" into "year"."weather"
  from "weather" group by time(1h), "device" end
```

Výpis B.4: Prvotní vytvoření administrátorského účtu, přihlášení, a vytvoření databáze InfluxDB.

```
> CREATE USER vut_user WITH PASSWORD 'vut_pass' WITH ALL PRIVILEGES

> auth
  username: vut_user password: vut_pass

> create user monitor with password 'monitor'
> create database MMDB
```

C Struktury databázových tabulek SQLite

Výpis C.1: Struktura tabulky `Device`.

```
CREATE TABLE "Device" (
  "Id"  integer NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
  "Address" varchar NOT NULL DEFAULT '0.0.0.0',
  "SnmpPort"  integer DEFAULT 161,
  "SnmpVersion" integer DEFAULT 1,
  "CommunityString" varchar DEFAULT 'public',
  "Latitude"  varchar DEFAULT 0,
  "Longitude" varchar DEFAULT 0,
  "IsPaused"  integer NOT NULL DEFAULT 1,
  "IsEnabledSignal" integer NOT NULL DEFAULT 0,
  "IsEnabledSignalQ"  integer NOT NULL DEFAULT 0,
  "IsEnabledTx" integer NOT NULL DEFAULT 0,
  "IsEnabledRx" integer NOT NULL DEFAULT 0,
  "IsEnabledTempOdu"  integer NOT NULL DEFAULT 0,
  "IsEnabledTempIdu"  integer NOT NULL DEFAULT 0,
  "IsEnabledVoltage"  integer NOT NULL DEFAULT 0,
  "SignalQDivisor"  integer DEFAULT 10,
  "IsTempIduOutdoor"  integer DEFAULT 0,
  "OidSignal_s" varchar,
  "OidSignalQ_s"  varchar,
  "OidTxDataRate_s" varchar,
  "OidRxDataRate_s" varchar,
  "OidTempOdu_s"  varchar,
  "OidTempIdu_s"  varchar,
  "OidVoltage_s"  varchar,
  "IsWatchedSignal" integer NOT NULL DEFAULT 0,
  "IsWatchedSignalQ"  integer NOT NULL DEFAULT 0,
  "IsWatchedTempOdu"  integer NOT NULL DEFAULT 0,
  "IsWatchedTempIdu"  integer NOT NULL DEFAULT 0,
  "IsWatchedVoltage"  integer NOT NULL DEFAULT 0,
  "IsWatchedPing" integer NOT NULL DEFAULT 0,
  "TresholdSignal"  integer NOT NULL,
  "TresholdSignalQ" integer NOT NULL,
  "TresholdTx"  integer NOT NULL,
  "TresholdRx"  integer NOT NULL,
  "TresholdTempOdu" integer NOT NULL,
  "TresholdTempIdu" integer NOT NULL,
  "TresholdVoltage" integer NOT NULL,
  "TresholdPing"  integer NOT NULL,
  "TreshUpSignal" REAL DEFAULT 0,
  "TreshUpSignalQ"  REAL DEFAULT 0,
  "TreshUpTx" REAL DEFAULT 0,
  "TreshUpRx" REAL DEFAULT 0,
  "TreshUpPing" REAL DEFAULT 0,
  "TreshUpTempOdu"  REAL DEFAULT 0,
  "TreshUpTempIdu"  REAL DEFAULT 0,
  "TreshUpVoltage"  REAL DEFAULT 0,
  "TreshDownSignal" REAL DEFAULT 0,
  "TreshDownSignalQ"  REAL DEFAULT 0,
  "TreshDownTx" REAL DEFAULT 0,
  "TreshDownRx" REAL DEFAULT 0,
  "TreshDownPing" REAL DEFAULT 0,
  "TreshDownTempOdu"  REAL DEFAULT 0,
  "TreshDownTempIdu"  REAL DEFAULT 0,
```

```
  "TreshDownVoltage"  REAL DEFAULT 0,
  "RefreshSignal" integer DEFAULT 1000,
  "RefreshSignalQ"  integer DEFAULT 1000,
  "RefreshTx" integer DEFAULT 1000,
  "RefreshRx" integer DEFAULT 1000,
  "RefreshTempOdu"  integer DEFAULT 1000,
  "RefreshTempIdu"  integer DEFAULT 1000,
  "RefreshVoltage"  integer DEFAULT 1000,
  "RefreshPing" integer DEFAULT 1000
);
```

Výpis C.2: Struktura tabulky `Link`.

```
CREATE TABLE "Link" (
  "Id"  integer NOT NULL PRIMARY KEY AUTOINCREMENT,
  "HopCount"  integer NOT NULL DEFAULT 0,
  "Name"  varchar NOT NULL,
  "Note"  varchar,
  "DeviceBaseId"  integer NOT NULL DEFAULT 0,
  "DeviceEndId" integer DEFAULT 0,
  "DeviceR1Id"  integer DEFAULT 0,
  "DeviceR2Id"  integer DEFAULT 0,
  "DeviceR3Id"  integer DEFAULT 0,
  "DeviceR4Id"  integer DEFAULT 0,
  FOREIGN KEY("DeviceBaseId") REFERENCES "Device"("Id")
);
```

Výpis C.3: Struktura tabulky `Alarm`.

```
CREATE TABLE "Alarm" (
  "Id"  integer NOT NULL PRIMARY KEY AUTOINCREMENT,
  "Rank"  integer,
  "IsActive"  integer,
  "IsAck" integer,
  "IsShowed"  integer,
  "GenerTime" bigint,
  "SettledTime" bigint,
  "LinkId"  integer,
  "DeviceId"  integer,
  "Measure" integer,
  "Type"  integer,
  "Trend" integer,
  "GenerValue"  float,
  "SettledValue"  float,
  "DeviceType"  varchar
);
```

D SNMP OID Summit

Ukázka prvků SNMP MIB verze: `1.2` [19].

OID	Název	Struktura	Datový typ
1.3.6.1.4.1.23688.1.1.1.0	name	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.2.0	condition	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.3.0	mannum	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.4.0	status	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.5.0	rssi	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.6.0	ebno	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.7.0	berr	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.8.0	temperature	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.9.0	maxENspeed	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.10.0	currentRadioSpeed	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.11.0	type	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.12.0	txFreq	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.13.0	rxFreq	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.14.0	txDataSpeed	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.15.0	rxDataSpeed	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.16.0	remoteName	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.17.0	remoteIPaddress	Skalár	IpAddress
1.3.6.1.4.1.23688.1.1.18.0	atpc	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.19.0	txPower	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.20.0	acm	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.21.0	txModulation	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.22.0	rxModulation	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.23.0	txBandwidth	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.24.0	rxBandwidth	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.25.0	licenseExpire	Skalár	Gauge

E SNMP OID Summit - modely BT

Ukázka prvků SNMP MIB verze: `2.7` – první část je shodná s předchozí verzí `1.2`, přibylo rozšíření níže [20].

OID	Název	Struktura	Datový typ
1.3.6.1.4.1.23688.1.1.26	txRadioDataSpeed	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.27	rxRadioDataSpeed	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.28	txBWlimitEth1	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.29	txBWlimitEth2	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.30	txBWlimitSFP2	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.31	txPortPriorityEth1	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.32	txPortPriorityEth2	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.33	portBackup	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.34	powerSupplySource	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.35	powerInputVoltage	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.36	maxPower	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.37	minPower	Skalár	INTEGER
1.3.6.1.4.1.23688.1.1.38	rBstatus	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.39	rBrssi	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.40	rBebno	Skalár	Gauge
1.3.6.1.4.1.23688.1.1.41	rBberr	Skalár	Counter
1.3.6.1.4.1.23688.1.1.42	rBtxFreq	Skalár	DisplayString
1.3.6.1.4.1.23688.1.1.43	rBrxFreq	Skalár	DisplayString