VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MONITOROVÁNÍ SYSTÉMOVÝCH PROSTŘEDKŮ POMOCÍ OPENHAB

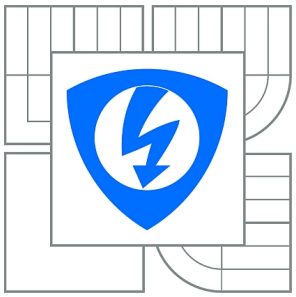
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MARTIN ŠTŮSEK
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

MONITOROVÁNÍ SYSTÉMOVÝCH PROSTŘEDKŮ POMOCÍ OPENHAB

MONITORING OF SYSTEM RESOURCES USING OPENHAB ENVIRONMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

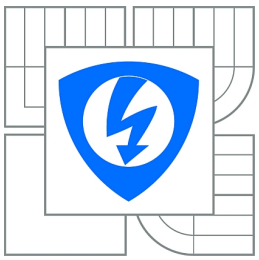
AUTOR PRÁCE — MARTIN ŠTŮSEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL MAŠEK

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Martin Štůsek	***ID:***	146975
Ročník:	3	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Monitorování systémových prostředků pomocí OpenHAB

POKYNY PRO VYPRACOVÁNÍ:

V rámci bakalářské práce se bude nutné seznámit se systémem OpenHAB (Open Home Automation Bus). Následně prostudovat detaily této platformy a teoreticky zpracovat dostupné bindings sloužící pro monitorování systémových prostředků (vytížení CPU, HDD, běžící procesy, …). Praktická část bakalářské práce bude spočívat ve vytvoření aplikace pro operační systémy Android/iOS, která bude umožňovat sledování využití systémových prostředků. Sledování systémových prostředků nebude omezené pouze na lokální stanici, ale bude možné monitorovat kterékoli vybrané stanice v síti. Sledované statistiky budou uživateli k dispozici lokálně i vzdáleně bez nutnosti připojení do stejné sítě jako jsou analyzované uzly.

DOPORUČENÁ LITERATURA:

[1] M2M communications: a systems approach. 1st ed. Editor David Boswarthick, Omar Elloumi, Olivier Hersent. Chichester: John Wiley, 2012, xxiii, 308 s. ISBN 978-1-119-99475-6.

[2] DONAHOO, Michael J a Kenneth L CALVERT. TCP/IP sockets in C: practical guide for programmers. 2nd ed. Boston: Morgan Kaufmann, c2009, xiii, 196 p. Morgan Kaufmann practical guides series. ISBN 01-237-4540-3.

Termín zadání:	10.2.2014	***Termín odevzdání:***	4.6.2014

Vedoucí práce: Ing. Pavel Mašek
Konzultanti bakalářské práce:

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

ABSTRAKT

Cílem této bakalářské práce je podrobné seznámení se s technologií OpenHAB a vytvoření funkční aplikace zobrazující využití systémových prostředků. Sledování systémových prostředků nebude realizováno pouze na jedné stanici, ale na všech instancích OpenHAB v lokální síti. Vytvořená aplikace umožňuje také vzdálený přístup ke sledovaným statistikám, bez nutnosti připojení do stejné sítě. Pro získávání systémových informací je využito balíčku systémových informací, který je postaven nad knihovnou SIGAR. Pro přenos mezi stanicemi a koncovým sběrným zařízením je využito vazby HTTP a REST API. První části bakalářské práce podrobně popisuje technologii OpenHAB, druhá část bakalářské práce popisuje použité technologie a vytvořené aplikace. V posledním bodě je uveden návod na zprovoznění OpenHAB na koncovém zařízení.

KLÍČOVÁ SLOVA

Google Charts, OpenHAB, RRD4J, SIGAR API, Sytémové informace, Vazby

ABSTRACT

The aim this bachelor thesis is detailed familiarity with the technology OpenHAB and create functional application that displays resource usage. Monitoring of system resources won't be implemented only in one station, but in all instances OpenHAB in a local network. Created application also allows remote access to reference statistics, without having to connect to the same network. For obtaining system information is used system information package, which is built over SIGAR library. To transfer between stations and end-collection device is used HTTP binding and REST API. The first part of the bachelor thesis describes in detail the technology OpenHAB, the second part describes the technology used and the created applications. In the last section gives guidance on commissioning of OpenHAB on the end device.

KEYWORDS

Bindings, Google Charts, OpenHAB, RRD4J, SIGAR API, System information

ŠTŮSEK, Martin *Monitorování systémových prostředků pomocí OpenHAB*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 70 s. Vedoucí práce byl Ing. Pavel Mašek,

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Monitorování systémových prostředků pomocí OpenHAB“ 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

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Pavlu Maškovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technická 12, CZ-61600 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

OpenHAB (Open Home Automation Bus) je poměrně nový projekt, jehož hlavním cílem je automatizace domácností. Jeho modularita mu však dovoluje použití v rozmanitých aplikacích. Jedná se o projekt s otevřeným zdrojovým kódem šířený pod licencí EPL (Eclipse Public License). OpenHAB funguje jako serverová aplikace, na počítači musí běžet tzv. runtime, ten je dostupný pro všechny nejrozšířenější operační systémy jako Windows, Linux a MacOS. OpenHAB může běžet i na méně výkonných zařízeních založených na ARM architektuře jako RasberryPI či BeagleBone Black. Pro ovládání serveru existují i klientské aplikace pro mobilní zařízení, pomocí kterých můžeme na dálku sledovat stav serveru. Tyto aplikace jsou dostupné pro většinu mobilních operačních systému. Aplikace pro Android a iOS jsou již dokončeny, vývoj pro Windows Phone je ve stádiu beta testování.

Funkčnost OpenHAB serveru může být libovolně rozšiřována, k tomu slouží tzv. vazby (bindings). Těchto balíčků je aktuálně dostupných více než 40 a přináší rozmanité funkce od komunikace s Asterisk ústřednami, až po po ovládání inteligentních žárovek a mnoho dalšího.

Cílem této bakalářské práce je sledování systémových prostředků na stanicích v lokální síti a jejich vizualizace na koncovém sběrném zařízení. Pro sledování systémových prostředků bude využito vazby systémových informací, pro přenos těchto informací mezi stanicí a koncovým sběrným zařízením bude využito vazby HTTP.

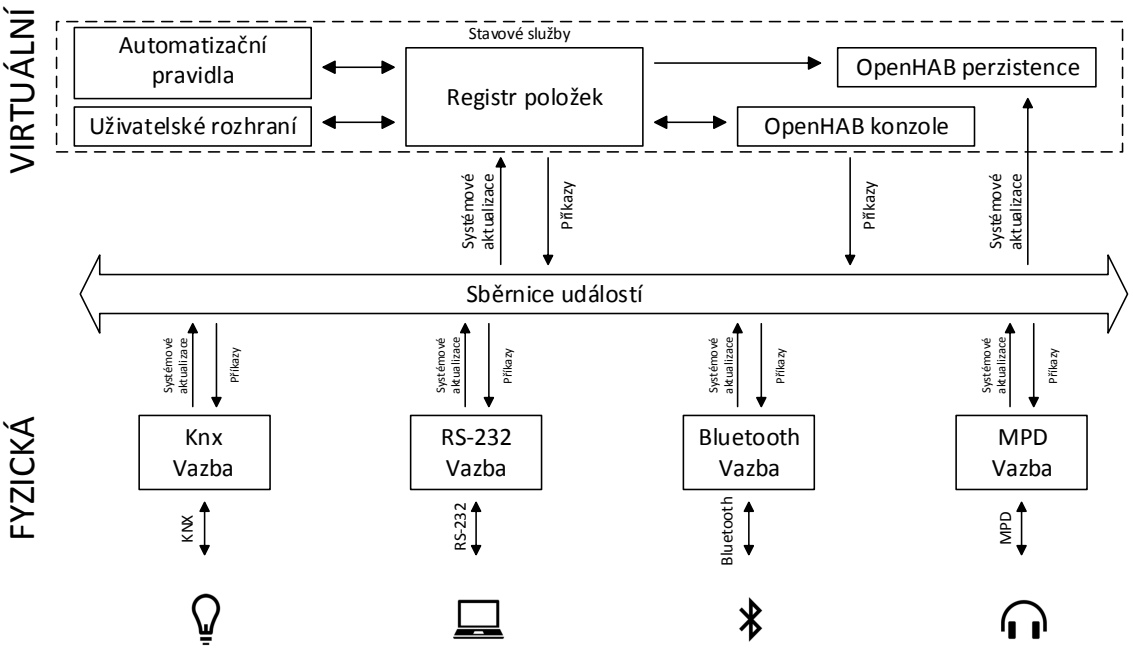
V teoretické části této bakalářské práce je popsána technoligie OpenHAB a její konfigurace , v praktické části jsou popsány vytvořené aplikace. Jsou vytvořeny aplikace pro nalezení všech distribucí OpenHAB v síti, pro vzdálený přístup, databáze pro uložení stavů položek a aplikace pro zobrazení grafů s využitím Google Charts.

1 OPENHAB

Projekt OpenHAB je založen na technologii Java, proto pro svůj běh vyžaduje přítomnost Java Runtime Enviroment. Z toho plynoucí výhody jsou nezávislost na platformně i architektuře procesoru.

1.1 Architektura

OpenHAB využívá OSGi (Open Services Gateway initiative) Framework, který umožňuje přidávání i odebírání modulů za běhu aplikace bez nutnosti zastavení služby. Blokové schéma aplikace OpenHAB můžeme vidět na Obr. 1.1, který vychází z [8].



Obr. 1.1: Blokové schéma architektury OpenHAB.

1.1.1 Sběrnice událostí

Hlavním prvkem architektury OpenHAB je sběrnice událostí (Event Bus), která umožňuje komunikaci mezi moduly. Všechny moduly mohou využít sběrnici k oznámení událostí ostatním a aktualizovat svůj stav na základě vnějších podnětů. Existují dva typy těchto událostí [8]:

- Příkazy, které vykonávají činnost nebo mění stav zařízení.
- Systémové aktualizace, které informují o změně stavu zařízení.

Všechny Bindings komunikují po sběrnici, z čehož plyne velmi nízká vazba mezi svazky, což umožňuje dynamickou povahu OpenHAB.

Ve většině případů bude na centrálním serveru běžet pouze jeden openhab-runtime, ale OSGi EventAdmin může být využit jako řídící služba, která umožní propojení více distribucí OpenHAB právě přes sběrnici událostí [8, 11].

1.1.2 Registr položek

Všechny funkce nelze realizovat bezestavovými prvky, proto je pro uložení aktuálního stavu všech prvků zaveden registr položek (Item Registr), který se stará o to, aby byly všechny tyto informace synchronizovány mezi všemi balíčky. Umožňuje také svůj aktuální obsah ukládat do databáze nebo na disk, aby tato data mohla být přístupná i po restartu systému [8].

1.1.3 Uživatelské rozhraní

Pro uživatelsky přívětivé zobrazení stavu všech prvků slouží uživatelské rozhraní, které svá data získává z registru položek. Uživatelské rozhraní neslouží pouze k zobrazení stavu položek, ale můžeme pomocí něj stavy položek měnit [8].

1.1.4 Automatizační pravidla

Pokud nechceme aktuální stavy položek jen zobrazovat a ručně na změny jejich stavů reagovat, ale chceme automaticky vykonávat příkazy, využijeme automatizační pravidla (Automation Rules). Můžeme si definovat rozmanitá pravidla pro reakci na změny stavů, všechna pravidla jen nadefinujeme v konfiguračním souboru [8].

1.1.5 OpenHAB konzole

Pro ovládání serveru je pro uživatele jistě přívětivější grafické rozhraní, OpenHAB ale umožňuje provádět odesílání příkazů i čtení aktuálních hodnot položek z konzolového okna [8].

1.1.6 OpenHAB Perzistence

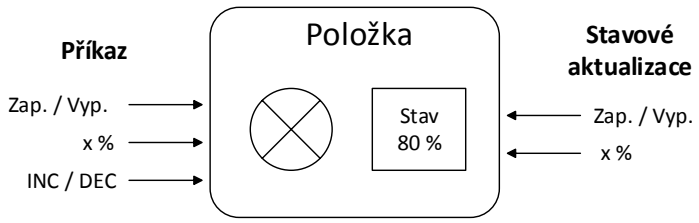
Pokud registr položek ukládá data do databáze o provedení se stará právě OpenHAB Perzistence. Jako výchozí databáze se využívá RRD4J (Round Robin Database for Java) což je databáze vytvořená v jazyce Java, která slouží i pro vykreslování grafů v uživatelském prostředí. Je podporováno více databází např. SQL (Structured Query Language) nebo DB4O (Database for Object) [8].

1.1.7 Vazby

Pro komunikaci hardware se sběrnicí slouží vazby (Bindings), které realizují spojení s daným typem hardware. Dle použitých vazeb se také zvolí vhodné protokoly pro přenos dat. Proto sběrnice může přistupovat ke všem prvkům stejným způsobem a nemusí řešit které protokoly využívat [8].

1.1.8 Koncepce přístupu k položkám

Každá položka obsahuje informace o stavu např. žárovka je zapnuta, světlo je ztlumeno na 50 %. Tyto stavy lze měnit pomocí příkazů např. zapni žárovku, ztlum světlo na 40 %. Všechny změny musíme také provést v systému, proto je vyslán příkaz pro aktualizaci stavu a položce je v registru nastavena nová hodnota. Vše je názorně zobrazeno na Obr. 1.2, který je převzat z [8]. Jsou zařízení, které obsahují více položek, např. rádio může obsahovat položku pro hlasitost, aktuální stanici. Cílem tohoto způsobu dělení je rozdělit složitější zařízení na jednoduché bloky a usnadnit zavádění nových zařízení [8].



Obr. 1.2: Koncepce přístupu k položce typu žárovka.

1.2 Zabezpečení

K zabezpečení komunikace využívá OpenHAB dva mechanismy. Prvním z nich je HTTPS (Hypertext Transfer Protocol Secure), který využívá pro zabezpečení komunikace protokol SSL (Secure Sockets Layer). OpenHAB podporuje HTTPS již v základu a pro přístup k serveru slouží port 8443.

Druhým z použitých mechanismů je autentizace. Všechny autentizační požadavky jsou odesílány na přístupový bod k JAAS (Java Authentication and Authorization Service), což je služba, která umožňuje vytvářet povolení a autentizovat uživatele. Ve výchozím nastavení je použit soubor `users.cfg`, kde můžeme vytvořit jednoduchý seznam uživatelů a jejich hesel se základními úlohami který poté slouží k autentizaci uživatelů. Pokud ale potřebujeme využít pokročilejší konfiguraci, musíme načíst konfigurační soubor pro JAAS LoginModules [11].

1.2.1 Nastavení zabezpečení

Pokud chceme nastavit zabezpečení, máme na výběr ze tří možností a to [11]:

- ON zabezpečení je povoleno globálně,
- OFF zabezpečení je globálně zakázáno,
- EXTERNAL všechny požadavky které nepřicházejí z interní sítě, musí být autorizovány.

1.3 Interakce s dalšími systémy

Pro usnadnění spolupráce s dalšími systémy obsahuje OpenHAB několik užitečných technologií, které ulehčují programátorům práci. Těchto technologií je např. využito k získávání/aktualizaci stavů položek, nebo k nalezení dalších distribucí OpenHAB v síti.

1.3.1 Rest API

Rest API (Representational State Transfer Application Programming Interface) slouží především k interakci OpenHAB s dalšími systémy. Umožňuje přístup k položkám, jejich stavům, stavovým aktualizacím, ale také odesílat příkazy. Rest API podporuje několik různých typů odpovědí, což znamená, že je možno zvolit, v jakém formátu přijde odpověď přidáním příslušného typu do hlavičky HTTP (Hypertext Transfer Protocol) např. XML (Extensible Markup Language), Json (JavaScript Object Notation). Přidáním příslušných parametrů do hlavičky můžeme získat jako odpověď soupis všech položek, sitemaps a widgets [11].

1.3.2 Server-Push

OpenHAB využívá pro server-push funkci Atmosphere Framework. Tato služba je využívána k automatickému obnovení hodnot uživatelského rozhraní při každé změně stavu položek.

Pro server-push službu můžeme použít long-polling, HTTP stream nebo web-socket. Abychom serveru sdělili, kterou možnost jsme zvolili, přidáme do HTTP hlavičky námi zvolenou možnost. Navíc je doporučeno zvolit pro každého klienta unikátní ID. Tato možnost přináší snížení využití sítě. Server automaticky porovnává, zda je aktuální zpráva stejná jako předchozí a pokud ano, zprávu neodesílá. Vlastní ID je také vyžadováno, pokud chceme ze serveru přijímat aktualizace ikon a štítků při spojení stream [11].

1.3.3 Service discovery

Tato služba byla vytvořena pro zjednodušení spolupráce mezi aplikacemi uživatelského rozhraní na různých platformách a OpenHAB. Tento modul využívá mDNS (multicast Domain Name System)/Bonjour. Tato knihovna je obsažena přímo v modulu, a proto není potřeba dalších zásahů, aby byl modul plně funkční. Služba je využita např. pro zjištění dostupných REST rozhraní na síti [11].

Při spuštění OpenHAB je pomocí technologie mDNS registrována na lokální síti každá distribuce OpenHAB. Služby jsou registrovány pro zabezpečený i nezabezpečený přenos.

Servery jsou registrovány pod názvem [11]:

- `_openhab-server._tcp.local.` pro http,
- `_openhab-server-ssl._tcp.local.` pro https.

Po úspěšném spuštění služby je získáno několik základních informací o serveru:

- IP adresu,
- TCP port,
- Uri atribut.

Multicast DNS

Multicast DNS (multicast Domain Name System) slouží k vyřešení záznamů služeb bez použití centrálního DNS serveru. Pakety mDNS jsou z 99 % kompatibilní s pakety klasického DNS. Služba využívá port 5353 a multicastovou IP adresu 224. 0. 0. 251.

Zbytečnému zahlcování sítě je předcházeno několika metodami např. využití TTL (Time To Live) nebo multicastovými odpověďmi, kdy ostatní klienti naslouchají a plní svou vyrovnávací paměť.

Multicast DNS využívá rezervovaný název `.local`, proto všechny dotazy s tímto názvem budou směrovány k mDNS na místo klasického DNS. Z toho plynou i možnosti nerozeznání názvu služby mDNS od klasického DNS a z tohoto důvodu odeslání dotazu na špatnou službu [19].

2 POUŽITÉ VAZBY

Server OpenHAB bez vazeb (bindings) funguje pouze jako zprostředkovatel služeb bez žádné větší funkcionality. Pokud chceme funkcionalitu serveru rozšířit, musíme doinstalovat vazby, které poskytují přídavné funkce pro rozmanité systémy.

Pro potřeby získávání systémových informací a komunikaci mezi servery je použita vazba systémových informací a vazba HTTP.

2.1 Vazba systémových informací

Abychom mohli získávat informace o sytému, musíme do OpenHAB nainstalovat vazbu systémových informací (System Info Binding). Instalace vazeb probíhá nakopírováním staženého balíčku do složky `/addons`, která je umístěna v kořenové složce OpenHAB serveru. Vazba systémových informací nám umožňuje monitorovat [11]:

- Systémovou paměť, procesor, dobu běhu, průměrné vytížení.
- Metriku síťového rozhraní.
- Metriku souborového systému.

2.1.1 Obecná konfigurace

V systému OpenHAB musí mít každá vazba nastaveny výchozí hodnoty např. čas obnovení, port pro komunikaci nebo v našem případě výchozí jednotky pro zobrazení velikost dat. Tato nastavení se provádí v souboru `openhab.cfg`. OpenHAB nabízí ve výchozím nastavení soubor `default_openhab.cfg`, ve kterém jsou připraveny ukázky nastavení pro všechny vazby a balíčky. Pro vazbu systémových informací jsou dostupná tato nastavení pro zobrazení jednotek [11]:

- **B** pro byty,
- **K** pro kilobyty
- **M** pro magabyty, jsou zvoleny jako výchozí,
- **T** pro terabyty.

Doba obnovení je ve výchozím nastavení na hodnotě 1000 ms [11].

2.1.2 Knihovna Hyperic SIGAR Native

Vazba systémových informací vyžaduje pro svůj běh Hyperic SIGAR Native (System Information Gatherer And Reporter) knihovnu, kterou musíme také nainstalovat obdobně jako vazby, zkopírováním balíčku do složky `/lib`.

Knihovna využívá SIGAR API, které poskytuje rozhraní pro získávání systémových informací. Jádro rozhraní je napsáno v jazyce C s vazbami implementovanými

do programovacích jazyků Java, Pearl, PHP, Python a C#. Knihovna je podporována velkým množstvím operačních systémů jako Windows, Linux, Mac OS, Solaris a FreeBSD. Knihovna je šířena pod licencí Apache 2.0 a může tak být získána ve formě zdrojového kódu nebo již zkompilována jako binární distribuci obsahující knihovny pro všechny podporované platformy [18].

2.1.3 Konfigurace vazby

Pokud chceme položku svázat se zařízením, musíme ji správně nakonfigurovat. To provedeme přidáním vazby do souboru konfiguračního souboru položek. Syntaxe je následující [11].

```
Typ proměnné Název proměnné = "<příkaz><doba obnovení>(<cíl>)"
```

Pole `<cíl>` nemusí být vždy vyplněno, je povinné pouze pro vazby vyžadující cíl pro svou funkčnost.

2.2 Vazba HTTP

Vazba HTTP není ve výchozí konfiguraci OpenHAB obsažena. Jako všechny vazby se stažený balíček nakopíruje do složky `/addons`. Poté je vazba HTTP připravena k použití a případné konfiguraci.

2.2.1 Konfigurace vazby

Vazba HTTP umožňuje odesílání i přijímání dat. Pro správnou funkci musíme provést konfiguraci vazby. Jednou z cest je provést konfiguraci v souboru položek, který je umístěn ve složce `/configurations/items`. Syntaxe pro vazbu HTTP může vypadat takto.

```
in:  http:"<[<adresa>:<doba obnovení>:<transformační pravidlo>]"
out: http:">[<příkaz>:<http metoda>:<adresa>]"
```

Pro odchozí komunikaci obsahuje vazba HTTP dva speciální příkazy:

- `*` slouží k volání adresy bez ohledu na vyslaný příkaz,
- `CHANGED` slouží k volání adresy při každé změně stavu položky [11].

2.2.2 Ukládání dat do mezipaměti

OpenHAB od verze 1. 3 podporuje také ukládání dat do mezipaměti. Tuto možnost můžeme využít např. při stažení dat, které obsahují informace pro více položek. Jednotlivé položky poté přistupují k informacím, aniž by musely data znovu stahovat.

Konfigurace pro ukládání dat do mezipaměti je uložena v souboru `openhab.cfg`. Konfigurace poté vypadá takto.

```
http:<název mezipaměti>.url= "adresa"
http:<název mezipaměti>.updateInterval= "doba obnovení"
```

Volání dat z mezipaměti v konfiguračním souboru položek poté může být provedeno následujícím způsobem [11].

```
Number Jméno {http="<mezipaměť:doba obnovení:transformace"}
```

3 KONFIGURACE OPENHAB RUNTIME

Konfigurace serveru je realizována čistě textovými soubory, proto je možné ji upravovat v jakémkoliv textovém editoru. Pro pokročilejší konfiguraci existuje nástroj OpenHAB designer, což je aplikace postavená nad vývojovým prostředím Eclipse [3].

V OpenHAB je konfigurace rozdělena do několika souborů, v nichž nastavujeme pravidla, položky i vzhled uživatelského rozhraní. Všechna nastavení jsou rozdělena do sedmi následujících skupin [11]:

- Položky (Items).
- Perzistence (Persistence).
- Pravidla (Rules).
- Skripty (Scripts).
- Mapa stránek (Sitemap).
- Akce (Actions).
- Transformace (Transform).

Všechny konfigurační soubory jsou uloženy ve složce `/configurations`, kde má každá skupina svoji podsložku pojmenovanou viz seznam výše.

Složka `configurations` obsahuje i další konfigurační soubory, `user.cfg` slouží pro nastavení uživatelského jména a jeho hesla, `openhab.cfg` slouží jako konfigurační soubor pro vazby a balíčky. Soubor `logback.xml`, který slouží pro automatický výpis stavu serveru do konzolového okna.

3.1 Položky

Konfigurace položek je umístěna ve složce `/configurations/items`. Všechny konfigurační soubory položek mají koncovku `.items`. Tyto soubory slouží k vytvoření položek a jejich propojení s vazbami. Položky mohou být několika typů, výčet aktuálně podporovaných je uveden v tabulce Tab. 3.1 níže, která je převzata z [11].

3.1.1 Syntaxe

Položky mají danou syntaxi, která je vidět níže. Pole ohraničena hranatými závorkami jsou volitelná [11].

```
Typ jméno ["štítek"] [<ikona>][(skupiny)][{konfigurace}]
```

- Typ - značí druh položky, které můžeme vidět v tabulce 3.1.
- Jméno - název proměnné, které využijeme pro volání položky v dalších konfiguračních souborech.

- Štítek - označuje text, který se u položky zobrazí v uživatelském prostředí. V této části můžeme přiřadit i jednotky viditelné v uživatelském rozhraní.
- Ikona - slouží pro přiřazení ikony viditelné v uživatelském rozhraní. Můžeme použít ikony obsažené v OpenHAB nebo použít své vlastní.
- Skupiny - každá položka může být členem několika skupin. Toho lze využít, abychom nemuseli do každé skupiny vytvářet novou položku. Místo toho ji připojíme do více skupin, ty od sebe oddělujeme čárkami.
- Konfigurace - v tomto poli nastavujeme vazby např. adresu hardware nebo dobu obnovení.

Tab. 3.1: Seznam podporovaných položek.

Název	**Popis**	**Příkazy**
Color	Informace o barvě v RGB soustavě.	OnOff, InDec, Percent
Contact	Stav dveřních okenních/kontaktů.	–
Date Time	Datum a čas.	–
Dimmer	Hodnota stmívače v procentech.	OnOff, IncDec, Percent
Group	Skupina položek.	–
Number	Číslo.	Decimal
Rolleshutter	Prvek pro žaluzie	UpDown, Stop, Percent
String	Textový řetězec.	String
Switch	Přepínač typicky pro světla.	OnOff

3.2 Perzistence

Pro ukládání dat slouží v OpenHAB tzv. perzistence, která nám umožňuje ukládání dat v průběhu času. OpenHAB není omezen pouze jediným druhem úložiště. V jeho konfiguraci může být nezávisle na sobě aktivních několik databází. Podporu databází musíme také instalovat stejně jako vazby. K tomu nám slouží balíčky zabalené v JAR (Java ARchive) souboru, které nakopírujeme do složky `/addons`. V OpenHAB jsou aktuálně dostupné tyto databáze [11]:

- **DB4O** (Database for Objects) odlehčená databáze využívající objekty jazyka Java,
- **RRD4J** (Round Rrobin Database for Java) verze round robin datbáze pro jazyk Java,
- **Open.Sen.Se** databáze umožňující zpracovávat a získávat data z mnoha různých zdrojů založená na platformě Internet of Things,
- **Logback** databáze která je využita při ukládání logů s velmi flexibilní syntaxi [11].

3.2.1 Konfigurace a syntaxe

Každá databáze vyžaduje pro svoji správnou funkčnost konfigurační soubor pojmenovaný podle typu databáze s koncovkou `.persist`, který je umístěn ve složce `/configuration/persistence`. V konfiguračním souboru musíme OpenHAB serveru definovat, které položky a kdy ukládat. K tomu slouží tzv. strategie (Strategies). Syntaxe pro konfiguraci perzistence vypadá následovně.

```
Strategies {
            <jméno strategie 1> : "<četnost ukládání 1>"
            <jméno strategie 2> : "<četnost ukládání 2>"
            ...
            default = < jméno strategie X>
           }
```

Pole `<četnost ukládání>` může být ve formátu přednastavených událostí např. každou hodinu, minutu nebo ve formě výrazu `"0 0 * * *"`, kde položky vyjadřují minuty, hodiny, den v měsíci, měsíc a den v týdnu v tomto pořadí.

Každé položce, jejíž stav chceme ukládat, musíme přiřadit strategii dle nasledující syntaxe.

```
Items {
        <seznam položek 1>: [strategy = <strategie 1>, ...]
        <seznam položek 2>: [strategy = <strategie X>, ...]
        ...
       }
```

V poli `<seznam položek>` můžeme zvolit strategii pro jedinou položku, skupinu položek nebo při použití znaménka `*` pro všechny položky sytému [11].

3.3 Pravidla

Soubory pro konfiguraci pravidel jsou umístěny ve složce `/configuration/rules`, s koncovkou `.rules`. Všechna pravidla obsažená v konfiguračním souboru mají společné souvislosti, tzn. že mezi si mezi sebou mohou vyměňovat proměnné nebo k nim navzájem přistupovat [11].

3.3.1 Syntaxe

Pravidla využívají programovací jazyk Java, s nadstavbou Xtend, která umožňuje upravit syntaxi jazyka, ale výsledný kód je stále kompatibilní s jazykem Java. Struktura pravidel se skládá ze tří částí [11]:

- import (Imports),
- deklarace proměnných (Variable Declarations),
- pravidla (Rules).

Import

Tato sekce obsahuje import použitých knihoven. Stejně jako v programovacím jazyce Java nemusíme znát celý název importovaného typu. Přilinkování příslušných knihoven je označeno klíčovým slovem `import`, za kterým následuje název importované knihovny [11].

Deklarace proměnných

V pravidlech můžeme využívat vlastních proměnných. Ty jež vyjadřují hodnotu, kterou nelze měnit, deklarujeme klíčovým slovem `val`. Hodnotu proměnné nastavujeme již při deklaraci. Proměnné, které svou hodnotu mění, např. počítadlo v cyklu, uvozujeme slovem `var` [11].

Pravidla

Vlastní pravidlo je uvozeno slovem `when` za kterým je uvedena podmínka po které následuje příkaz `then` a za ním vlastní tělo pravidla. Syntaxe pravidla, které po startu systému odešle e-mail je uvedena níže.

```
rule Start
when System started
then
    sendMail('xyz@mail.com', "OpenHAB", " OpenHAB is running.")
end
```

V těle můžeme podmínky rozvětvovat pomocí funkcí jako `if`, `switch` nebo jiných podobně jako v jazyce Java. Konec pravidla je uvozen slovem `end`. V OpenHAB dělíme pravidla do tří základních skupin, podle toho kdy jsou vykonávány nebo na co reagují [11]:

- Pravidla závislá na položkách reagují na příkazy nebo změnu stavu položek.
- Časově závislá pravidla jsou vykonávány v určený čas, můžeme využít přednastavených časů, nebo použít cron, pomocí kterého si můžeme časy nastavit libovolně. OpenHAB využívá balíčku Quartz což je plánovač napsaný v jazyce Java, který nám umožňuje specifičtější nastavení spuštění oproti standartní službě cron, např. si můžeme určit den i hodinu, kdy se bude událost vykonávat [20].
- Poslední jsou pravidla závislá na stavu systému, která se vykonávají při změně stavu systému např. při startu nebo vypnutí.

3.4 Skripty

Stejně jako pravidla využívají skripty jako základ prostředí Java s nadstavbou Xtend. Syntaxe je proto velmi podobná jazyku Java, ale s mnoha vylepšeními umožňujícími psát stručný kód.

Konfigurační soubory skriptů mají koncovku `.scripts` a jsou umístěny ve složce `/configuration/scripts`. Skripty mají přístup ke všem deklarovaným položkám a mohou využívat všech jejich příkazů např. `ON`, `OFF`. Dále mohou vyvolávat všechny standardní akce podporované aplikací OpenHAB.

Pro vykonání skriptu jej musíme vyvolat. Skript je vždy definován jménem souboru a tímto jménem je také volán.

```
callscript(jméno skriptu)
```

Každý skript vrací hodnotu posledního výrazu, ta může být i nulová. Skripty můžeme volat z různých míst, jako jsou pravidla, kalendář Google nebo z konzole XMPP (Extensible Messaging and Presence Protocol) [11].

3.5 Mapa stránek

Grafické rozhraní potřebuje informace které prvky a kde zobrazit. Tyto informace jsou obsaženy v souboru s koncovkou `.sitemap`, který je umístěn ve složce konfiguračních souborů `/configuration/sitemap`.

Pro tvorbu rozhraní obsahuje OpenHAB několik grafických prvků. Každý prvek vyžaduje alespoň jeden povinný parametr, ten se liší u každého prvku. Většina prvků vyžaduje parametr `item`, kterým definujeme jakou položku má prvek zobrazovat. Další položky jsou nepovinné jako např. ikona nebo štítek. Speciální prvky vyžadují také webovou adresu pro zobrazení stránky či obrázku anebo také dobu obnovení. Ve verzi 1. 5. 0. obsahuje OpenHAB položky uvedené v Tab. 3.2 , která čerpá z [11].

Tab. 3.2: Seznam grafických prvků zobrazitelných v OpenHAB.

Název prvku	**Popis**
Colorpicker	Slouží pro výběr barvy např. pro RGB diodu.
Chart	Zobrazí graf z dat uložených v rrd4j databázi.
Frame	Rozdělí prvky uživatelského rozhraní do rámečků.
Group	Zobrazí všechny položky náležící do skupiny.
Image	Zobrazí obrázek.
List	Zobrazí stav položek.
Switch	Slouží pro zobrazení přepínače.
Selection	Zobrazí menu pro výběr.
Setpoint	Zobrazí tlačítka plus a mínus pro nastavení hodnot.
Slider	Zobrazí posuvník.
Text	Zobrazí text a stav položek nebo vstup do další úrovně.
Video	Zobrazí video.
Webview	Zobrazí internetovou stránku.

3.5.1 Dynamická mapa stránek

V průběhu bakalářské práce byl učiněn přechod na OpenHAB 1. 5. 0, který umožnil využití dynamické mapy stránek. Uživatelské rozhraní může díky dynamické mapě stránek např. měnit barvu položek na základě jejich stavů. U všech položek grafického rozhraní můžeme měnit tyto vlastnosti [11]:

- viditelnost,
- barvu štítku,
- barvu hodnoty.

Viditelnost

Ve výchozí konfiguraci jsou prvky viditelné. Pokud chceme prvky skrývat využijeme vlastnost `visibility`. Pro konfiguraci pravidel vlastnosti `visibility` můžeme využít operátory `==, >=, <=, !=, >, <`. Pravidla můžeme libovolně skládat za sebe, pokud alespoň jedno pravidlo vrátí hodnotu `true` bude prvek viditelný [11].

Barvy

Díky vlastnosti `labelcolor` a `valuecolor` můžeme měnit barvu štítků respektive hodnot položek. Jako v případě viditelnosti můžeme pravidla skládat za sebe a využít několika operátorů. OpenHAB obsahuje základních 17 barev založených na CSS definici. Můžeme ale použít jakoukoliv barvu ve formátu `#xxxxxx` [11].

3.5.2 Syntaxe

Každý konfigurační soubor musí být uvozen slovem `sitemap`, za kterým následuje název mapy stránek. Samotné nastavení položek je uzavřeno do bloku složenými závorkami. Zobrazení rámečku s elementem text a obrázkem vypadá následovně [11].

```
sitemap název_mapy
   {
     Frame [label=<štítek>] [icon=<ikona>] [item=<položka>]
      {
        Text item=<položka> [label=<štítek>] [icon=<ikona>]
        Image url=<url> [label=<štítek>] [refresh=xxxx]
       }
    }
```

3.6 Akce

Akce jsou předdefinované metody napsané v jazyce Java, které jsou staticky importovány (vše probíhá automaticky). Akce jsou využity v pravidlech a skriptech pro provádění specifických operací. Akce jsou rozděleny do dvou skupin [11]:

- Akce jádra.
- Akce doplňků.

3.6.1 Akce jádra

Tuto skupinu můžeme ještě rozdělit do několika bloků. Prvním z nich jsou akce spojené s událostní sběrnicí. Sem patří akce jako `sendCommand (položka, příkaz)`, která odešle příkaz položce. Akce `postUpdate (položka, stav)` odešle položce aktualizaci stavu. Dále je zde akce `Map`, která uloží aktuální stav položek a akce `restoreStates` díky níž můžeme stavy obnovit.

Další skupina akcí souvisí se zvukem. Akce nám dovolují ovládat hlasitost systému, přehrávat zvuk ze souboru či internetové adresy. OpenHAB je také vybaven modulem pro převod textu na řeč.

Systém umožňuje i logování na několika úrovních na informační, výstražné, chybové a ladicí. OpenHAB obsahuje i další akce jako vytvoření časovače, zaslání HTTP požadavku atd. Všechny akce jsou dostupné na stránkách projektu [11].

3.6.2 Akce doplňků

Každá vazba obsahuje také vlastní akce, ty se liší podle druhu zaměření každé vazby. Pro příklad doplněk NotifyMyAndroid Action odešle zprávu definovanému android zařízení. Doplněk Mail odešle e-mail přes SMTP (Simple Mail Transfer Protocol) server, který jsme si nakonfigurovali v souboru `openhab.cfg`. Zavedena je i podpora odesílání příloh [11].

3.7 Transformace

Pokud potřebujeme získávat informace z webového rozhraní a dále je interpretovat ve srozumitelné formě, musíme provést transformaci. Tato metoda je vhodná například pro zobrazení předpovědi počasí, teploty, vlhkosti a dalších položek podporovaných rozhraním. Konfigurační soubor pro transformace je umístěn ve složce `/configuration/transform` a pro transformaci mohou být využity XSL (eXtensible Stylesheet Language) nebo JSON (JavaScript Object Notation) soubory. Pro získání aktuální teploty z `http://weather.yahooapis.com`, můžeme použít XSL transformaci jejíž kód je dostupný zde [12].

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0"
  version="1.0">
<xsl:output indent="yes" method="xml" encoding="UTF-8"
 omit-xml-declaration="yes" />
<xsl:template match="/">
<xsl:value-of select="//item/yweather:condition/@temp"/>
</xsl:template>
</xsl:stylesheet>
```

Pokud chceme položce nastavit hodnotu získanou z transformace, provedeme to v konfiguračním souboru položek. Hodnota, na kterou se nastaví položka, je v XSL souboru uvozena slovem `select` [11].

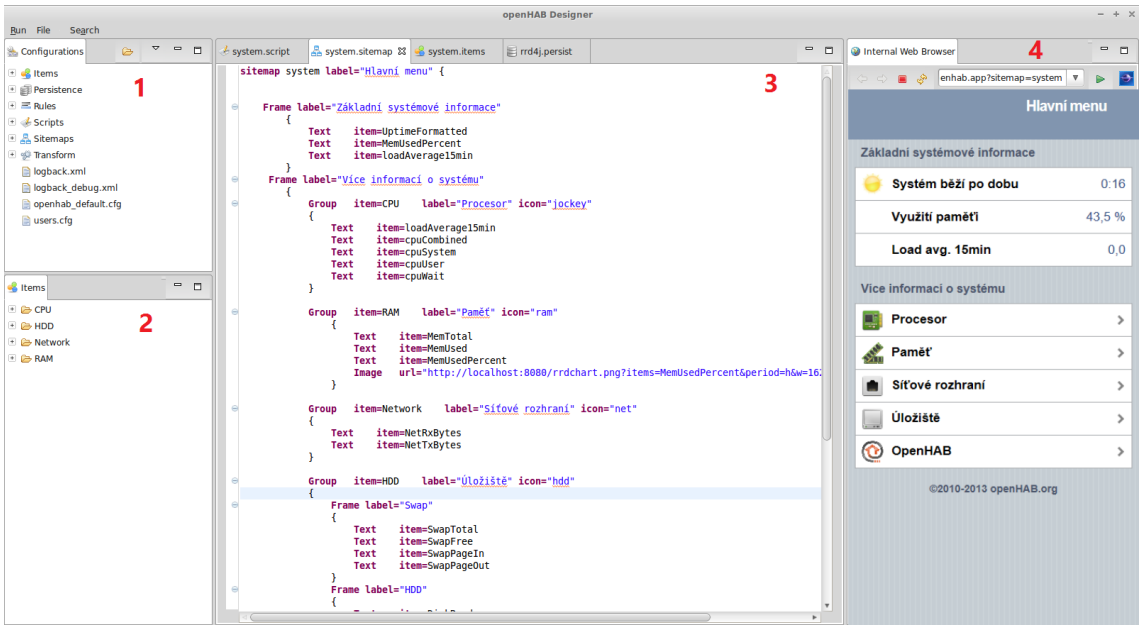
3.8 Nástroje pro změnu konfigurace

Pro změnu konfigurace můžeme využít jakéhokoliv textového editoru, ty syntaxy Openhab konfigurace neznají, a proto není syntaxe žádným způsobem zvýrazněna. Tento problém částečně řeší balíčky pro editory Notepad++, Vim a Midnight Commander, které přináší kontrolu syntaxe do těchto editorů. Nejsnažší cesta pro úpravu konfigurace, je však využití nástroje OpenHAB Designer.

3.8.1 OpenHAB Designer

Jedná se o program využívající Eclipse RCP (Rich Client Platform) vytvořený pro potřeby OpenHAB, který má svou vlastní syntaxi, kterou tato aplikace zná a přehledně jí zabarví pro zvýšení přehlednosti, jak je patrné na Obr. 3.1. Aplikace obsahuje i našeptávač známý z jiných vývojových prostředí. Po zmáčknutí kombinace kláves `Ctrl + mezerník` je vyvolána nabídka se všemi dostupnými akcemi. Vývojové prostředí je rozděleno do čtyř sekcí jak můžeme vidět na Obr. 3.1 [11]:

1. Konfigurace (Configurations).
2. Položky (Items).
3. Pracovní okno.
4. Webový prohlížeč (Internal Web Browser).



Obr. 3.1: Okno OpenHAB designer.

4 VYTVOŘENÉ APLIKACE

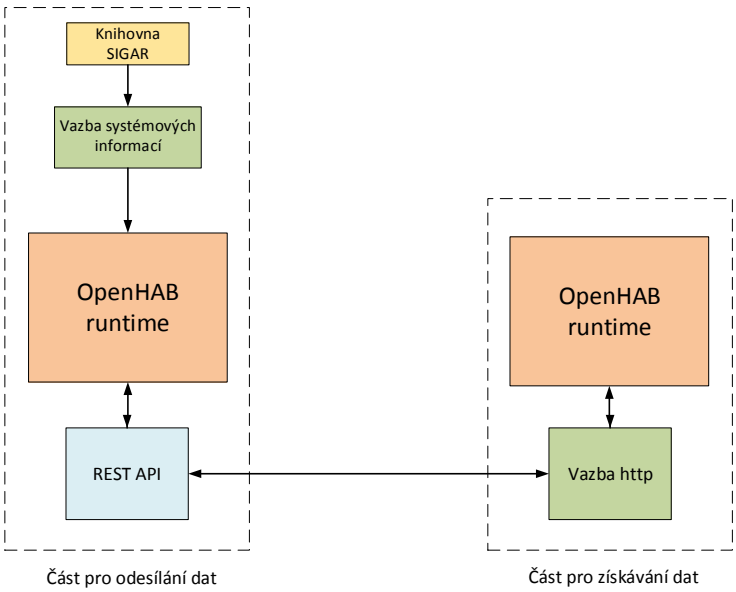
Byly napsány aplikace pro nalezení všech dostupných OpenHAB serverů v síti a dále byla vytvořena databáze pro ukládání stavů položek. Všechny aplikace jsou napsány v jazyce Java z důvodu co největší nezávislosti na použitém operačním systému.

4.1 Sledování více instancí OpenHAB

V základní konfiguraci OpenHAB neumožňuje sledování několika instancí v lokální síti. OpenHAB však obsahuje několik technologií sloužících pro interakci s dalšími systémy. Těchto technologií můžeme využít pro získávání dat z jednotlivých instancí OpenHAB.

4.1.1 Metoda získávání dat

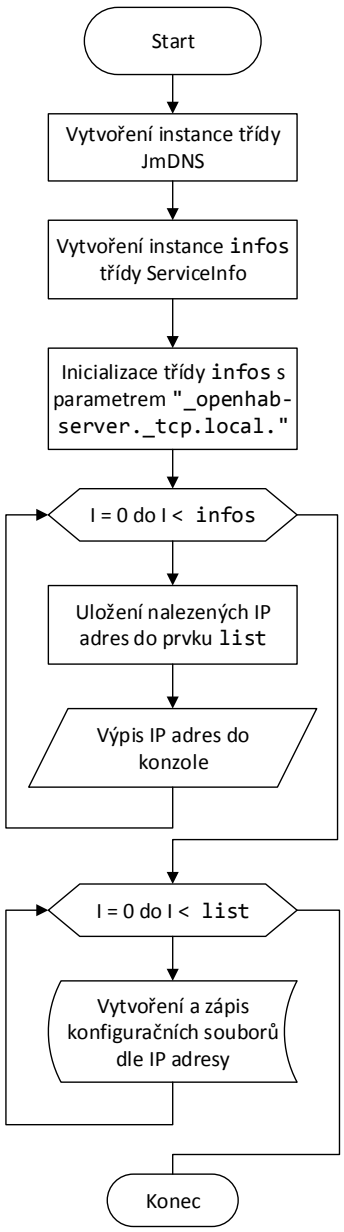
Hlavní myšlenkou sledování více instancí je využití REST API a také vazby HTTP. Rozhraní REST nám umožňuje skrze webovou adresu přistupovat přímo ke stavu položek. Toho je využito u vazby HTTP, která získává data pomocí metody `GET` skrze rozhraní REST. Získaná data z instancí OpenHAB na lokální síti jsou ukládána přímo do položek na serveru starajícím se o sběr dat. Schéma získávání dat je na Obr. 4.1.



Obr. 4.1: Schéma získávání dat z více instancí OpenHAB.

4.1.2 Nalezení všech instancí OpenHAB

Před spuštěním samotného OpenHAB serveru je spuštěn vytvořený program, který získá IP adresy všech dostupných instancí OpenHAB na síti. Pro získání IP adres je využito technologie Service Discovery, která využívá JmDNS (Java multicast Domain Name System) knihovnu. Vývojový diagram programu je zobrazen na Obr. 4.2.



Obr. 4.2: Vývojový diagram programu.

Program po spuštění nalezne všechny služby na lokální síti registrované pod jménem `_openhab-server._tcp.local`. Z těchto služeb získá IP adresy jednotlivých OpenHAB instancí. Z těchto získaných IP adres je vytvořena konfigurace OpenHAB serveru, který slouží k interpretaci získaných dat. Jsou vytvořeny konfigurační soubory obsahující všechny položky nalezených instancí OpenHAB.

Po zápisu konfiguračních souborů se program ukončí. Dále pokračuje spouštění OpenHAB serveru s právě vytvořenou konfigurací.

4.2 Vzdálený přístup k OpenHAB

Abychom mohli z Internetu přistupovat do naší lokální sítě, musíme znát její veřejnou IP adresu. Vyjádření IP adresy ve čtyřech oktetech ale není pro člověka tak dobře zapamatovatelné jako jméno domény. Proto je pro přístup do lokální sítě využito DDNS (Dynamic Domain Name System). Díky němuž přistupujeme do sítě skrze webovou adresu a nemusíme zadávat IP adresu, což je pro běžného uživatele mnohem příjemnější. Poskytovatelů těchto služeb je na Internetu velké množství. Liší se především v množství doplňkových služeb a cenových ohodnoceních. V této bakalářské práci je využito slyužby DNSdynamic, která je plně bezplatná [2].

4.2.1 Dynamické DNS

Základem klasických DNS (Domain Name System) byly statické databáze. Jakákolivk změna se prováděla editací Master souboru pro každou zónu. Pomocí jistých kódů je možné DNS zprávou přidávat či odebírat záznamy ze zvolených zón. Tuto skutečnost využívá dynamické DNS.

Doménová jména identifikují uzly uvnitř stromové struktury jmen. Každý uzel obsahuje sadu záznamů o prostředcích tzv. RRs (Resource Records). Všechny záznamy o prostředcích se stejným jménem, třídou a typem jsou sady záznamů o prostředcích tzv. RRset (Resource Record Set).

Dynamické DNS využívá pro změnu záznamu v databázi tzv. `UPDATE` zprávy. Ty se značí hodnotou `5` v poli `Opcode` DNS zprávy. Sekce `Update` DNS zprávy obsahuje záznamy, které mají být přidány nebo smazány. Existují čtyři výrazy, které mohou být ve zprávě obsaženy [21]:

- přidat RRs do RRset,
- smazat RRset,
- smazat všechny RRset ze jména,
- smazat RRs z RRset.

4.2.2 Služba DNSdynamic

Tato služba poskytující dynamické DNS je dostupná na `www.dnsdynamic.org`. Po úspěšné registraci nám umožňuje registraci neomezeného množství domén. K doméně je připojena naše veřejná IP adresa. Pokud jsme v síti se směrovačem s NAT (Network Address Translation), musíme na směrovači nastavit předávání portů. V tomto bodě je již funkční přístup k OpenHAB serveru z Internetu.

Jelikož internetový poskytovatel může naši veřejnou IP adresu měnit, nemusí být záznam na `www.dnsdynamic.org` vždy aktuální. Proto je poskytován klient pro automatické obnovováni DNS záznamů. Služba také obsahuje web API(Application Programming Interface), které umožňuje aktualizaci DNS záznamů pomocí automatických skriptů, nebo přímo přes URL (Uniform Resource Locator) [2].

4.3 Databáze pro OpenHAB

Z důvodu využití externího řešení pro grafické zobrazení dat uložených v databázi (Goole Charts), vznikl požadavek na databázi. Ta musí umožňovat přístup k uloženým datům skrze rozhraní vhodné pro webové aplikace.

4.3.1 Vybraná databáze

Jako výchozí databáze slouží v OpenHAB databáze RRD4J, kterou je využivá v balíček RRD4J perzistence. Balíček ale neumožňuje přímý přístup k datům k databázi, ty jsou pouze vykresleny v plošném grafu. Sama o sobě má databáze mnoho předností, mezi hlavní patří její fixní velikost.

Databáze je již od vytvoření naplněna nulovými daty, a proto její velikost již s přibývajícími daty neroste. RRD4J je dále schopna pracovat s několika konsolidačními funkcemi např. `AVERAGE, MAX, MIN, LAST`. Databáze umožňuje také vysokou kompresi dat, při zachování rychlého přístupu k datům. Proto byl v bakalářské práci upraven balíček RRD4J perzistence, do kterého byla přidána podpora přístupu k datům skrze webové rozhraní [14].

4.3.2 Struktura databáze

Databáze pracuje s pevným množstvím dat a ukazatelů na každý prvek. Data jsou zapisována na buňku v databázi, na kterou odkazuje ukazatel. Po zapsání dat se ukazatel posune na další buňku.

Můžeme si představit, že se pohybujeme po kružnici, která nemá konec ani začátek. Data jsou postupně ukládána do každé buňky, ale současně jsou nejstarší záznamy mazány jak můžeme vidět na Obr. 4.3. Tím je dosaženo fixní velikosti výsledného souboru.

Databáze nám umožňuje ukládat pouze číselné hodnoty, ale nemusí to být pouze celá čísla. Časové značky jsou ukládány ve formě počtu sekund od 1. ledna 1970 [14].

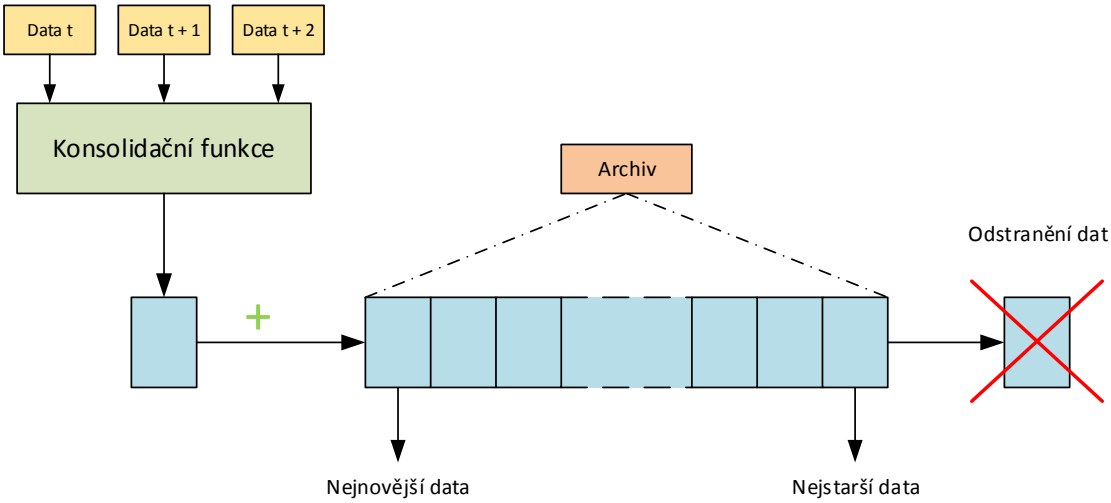
Archivy

RRD4J nám umožňuje vytvářet tzv. archivy. Ty jsou využívány pro uložení dat s různými periodami a rozlišením. Každý archiv je definován [14]:

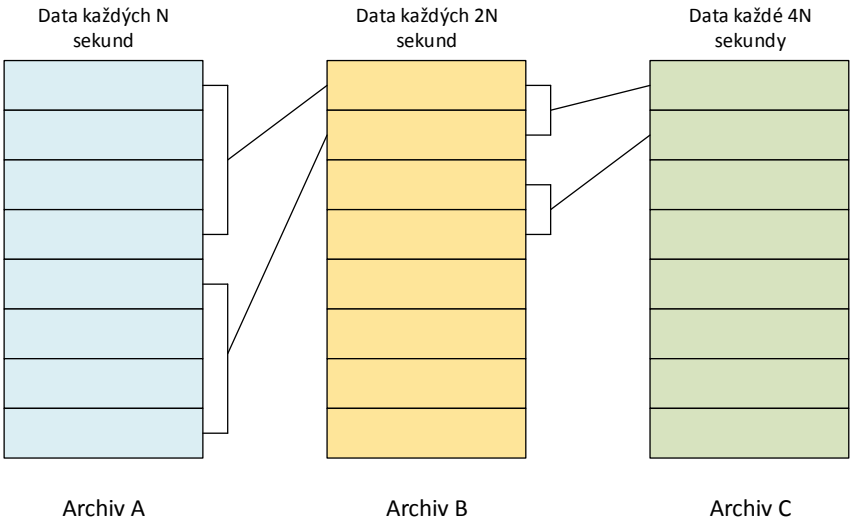
- velikostí kroku,
- konsolidační funkcí,
- počtem kroků,
- počtem řádků v databázi.

Konsolidační funkce určuje jaké hodnoty budou do archivu ukládány. Podporované funkce jsou průměr, maximální, minimální a poslední hodnoty. Velikost kroku určuje po kolika sekundách se budou data ukládat do databáze. Počet kroků určuje po kolika krocích se data přestanou ukládat do jednoho řádku dle konsolidační funkce a ukazatel se přesune na další řádek [14].

Počet archivů v jedné databázi není omezen. Pokud databáze obsahuje více archivů s různými velikostmi kroků. Jsou data přesouvána z archivu s menším krokem do archivu s větším krokem viz Obr. 4.4 [4].



Obr. 4.3: Princip ukládání dat v RRD4J.



Obr. 4.4: Princip archivů v RRD4J.

4.3.3 Vytvořená databáze

Pro potřeby zobrazení dat v rozumných intervalech bylo vytvořeno několik archivů. V databázi jsou obsaženy čtyři archivy s těmito patrametry:

- doba uložení **5** minut s rozlišením **1s**,
- doba uložení **1** hodina s rozlišením **10s** a konsolidační funkcí průměr,
- doba uložení **1** týden s rozlišením **30m** a konsolidační funkcí průměr,
- doba uložení **1** měsíc s rozlišením **2h** a konsolidační funkcí průměr,

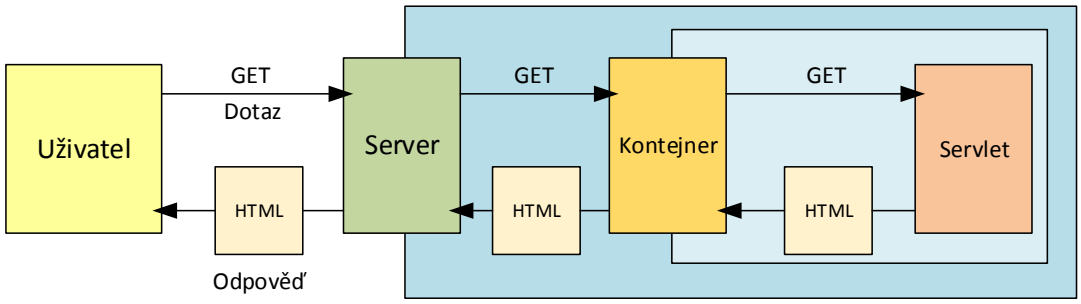
Dále byl pro databázi vytvořen servlet, pomocí kterého můžeme k datům v databázi přistupovat skrze protokol HTTP.

4.3.4 Servlet

Vytvořená data báze obsahuje servlet, který zasílá data ve formě JSON (JavaScript Object Notation). Takto získaná data jsou lehce zpracovatelná pro webový prohlížeč s podporou JavaScript.

Struktura servletu

Servlety zajišťují nízko úrovňovou komunikaci založenou na protokolech typu dotaz-odpověď. Uživatelské požadavky přicházejí k webserveru, který je předá do kontejneru a ten jej předá servletu. Servlet požadavek zpracuje a vygeneruje stránku, kterou v odpovědi odešle zpět ke klientovi. Celý proces je znázorněn na Obr. 4.5 [9].



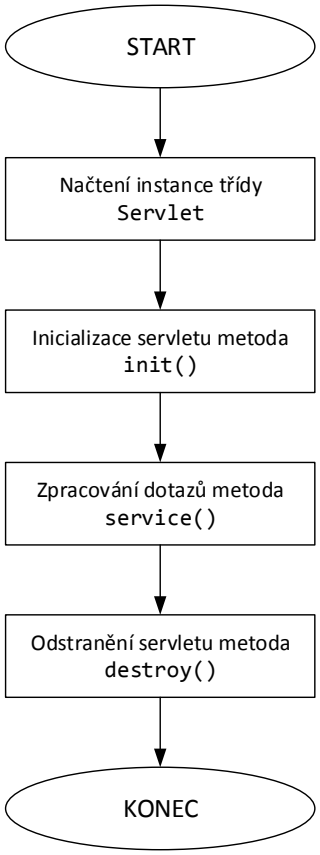
Obr. 4.5: Struktura servletu.

Životnost servletu

O celý životní cyklus servletu od načtení třídy, až po její zničení se stará kontejner. Celý cyklus se skládá ze těchto kroků:

- načtení třídy servletu,
- vytvoření instance servletu,
- volání metody `init()`,
- volání metody `service()`,
- zničení servletu `destroy()`.

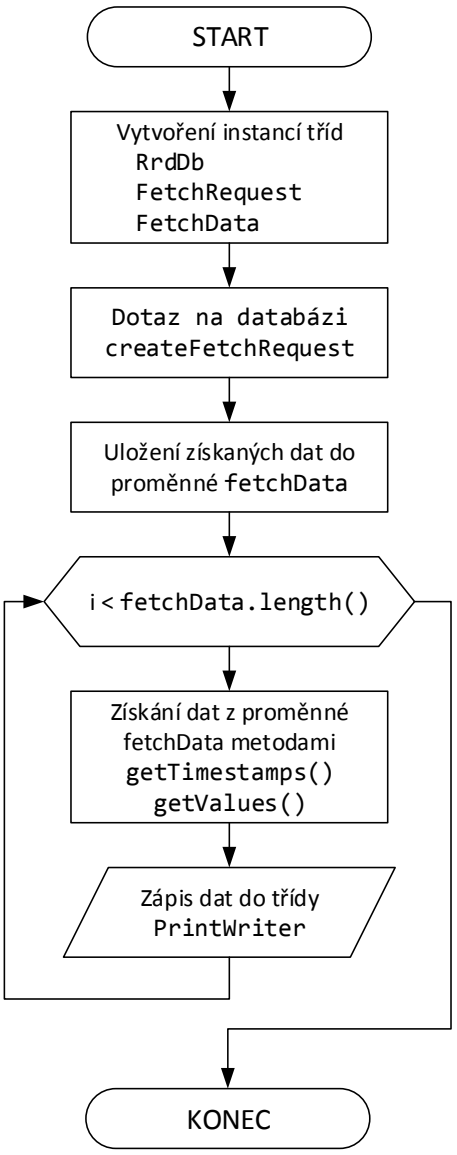
Metoda `init()` se volá po startu servletu, díky tomu můžeme např. načítat data z databáze před uvedením do provozu. Metoda `service()` je voláná při každém dotazu na server. Metoda `service()` vytvoří nové vlákno, které se stará o příchozí dotazy skrze metody `doGet()` (metoda `GET`) apod. Metoda `destroy()` je volána při ukončení či restartu aplikace. Celý cyklus je zobrazen na Obr. 4.6 [9], [5].



Obr. 4.6: Životnost servletu.

4.3.5 Získávání dat z databáze

Získávání dat z databáze probíhá v několika krocích jak můžeme vidět na Obr. 4.7. Nejprve musíme vytvořit instanci třídy `RrdDb`, které jako parametr předáme cestu k souboru. Poté musíme vytvořit instanci třídy `FetchRequest`, pomocí které tvoříme dotazy na databázi. Dále je třeba vytvořit instanci třídy `FetchData` do které jsou následně uložena data získaná z dotazu. Získaná data jsou ve formátu, se kterým pracuje databáze, abychom mohli data dále zpracovávat, musíme je převést. K tomu je využito metody `getTimestamps`, která vrací časy záznamů do pole datového typu `Long` ve formě sekund od roku 1970. Pro získání hodnot je využito metody `getValues()`, který vrací hodnoty do pole typu `Double` [7].



Obr. 4.7: Vývojový diagram získávání dat.

4.3.6 Struktura získaných dat

Pro vytvoření odpovědi na dotaz je využito instance třídy `PrintWriter`. Ta zapíše získaná data a odešle je ve formě odpovědi. Odeslaná odpověď je odeslána jako objekt JSON.

Formát odpovědi

Servlet s odpověďmi obsahující data ve formě JSON je dotupný na adrese.

```
http://IP_adresa:port/data.json
```

Data odesílaná v odpovědi mají tento formát.

```
[
   {"time" : "časová_značka",
    "value" : "získaná_hodnota"}
]
```

Konfigurovatelné parametry odpovědi

Parametry dat v odpovědi můžeme nastavovat pomocí parametrů webové adresy v dotazu na servlet. Seznam dostupných parametrů je uveden v Tab. 4.1.

Tab. 4.1: Dotupné parametry webového rozhraní.

Parametr	Popis	Formát hodnot	Výchozí hodnota
item	Udává název položky.	–	–
period	Perioda vrácených dat.	5m, h, d, m, last	5m
start	Začátek vrácených dat.	yyyy,M,d,H,m	–
stop	Konec vrácených dat.	yyyy,M,d,H,m	–
step	Krok vrácených dat.	–	–

Příklad použití parametrů

Adresa pro webový dotaz k nastavení parametrů vrácených hodnot může vypadat následovně.

```
http://localhost:8080/data.json?item=XYZ&period=h&step=10
```

Při zadání této adresy nám budou vráceny data ve formátu JSON, pro položku XYZ. Vrácená data budou za poslední hodinu a vrácena bude každá desátá hodnota.

5 ZOBRAZENÍ DAT

Jelikož serverová aplikace OpenHAB neobsahuje uživatelské rozhraní, je pro běžného uživatele velmi obtížné získávat informace o stavu položek či odeslaných příkazech. OpenHAB proto umožňuje přistupovat k serveru skrze webové rozhraní, které je již uživatelsky příjemnější.

5.1 Uživatelská rozhraní

OpenHAB obsahuje několik uživatelských rozhraní, takže nejsme odkázáni pouze na jedno rozhraní, které se nemusí líbit každému. Díky REST API je navíc možno získávat hotnoty stavů položek a zobrazovat je ve svém vlastním uživatelském rozhraní, díky tomu jsou možnosti zobrazení dat téměř neomezené. Od verze OpenHAB 1. 4. 0 je možno využívat tři uživatelská rozhraní [11]:

- Classic UI (User Interface),
- Comet Visu.
- GreenT UI.

Na mobilních operačních sytémech nejsme omezeni použitím pouze těchto uživatelských rozhraních, ale můžeme využít aplikace přímo určené pro OpenHAB. Například v systému Android je to aplikace HABDroid [11].

5.1.1 Classic UI

Toto uživatelské rozhraní využívá OpenHAB jako výchozí a je obsaženo již v základu. Je přístupné na této webové adrese.

```
protokol://adresa serveru:port/openhab.app?sitemap=mapa stránky
```

Po grafické stránce vychází uživatelské rozhraní z operačního systému iOS od firmy Apple jak je vidět na Obr. 5.1. Toto uživatelské rozhraní pro získávání systémových informací nevyužívá REST API, ale přistupuje ke stavům položek přímo. Rozhraní využívá technologii HTML (Hyper Text Markup Language) a Java Script s frameworkem WebApp (Web Application Micro Framework), proto můžeme používat rozhraní na rozličných zařízeních či prohlížečích využívající jádro WebKit [11].

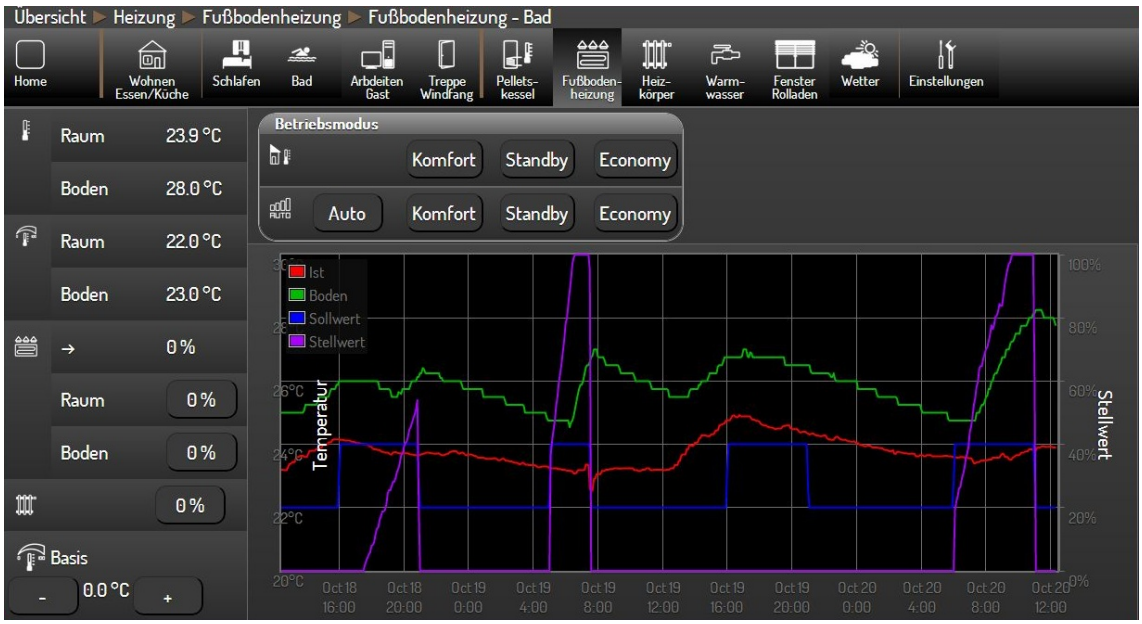
Obr. 5.1: Uživatelské rozhraní ClassicUI.

5.1.2 Comet Visu

Comet Visu je nezávislé na konfiguračních souborech mapy stránek pro OpenHAB, proto musíme vytvořit další konfigurační soubor pro rozhraní Comet Visu.

Pro správnou funkčnost Comet Visu musíme do složky `/addons` umístit soubor `org.openhab.io.cv*.jar` a umístit složku s Comet Visu do adrsáře `/webapps`.

Konfigurační soubor pro Comet Visu je ve formě XML, tento kód je poté zpracován a vykreslen ve webovém prohlížeči. Již v základu obsahuje Comet Visu několik témat, příklad tématu `Metal` je na Obr. 5.2. Comet Visu je ale také velmi pžizpůsobitelné, vzhled můžeme upravit pomocí CSS (Cascading Style Sheets) [11].



Obr. 5.2: Uživatelské rozhraní Comet Visu.

5.1.3 GreenT UI

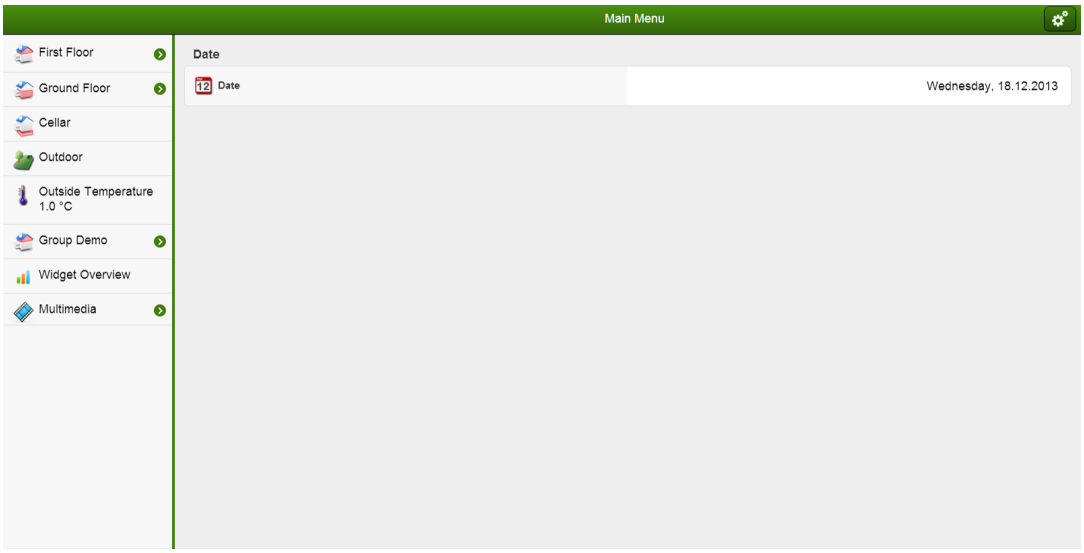
Uživatelské rozhraní GreenT využívá technologii HTML a Java Script se Sencha 2. 0 frameworkem. Rozhraní můžeme využívat na rozličných zařízeních i webových prohlížečích založených na jádře WebKit.

Ve výchozí instalaci OpenHAB není toto uživatelské rozhraní obsaženo, proto jej musíme doinstalovat. To provedeme zkopírováním staženého balíčku do adresáře `/webapps`, k rozhraní potom přistupujeme skrze webovou adresu.

```
protokol://adresa serveru:port/greent
```

Mapu stránek si volíme až při startu webového rozhraní, nebo si jednu můžeme zvolit jako výchozí.

Uživatelské rozhraní obsahuje nastavení pro zobrazení na různých zařízeních. Na výběr je ze vzhledů přizpůsobených pro mobilní telefony, tablety a osobní počítače. Zobrazení na počítači můžeme vidět na Obr. 5.3. GreenT nám také umožňuje změnit si barvy nebo jazyk uživatelského rozhraní [11].



Obr. 5.3: Uživatelské rozhraní GreenT.

5.1.4 Aplikace HABDroid

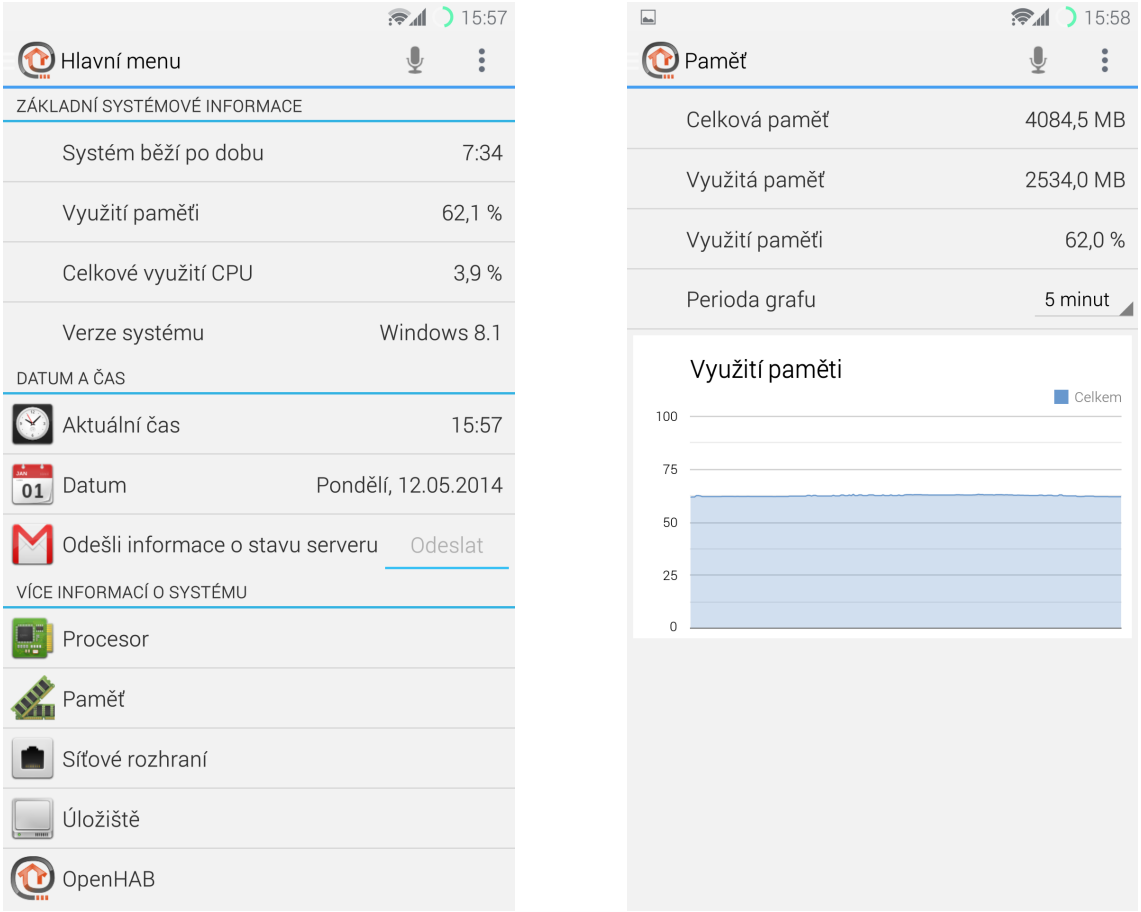
Pro vzdálené ovládání OpenHAB serveru můžeme využít aplikaci HABDroid, která nám umožňuje zobrazení uživatelského rozhraní na chytrém mobilním telefonu či tabletu se systémem Android. Pro správnou funkci vyžaduje HABDroid systém Android ve verzi nejméně 4. 0. 3. Aplikace využívá tzv. HOLO vzhled a to v bílé nebo

černé barvě. Po úspěšném přihlášení k serveru je zobrazeno uživatelské rozhraní, které můžeme vidět na Obr. 5.4, kde můžeme vidět informace o stavu serveru nebo ovládat položky jako na počítači.

Aplikace HABDroid využívá technologii mDNS pro získání informací o běžících serverech v lokální síti. Díky této technologii by si aplikace měla veškeré informace pro připojení získat sama. Funkčnost této technologie ale není zajištěna v každé síti a proto si můžeme adresu serveru nastavit sami.

Díky aplikaci HABDroid můžeme adresu a nastavení serveru získat pomocí NFC (Near Field Communication) štítku, nebo také takovýto štítek vytvořit. Pomocí něj pak můžeme ovládat spínače, žaluzie a další podporované prvky pouhým přiložení zařízení k štítku.

HABDroid podporuje také dynamickou mapu stránek, takže se viditelnost či barva prvků může měnit dle stavu položek [11].



Obr. 5.4: Zobrazení uživatelského rozhraní v aplikaci HABDroid.

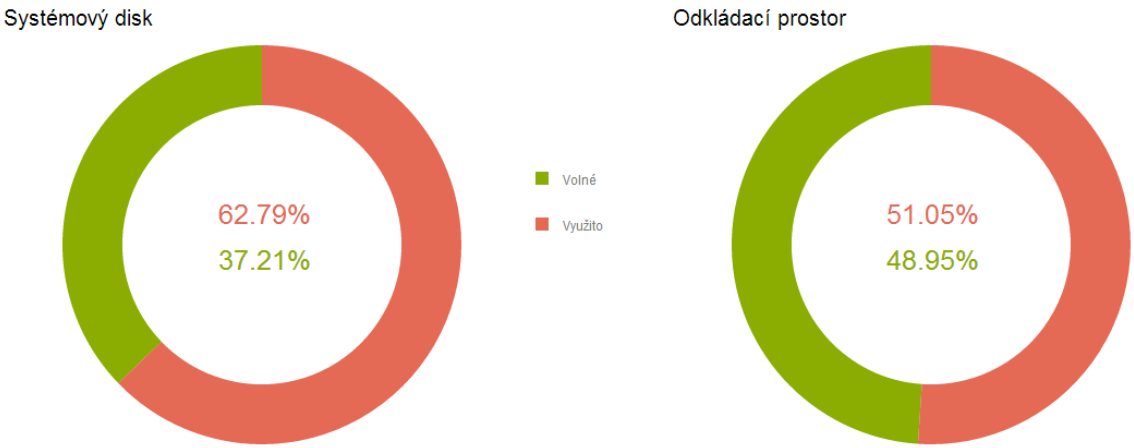
5.2 Grafické zobrazení dat

Pokud chceme v uživatelském prostředí OpenHAB serveru zobrazit grafy, tak máme možnost využít prvku `Chart`, který obsahuje mapa stránek. V uživatelském prostředí je poté tento graf vykreslen jako bitmapový obrázek. Jeho rozlišení je velmi nízké a nedokáže se přizpůsobit různým velikostem zobrazovacích zařízení, prvek `Chart` dále umožňuje zobrazení pouze plošných grafů. Proto je v této bakalářské práci využito Google Charts.

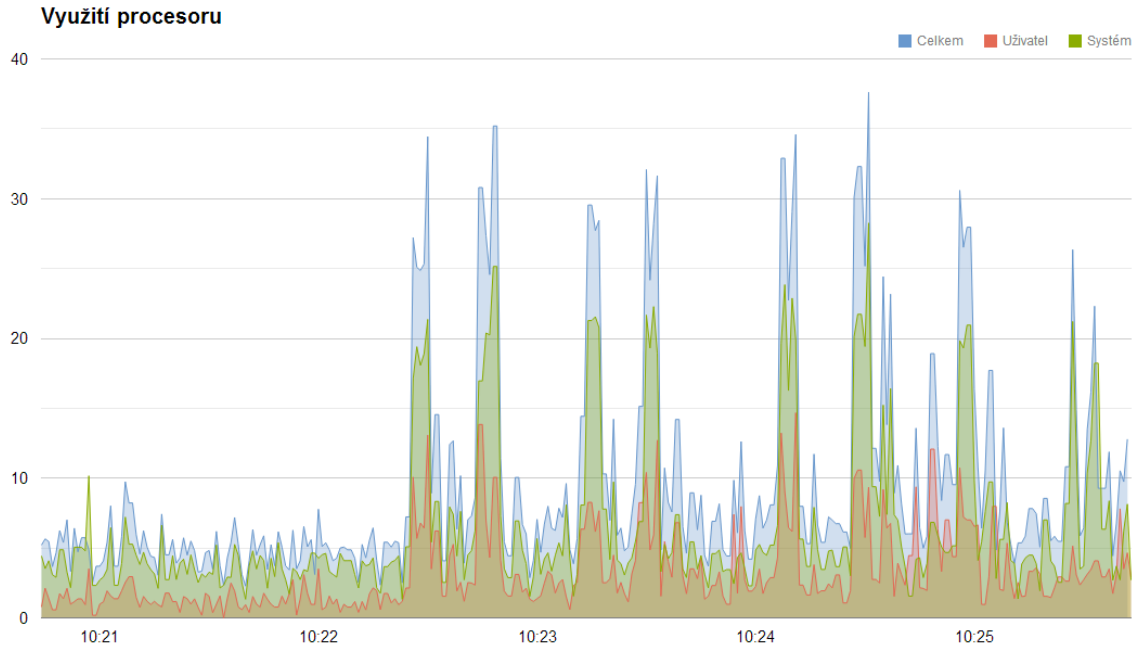
5.2.1 Google Charts

Umožňují zobrazení mnoha typů grafů jako sloupcový, koláčový, plošný atd. Velkou výhodou je také implementace prostřednictvím JavaScriptových knihoven. Jednoduchá je také změna grafu na jiný typ. Toho je dosaženo tak, že všechna data se nejprve vloží do datové tabulky. Poté je vytvořen objekt typu graf z dat obsažených v tabulce. Graf je poté vykreslen v elementu `<div>`, podle `id` které jsme přiřadili objektu graf.

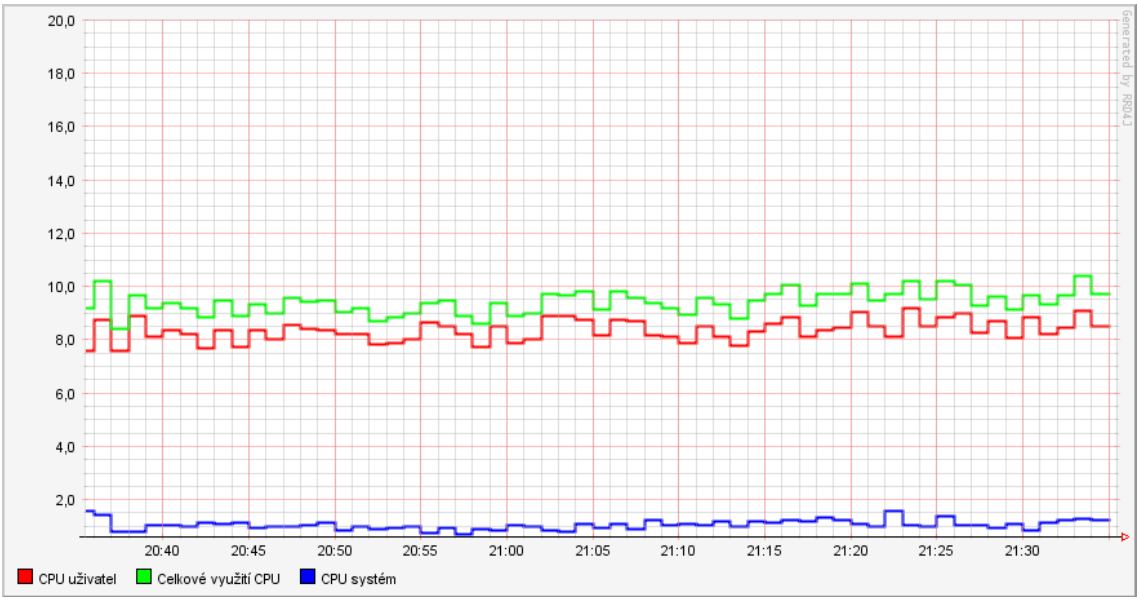
Velmi rozsáhlé jsou také možnosti vzhledu grafů jako změna barvy os, maximální hodnoty na osách, barva grafu a mnohé další. Grafy jsou vykreslovány ve formátu SVG (Scalable Vector Graphics). Podporované jsou také starší verze Internet Explorer, díky implementaci VML (Vector Markup Language). Na Obr. 5.5 a 5.6 jsou uvedeny příklady grafu vykresleného pomocí Google Charts a na Obr. 5.7 je graf vykreslený prvkem `Chart` [6].



Obr. 5.5: Zobrazení koláčového grafu.



Obr. 5.6: Zobrazení plošného grafu.



Obr. 5.7: Graf vykreslený rozhraním OpenHAB.

5.2.2 Metoda získávání dat

Pro získávání dat je využito JavaScriptové knihovny jQuery a AJAX (Asynchronous JavaScript and XML) dotazů. Data jsou získávána z vytvořené databáze pomocí webového rozhraní, které vrací data ve formě JSON. Abychom získali data, vyšleme požadavek s metodou `GET`, získaná data dále zpracováváme. Hodnota stavu položky je v elementu `value`, časová značka je v elementu `time`. Musíme projít celý získaný objekt a hodnoty uložit do datové tabulky. Příklad možného odeslání požadavku `GET` je uveden níže.

```
request = $.ajax({type : "GET", dataType : "jsonp", url : url})$
request.done(function(data)
     {
       for (i = 0; i < data.length; i++)
           value[i] = Number (data[i].value);
            time[i] = (new Date (Number (data[i].time) * 1000));
      });
```

5.3 Ovládání aplikace

Pro aplikaci bylo zvoleno uživatelské rozhraní GreenT, kvůli jeho lepšímu využití plochy na monitoru větší úhlopříčky.

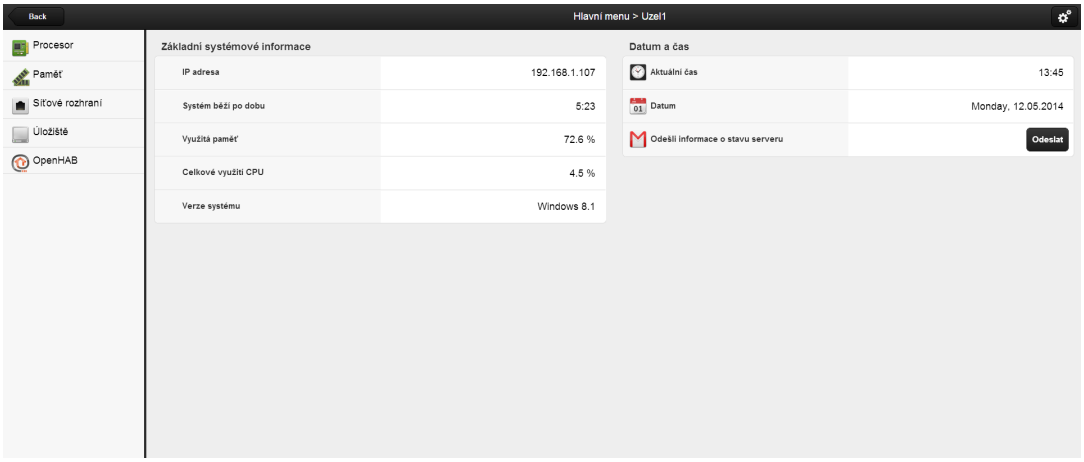
5.3.1 Hlavní menu

Po přihlášení k serveru se dostáváme do nabídky se všemi nalezenými instancemi OpenHAB. Tyto odkazy slouží ke vstupu do hlavního menu jednotlivých stanic. Hlavní menu obsahuje odkazy pro přístup do dalších úrovní aplikace jak můžeme vidět na Obr. 5.8. Menu s odkazy do dalších úrovní je rozděleno do pěti položek:

- **Procesor**,
- **Paměť**,
- **Síťové rozhraní**,
- **Úložiště**,
- **OpenHAB**.

Vpravo se nalézá výpis základních informací o stavu serveru, který je rozdělen do dvou kategorií.

- **Základní systémové informace**, tato sekce obsahuje informace o IP adrese, době běhu sytému, využití paměti, využití procesoru a verzi operačního systému.
- **Datum a čas**, v této sekci jsou informace o aktuálním čase, datu a tlačítko pro odeslání základních informací o stavu serveru.

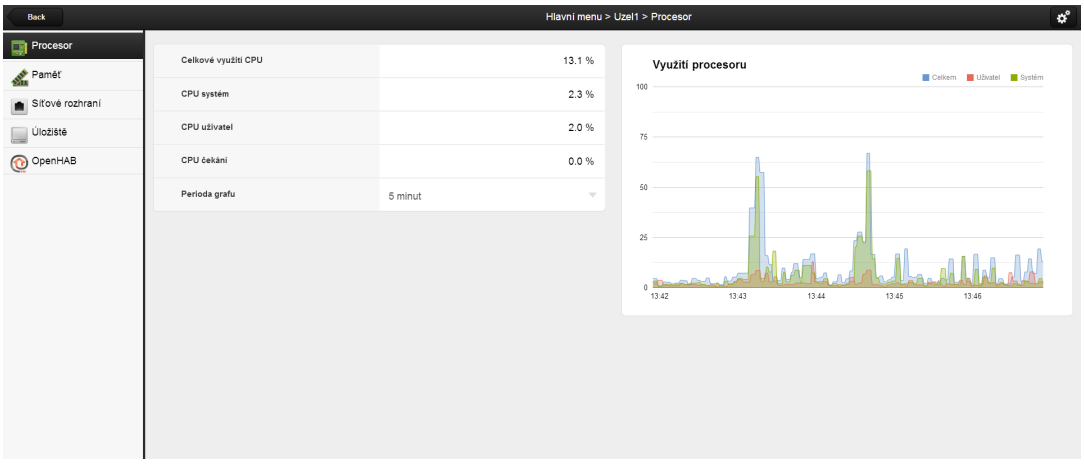


Obr. 5.8: Hlavní menu aplikace.

5.3.2 Procesor

Tato sekce je rozdělena do dvou bloků. Vpravo se nachází graf, v kterém je zobrazeno vytížení procesoru. Sekce Procesor je zobrazena na obrázku Obr. 5.9. Levý blok je rozdělen do pěti položek obsahujícím informace o procesoru:

- **Celkové využití CPU** - informace o celkovém využití procesoru,
- **CPU systém** - využití procesoru systémovými procesy,
- **CPU uživatel** - využití procesoru uživatelskými procesy,
- **CPU čekání** - doba čekání procesů na zpracování,
- **Perioda grafu** - nastavení periody grafu (5 minut, týden, hodina, měsíc).

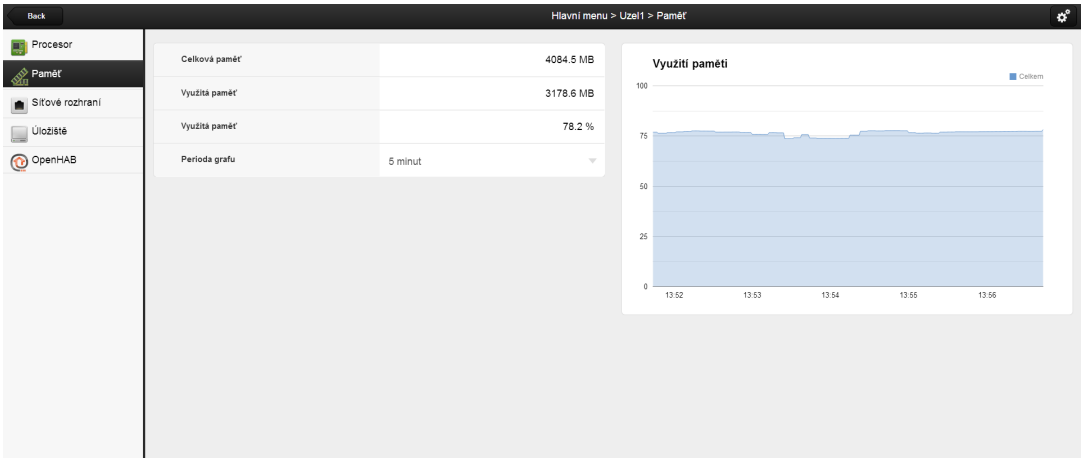


Obr. 5.9: Zobrazení sekce Procesor.

5.3.3 Paměť

Sekce paměť obsahuje dva bloky, jak je patrné z Obr. 5.10. Pravá sekce obsahuje graf využití paměti v procentech. V levé sekci jsou zobrazeny informace o využití operační paměti:

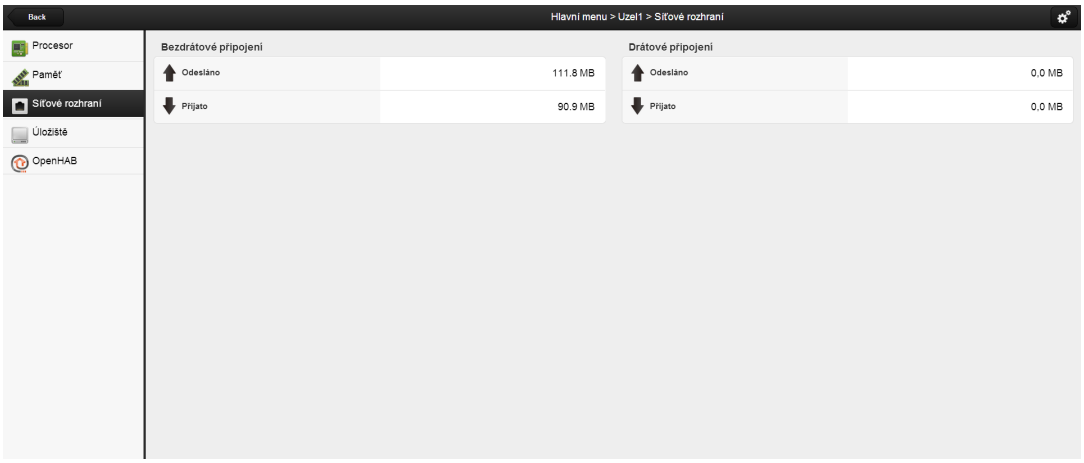
- **Celková paměť** - informace o celkovém velikosti paměti,
- **Využitá paměť** - využití paměti v megabajtech,
- **Využitá paměť** - využití paměti v procentech,
- **Perioda grafu** - nastavení periody grafu (5 minut, týden, hodina, měsíc).



Obr. 5.10: Zobrazení sekce Paměť.

5.3.4 Síťové rozhraní

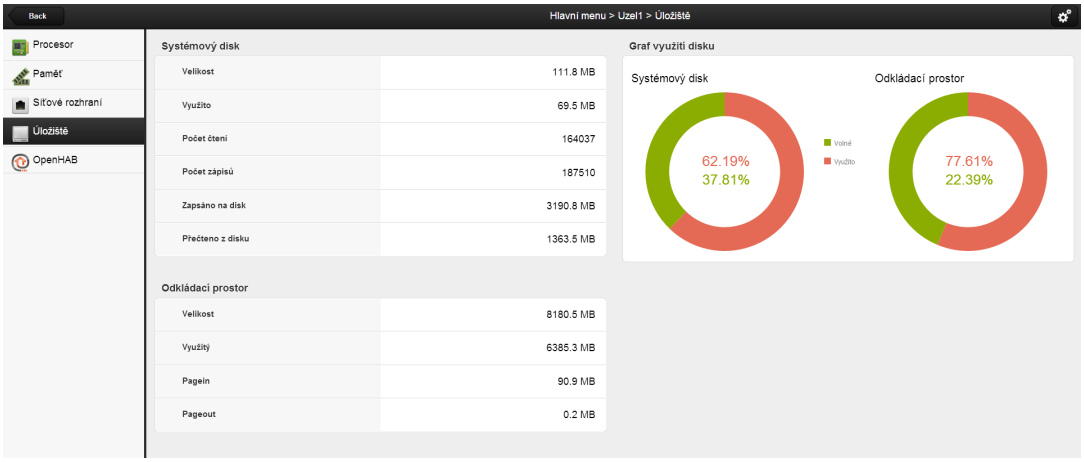
V této sekci jsou zobrazena data ze síťových karet, sekce je rozdělena do dvou bloků, jak je patrné z Obr. 5.11. V blocích je uveden počet přijatých a odeslaných bytů. V pravé sekci jsou zobrazena data z drátového připojení a v levé data z bezdrátového modulu.



Obr. 5.11: Zobrazení sekce Síťové rozhraní.

5.3.5 Úložiště

Sekce je rozdělena do tří bloků jak lze vidět na Obr. 5.12. První blok zobrazuje informace o využití pevného disku, druhý zobrazuje informace o využití odkládacího prostoru. Třetí blok obsahuje koláčový graf s aktuálním využitím pevného disku a odkládacího prostoru.



Obr. 5.12: Zobrazení sekce Úložište.

Systémový disk

V tomto bloku jsou uvedeny základní informace o systémovém disku:

- **Velikost** - informace o celkové velikosti disku,
- **Využito** - využití disku v megabajtech,
- **Počet čtení** - počet čtení z disku,
- **Počet zápisů** - počet zápisů na disk,
- **Zapsáno na disk** - velikost zapsaných dat od spuštění systému,
- **Přečteno z disku** - velikost přečtených dat od spuštění systému.

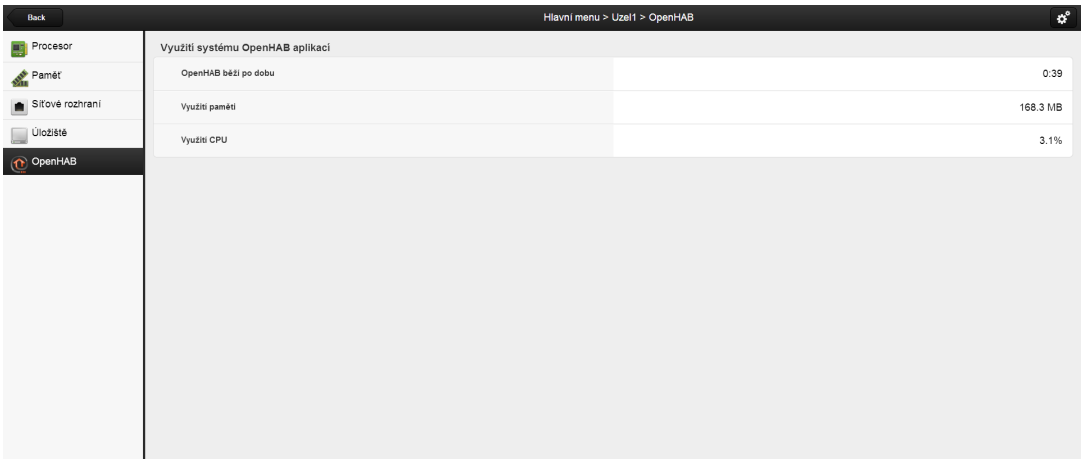
Odkládací prostor

Sekce obsahuje informace o využití odkládacího prostoru:

- **Velikost** - informace o velikosti odkládacího prostrou,
- **Využito** - využití odkládacího prostoru,
- **Pagein** - množství dat přenesených z odkládacího prostoru do paměti,
- **Pageout** - množství dat přenesených z paměti do odkládacího prostoru.

5.3.6 OpenHAB

Tato sekce obsahuje informace o využití stanice aplikací OpenHAB jak lze vidět na Obr 5.13. Položky zobrazují informace o době běhu OpenHAB serveru, využití paměti serverem a využití procesoru OpenHAB serverem.



Obr. 5.13: Zobrazení sekce OpenHAB.

6 ALTERNATIVY K OPENHAB

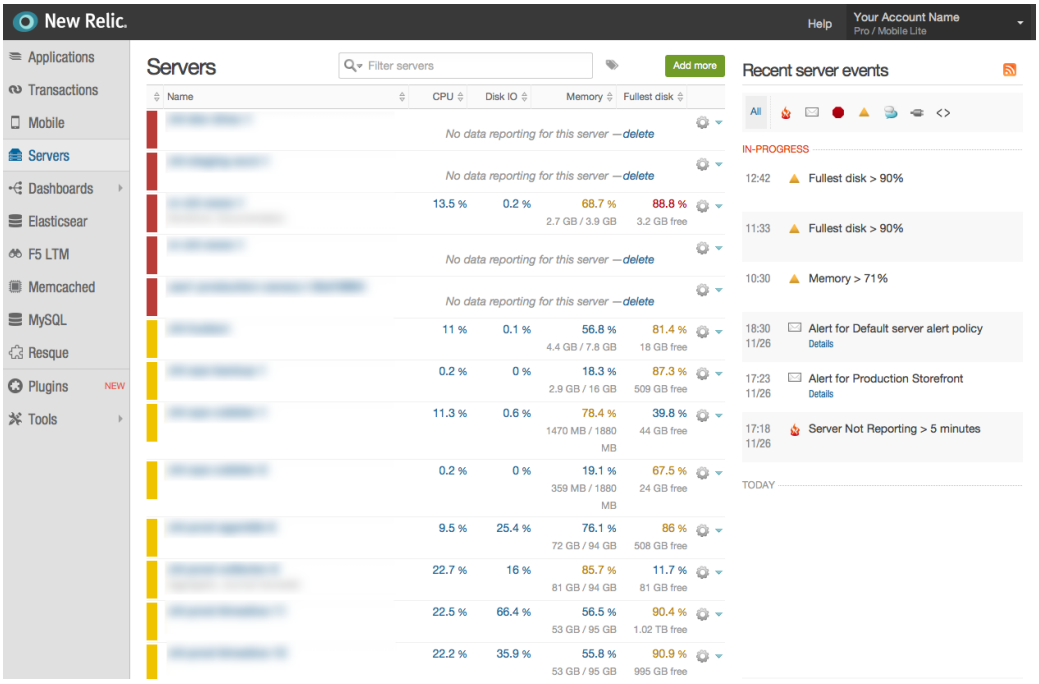
Sledováním systémových prostředků se zabývá velká řada společností. Všechny aplikace podávají základní informace o využití systémových informací jako jsou procesor, pevný disk a paměť.

Čím se od sebe aplikace odlišují jsou přidané funkce, které umožňují např. odesílání informací o stavu stanice či ukládání stavových informací do datbáze. Za tyto přidané funkce si většinou musíme zaplatit. V této kapitole bude uveden stručný přehled dostupných řešení.

6.1 New Relic SERVERS

Firma Relic nabízí množství produktů, které umožňují monitorování dat na různých zařízeních. Produkty firmy Relic nabízejí např. monitorování dat z prohlížeče, monitorování mobilních telefonů či měření výkonu aplikací. Pro monitorování serverů slouží aplikace New Relic SERVERS. Ta je dostupná pro operační systémy Linux, Windows, SmartOS a je plně bezplatná [10].

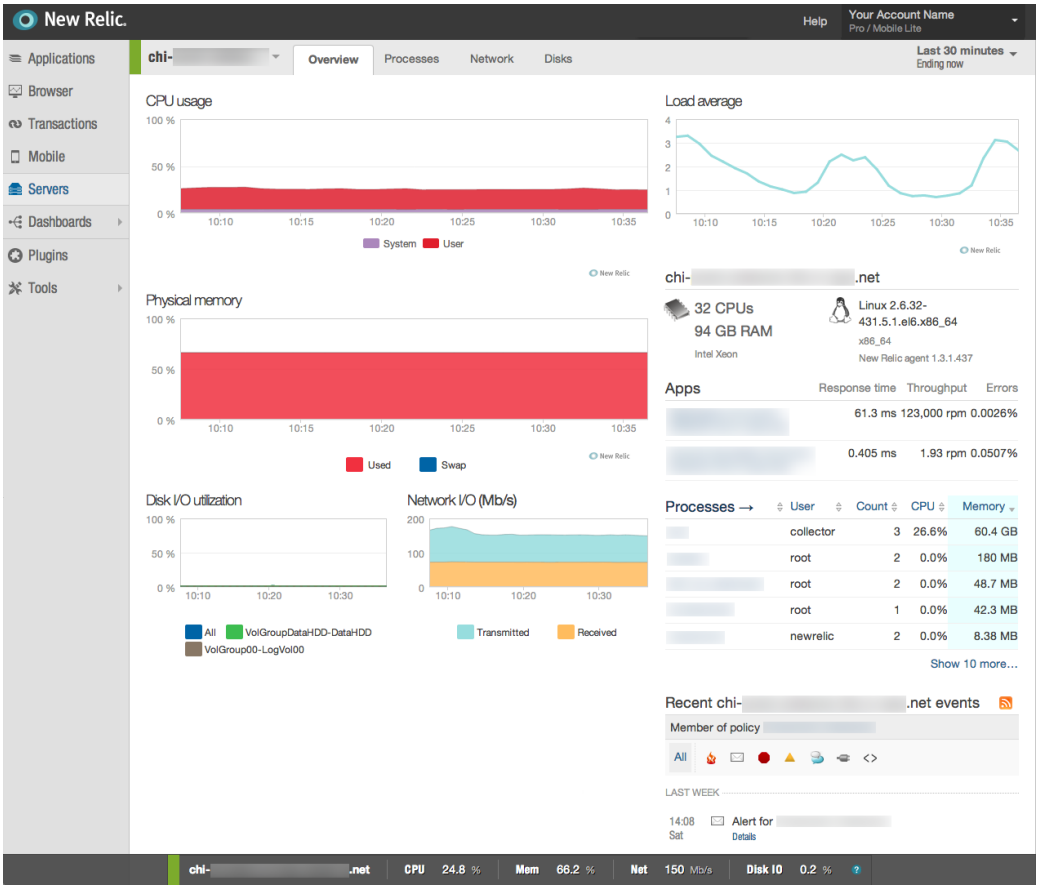
Po uspěšné instalaci a spuštění aplikace viz Obr. 6.1 je v záložce `Servers` seznam všech nalezených serverů. U každého serveru je také zobrazeno využití procesoru, paměti a disku. Vpravo je pak zobrazen přehled posledních událostí.



Obr. 6.1: Přehled dostupných serverů.

Pokud potřebujeme detailnější informace o stavu serveru, stačí na vybraný server přejít kliknutím. Dostaneme se na plochu Obr. 6.2, kde jsou zobrazeny detailní informace o využití serveru.

Na ploše jsou zobrazeny informace o operačním systému a jeho architektuře včetně počtu jader procesoru a velikosti operační paměti. Je zde také seznam monitorovaných aplikací a dále seznam běžících procesů. Informace o využití procesoru, paměti, disku a sítě jsou také přehledně zobrazeny v grafech. Pro procesy, využití disku a sítě je dostupné ještě podrobnější zobrazení informací, které se zobrazí po kliknutí na tyto položky [10].



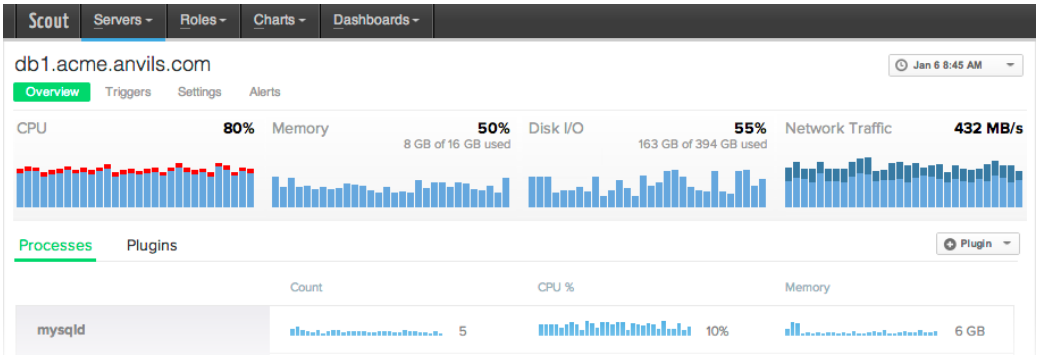
Obr. 6.2: Podrobné informace o stavu serveru.

6.2 Scout

Scout využívá tzv. Scout Agent, který běží na sledovaném serveru a získává z něj data. Scout Agenty využívá systémové prostředky pouze, když je spuštěn službou Cron a neměl by využívat více než 15 MB operační paměti. Agent je napsán v jazyce Ruby a šířen pod licencí MIT (Massachusetts Institute of Technology). Scout oficálně nepodporuje operační systém Windows.

Všechna data získaná Scout Agentem jsou odeslána na server společnosti Scout, kde jsou uložena do databáze pro další zpracování. Služba není bezplatná a je rozdělena do tří cenových balíčků dle poskytovaných služeb. U balíčků ve vyšší cenové hladině je dostupné např. odesílání upozornění pomocí SMS (Short Message Service).

Všechny funkce jsou v aplikaci Scout implementovány pomocí zásuvných modulů, kterých je více než 70. Veškerá konfigurace i instalace zásuvných modulů probíhá skrze webové rozhraní na serverech firmy Scout. Všechny zásuvné moduly můžeme libovolně upravovat, nebo napsat své vlastní. Na Obr. 6.3 je uveden příklad uživatelského rozhraní [17].



Obr. 6.3: Uživatelské rozhraní služby Scout.

6.3 PRTG Network Monitor

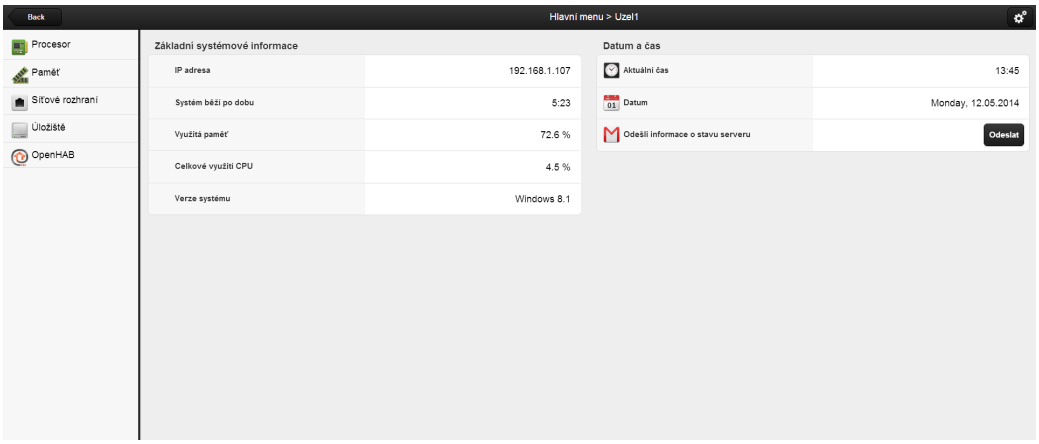
Tato aplikace je určena primárně pro sledování využití sítě. Umožňuje ale také sledování využití procesoru a paměti. Aplikace pracuje na serverech s operačním systémem Linux i Windows. PRTG Network Monitor obsahuje centrální server, který získává data pomocí vzdálených sond. Tyto sondy můžeme nainstalovat do lokální sítě a data z nich odesílat na centrální server. Sonda získává data pomocí tzv. senzorů, kterých existuje více než 150 druhů. V bezplatné verzi je počet senzorů omezen na 10.

Data o využití systémových prostředků jsou získáváná skrze WMI (Windows Management Instrumentation), WBEM (Web Based Enterprise Management) a SNMP (Simple Network Management Protocol). PRTG Network Monitor je velmi nenáročný na systémové prostředky, každý senzor využívá 150 kB operační paměti. Ovládání aplikace je uživatelsky přívětivé, veškerá konfigurace probíhá ve webovém rozhraní [13].

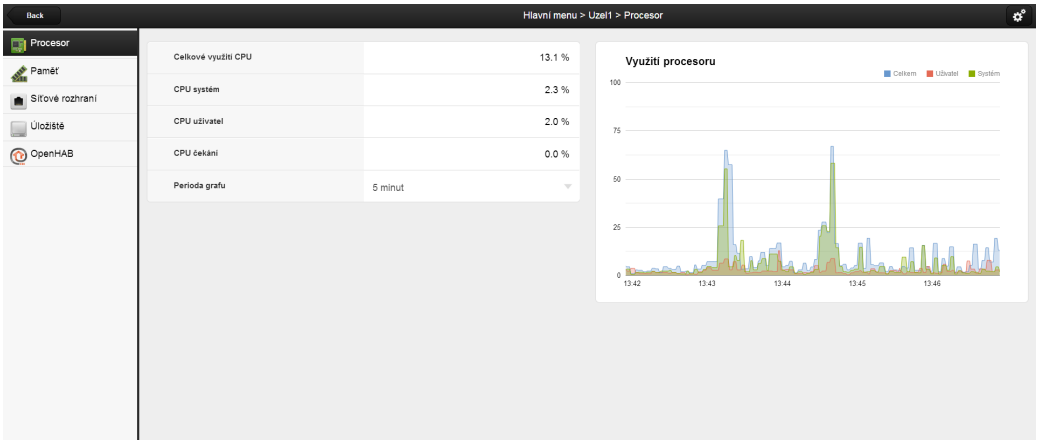
6.4 Porovnání s OpenHAB

Pomocí OpenHAB lze realizovat téměř totožné funkce, jaké nabízí popsané alternativy. Velkou výhodou OpenHAB řešení je jeho otevřený zdrojový kód. Umožňující provádět změny, které nám uzavřená řešení nedovolují. Kolem OpenHAB existuje rozsáhlá komunita, která v každé nové verzi přidává více funkcí a podporovaných technologií. OpenHAB umožňuje zasílání notifikací skrze e-mail, Tweeter či Notify My Android. Uživatel má také možnost své nastavení synchronizovat s úložištěm Dropbox.

Pro OpenHAB existuje také několik uživatelských rozhraní, takže uživatel není omezen vzhledem, který nabízí webová služba. Příklad uživatelského rozhraní jsou uvedeny na Obr. 6.4 a Obr. 6.5. OpenHAB nám umožňuje ukládat data do velkého množství databází bez omezení časem či jemností rozlišení. Sledované stanice můžeme zkontrolovat také pomocí mobilního telefonu s nainstalovanou aplikací pro OpenHAB.



Obr. 6.4: Hlavní menu v rozhraní GreenT.



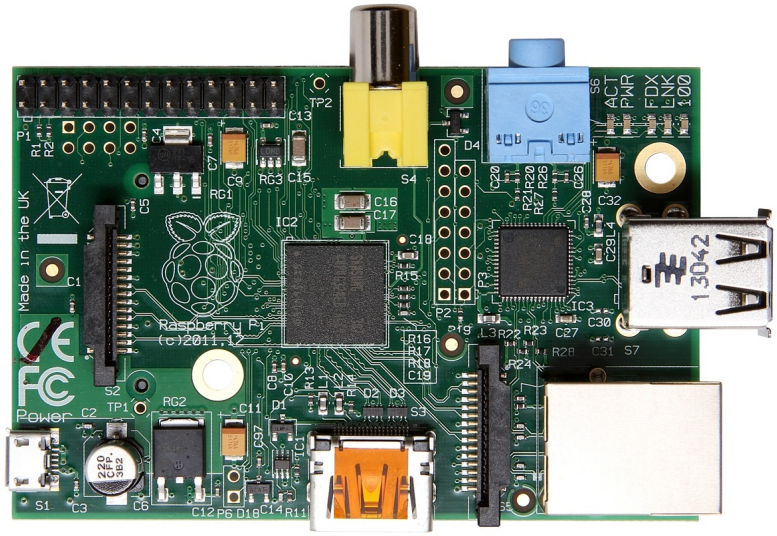
Obr. 6.5: Sekce Procesor v rozhraní GreenT.

7 KONCOVÉ SBĚRNÉ ZAŘÍZENÍ

Všechna data, která jsou získána ze sítě jsou odesílána na jeden koncový prvek. Na tomto prvku běží pouze jedna instnance OpenHAB serveru, z tohoto důvodu nemusí koncový uzel disponovat vysokým výpočetním výkonem. Důležitějším atributem je zde spotřeba, která by měla být co nejnižší. Z těchto důvodů bylo vybráno Raspberry PI pro sběr všech získaných dat.

7.1 Raspberry PI

Tento mini počítač založený na architektuře ARM vyvíjí britská nadace Raspberry PI Foundation. Raspberry PI je dostupné ve dvou verzích A a B. Model B můžeme vidět na Obr. 7.1. Verze B na rozdíl od verze A obsahuje integrovaný USB (Universal Serial Bus) rozbočovač a síťovou kartu. Model A i B je poháněn SoC (System on Chip) Bradcom BCM2835, který obsahuje procesor taktovaný na 700 MHz s jádrem ARM1176JZF-S náležícím do rodiny procesorů ARM11. Dále je obsažen také grafický akcelerátor Broadcom VideoCoreIV, který podporuje OpenGL ES 2.0. Akcelerátor obsahuje také kodér a dekodér MPEG-4 AVC. Po zaplacení licenčních poplatků je dostupný kodér i dekodér pro MPEG-2 a VC-1. Model A je vybaven 256MB operační paměti, model B od revize 2 disponuje 512MB operační paměti. Oba modely svou operační paměť sdílejí s grafickým akcelerátorem. Raspberry PI neobsahuje žádné úložiště, proto je pro instalaci systému určena SD karta, na kterou se systém nainstaluje. Systém můžeme nainstalovat i na pevný disk připojený pomocí USB [15].



Obr. 7.1: Raspberry PI model B.

Raspberry PI je osazeno 3,5mm jack zvukovým výstupem, dále jedním HDMI (High-Definition Multimedia Interface) video a audio výstupem. Je také osazen jeden kompozitní výstup pro připojení televizoru v PAL (Phase Alternating Line) či NTSC (National Television System Committee) standardu. Model B je vybaven dvěmi USB výstupy a jedním portem Ethernet (RJ45), model A nabízí pouze jeden USB port a neobsahuje Ethernet port.

Z těchto rozdílných parametrů plyne i rozdíl ve spotřebě obou zařízení. Model A má spotřebu pouze 300 mA (1,5 W) zatím co model B sputřebuje 700 mA (3,5 W). Pro napájení obou modelů slouží microUSB port nebo GPIO (General Purpose Input Output) s napětím o hodnotě 5V.

Raspberry PI obsahuje také specializované porty, které slouží k připojení rozšiřujících modelů. Mezi tyto rozšiřující porty patří [22]:

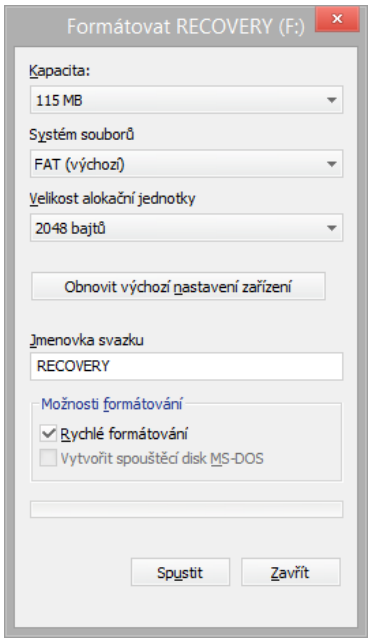
- **GPIO** poskytuje vstupně výstupní porty pro připojení např. UART či I^2C,
- **CSI** (Camera Serial Interface) slouží pro připojení speciálních kamer,
- **DSI** (Display Serial Interface) umožňuje připojení externího LCD modulu.

7.2 Instalace operačního systému

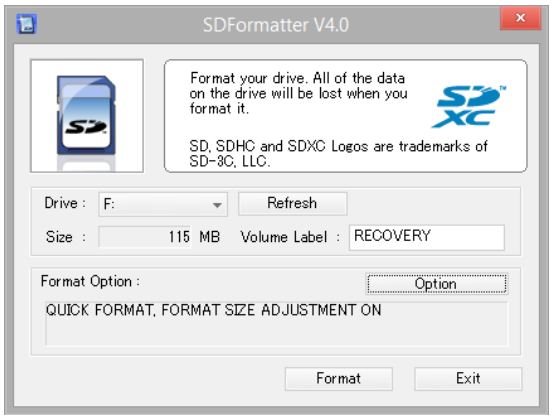
Díky velké oblibě zařízení Raspberry PI existuje mnoho programů pro usnadnění operačního systému. Komunita Raspberry PI Foundation, která stojí za vývojem Raspberry PI, představila instalační nástroj NOOBS (New Out of Box Software). Tento program jen zkopírujeme na paměťovou kartu a po spuštění Raspberry PI se systém nainstaluje.

7.2.1 Příprava paměťové karty

Pro jednoduchou instalaci využijeme instalační nástroj NOOBS. Abychom jej mohli použít musíme paměťovou kartu o velikosti nejméně 4GB naformátovat. Aplikace NOOBS vyžaduje souborový systém FAT (File Allocation Table). V instalačních instrukcích je doporučeno na systému Windows použít aplikaci SD Formatter Obr. 7.3, ale můžeme využít systémový nástroj pro formátování Obr. 7.2. Pro formátování karty v systému Linux můžeme použít nástroj Gparted. Po uspěšném naformátování karty nakopírujeme program NOOBS na paměťovou kartu. Program je zabalen v archivu zip, proto jej musíme před samotným kopírováním rozbalit. Poté již jen přesuneme obsah archivu na paměťovou kartu [15].



Obr. 7.2: Formátování Windows

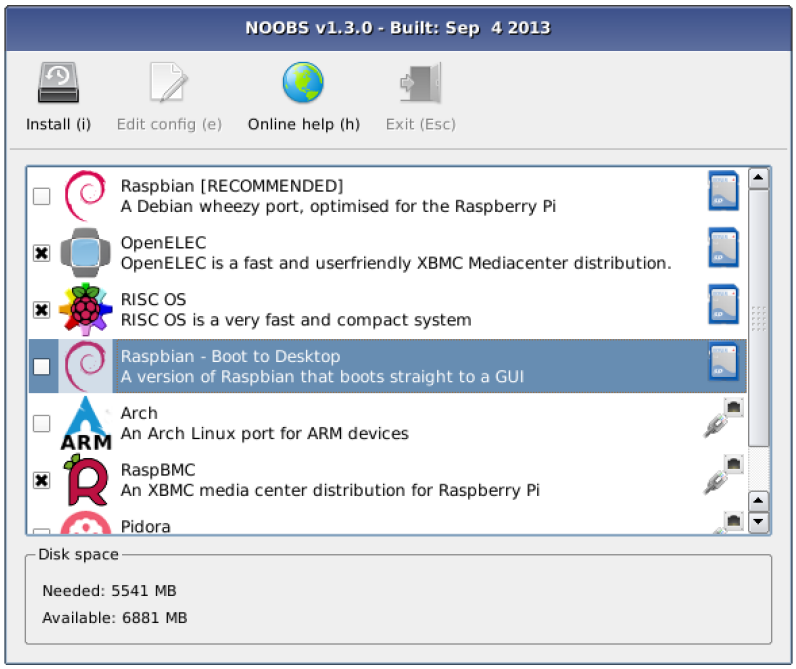


Obr. 7.3: Aplikace SDFormatter

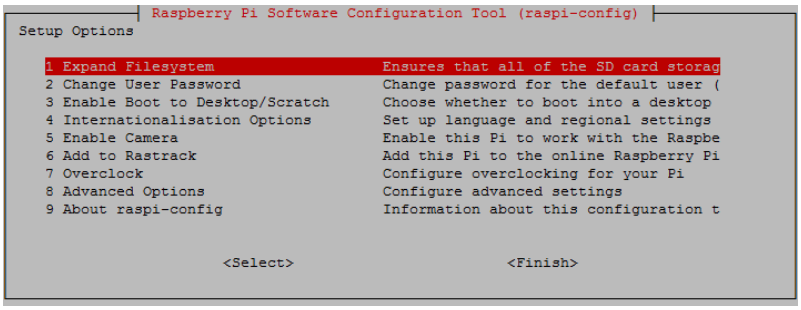
7.2.2 Instalace systému

Instalace operačního systému s programem NOOBS je velmi jednoduchá. Vložíme kartu do počítače a poté jen připojíme napájení. Program NOOBS funguje jako recovery režim, který nám dává možnost vybrat si systémy k instalaci. Od verze 1.3 umožňuje instalaci více systémů na jednu kartu, mezi těmito systémy si můžeme při startu vybírat. Pokud se operační systém poškodí umožňuje nám program NOOBS i jeho obnovu [15].

Po připojení napájení se nám při prvním spuštění zobrazí nabídka na instalaci systému viz Obr. 7.4. Vybereme si požadovaný operační systém a klikneme na `Install`. Po úspěšné instalaci systému se nám zobrazí nabídka pro konfiguraci viz Obr. 7.5. Zde si můžeme vybrat rozložení klávesnice, velikost přidělené paměti grafického akcelerátoru, prostředí do kterého se nám systém nastartuje nebo změnit velikost oddílů na paměťové kartě. Potvrzení nastavení tlačítkem `Finish` ukončí nabídku a dokončí spuštění systému.



Obr. 7.4: Výběr systémů k instalaci.



Obr. 7.5: Nabídka raspi-config.

7.3 Nastavení systému

V následující kapitole bude popsána instalace Java Runtime Enviroment a možný postup nastavení bezdrátového modulu.

Raspberry PI model A neobsahuje síťovou kartu a jedním z řešení tohoto problému je použití bezdrátového modulu. Abychom na Rasberry PI mohli spustit OpenHAB server musí být v systému nainstalován balíček Java Runtime Enviroment.

7.3.1 Nastavení bezdrátového modulu

Bezdrátový modul můžeme nastavit dvěma způsoby. Distribuce Raspbian obsahuje předinstalovanou aplikaci pro nastavení bezdrátových modulů, ta ale ve většině případů nefunguje. Druhou možností je nastavení bezdrátového připojení skrze terminál.

Musíme editovat soubor `/etc/network/interfaces`, to provedeme např. v editoru vim. Na konec souboru přidáme tyto řádky.

```
allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp
      wpa-ssid "SSID Wi-FI"
      wpa-psk "Heslo"
```

Po uložení souboru by mělo být bezdrátové připojení plně funkční.

7.3.2 Instalace Java

Linuxová distribuce Raspbian Wheezy je Java obsažena v základní instalaci. Můžeme ji nainstalovat i sami skrze terminál zadáním příkazů.

```
sudo apt-get update && sudo apt-get install oracle-java7-jdk
```

Po instalaci můžeme ověřit zda Java funguje správně příkazem.

```
java -version
```

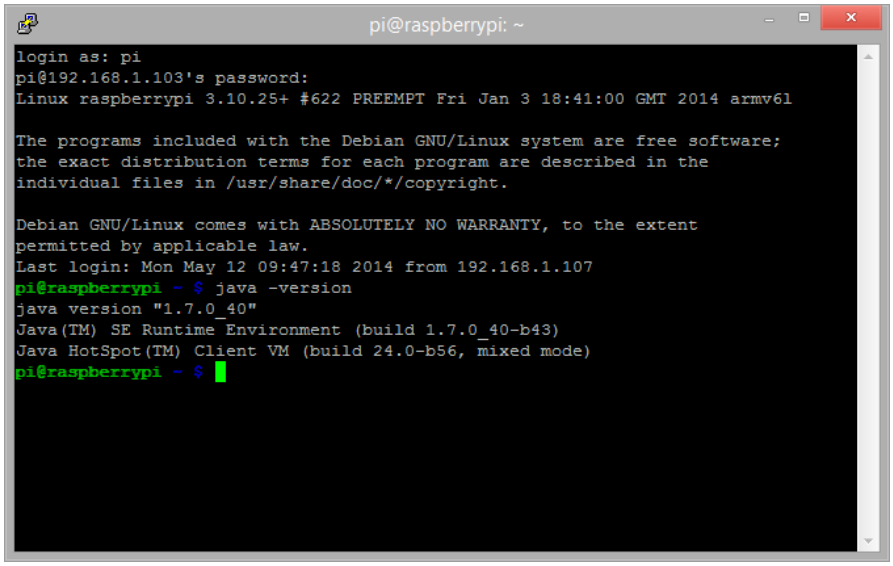
Výpis informací o Java Runtime Enviroment v konzoli můžeme vidět na Obr. 7.6.

7.3.3 Instalace OpenHAB

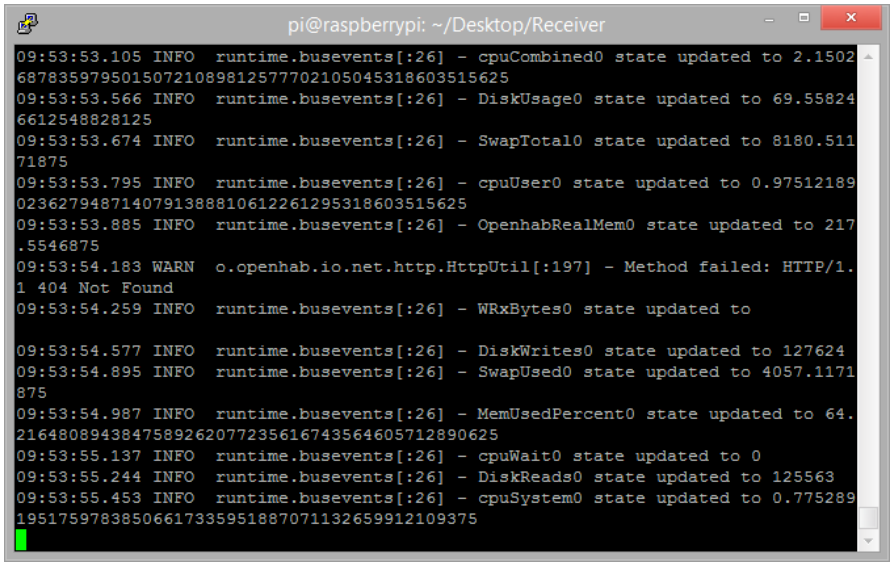
Před instalací OpenHAB musíme ze stránek `www.openhab.org` stáhnout OpenHAB Runtime, balíček je ve formě `zip`, proto jej můsíme rozbalit. Dalším krokem je

vytvoření konfiguračních souborů ve složce `/configuration`, ty můžeme vytvořit sami, nebo si pro demonstraci stáhnout demo konfiguraci ze webových stránek.

Pokud je v systému nainstalován Java Runtime Enviroment, stačí spustit terminál a přejít do složky OpenHAB. Server spustíme zadáním příkazu `sh start.sh`, poté budou spuštěny všechny potřebné moduly pro správnou funkci OpenHAB serveru. Konzolové okno OpenHAB serveru můžeme vidět na Obr. 7.7 V systému Windows se OpenHAB server spouští příkazem `start.bat`.



Obr. 7.6: Výpis informací o JRE.



Obr. 7.7: Okno OpenHAB Runtime.

8 ZÁVĚR

Cílem bakalářské práce bylo vytvoření funkční aplikace pro sledování využití systémových prostředků všech dostupných stanic v lokální síti s využitím technologie OpenHAB. Prvním krokem bylo zvolení vhodné metody pro získávání systémových informací ze stanic v lokální síti. Nakonec byla zvolena HTTP vazba, která získává data pomocí REST API.

Bylo zjištěno, že vazba systémových informací nedokáže získat data o oddílech v systému Windows. Tato chyba byla způsobena špatným zpracováním konfiguračního souboru a podařila se odstranit úpravou zdrojového kódu vazby.

Při aplikaci Google Charts jako prvku pro grafické zobrazení dat bylo zjišťěno, že žádný z balíčků perzistencí neumožňuje přístup k datům skrze programátorské rozhraní. Proto vznikl požadavek na vytvoření databáze umožňující přístup k uloženým datům. Po prozkoumání několika možností byla nakonec upravena perzistence RRD4J. Velkou výhodou této databáze je její konstantní velikost. Do balíčku RRD4J perzistence byl přidán servlet, který umožňuje přístup k datům s návratovým datovým typem JSON.

Cíle bakalářské práce, které jsou uvedeny v zadání jsou splněny. Výsledná vytvořená aplikace umožňuje nalezení všech dotupných instancí OpenHAB v síti. V uživatelském rozhraní jsou pak získaná data zobrazena v několika logicky dělených sekcích. Pro grafické zobrazení získaných dat slouží Google Charts, které se dokáží přizpůsobit různým rozlišením zobrazovacích zařízení.

Další vhodné rozšíření vytvořené aplikace by byla nezávislost na přesně definovaných názvech položek. Nyní musí názvy položek na síťových distribucích přesně odpovídat názvům položek na koncovém zařízení, jinak nebudou získána žádná data. Tato funkcionalita by se dala realizovat pomocí REST API, pomocí kterého by se získaly konfigurační soubory v XML formátu. Tyto sobory by poté musely být programově zpracovány a uloženy do konfiguračních souborů.

LITERATURA

[1] BOGAERT, Mathias. *RRD4J Documentation* [online]. 2014 [cit. 2014-04-13]. Dostupné z: `http://www.loriotpro.com`

[2] DNSDYNAMIC.ORG. *DNSdynamic: Absolutely Free Dynamic DNS* [online]. 2011 [cit. 2014-03-11]. Dostupné z: `https://www.dnsdynamic.org/`

[3] THE ECLIPSE FOUNDATION. *Eclipse: The Eclipse Foundation open source community website.* [online]. 2014 [cit. 2014-05-23]. Dostupné z: `http://www.eclipse.org/`

[4] FAYE, Fabien. *Generationip: rrdtool Round Robin Database Howto Version 1* [online]. 2014 [cit. 2014-04-13]. Dostupné z: `http://www.generationip.com/`

[5] FINNESAND DATA. *Servlets learning: Java Servlets Lifecycle* [online]. 2014 [cit. 2014-04-13]. Dostupné z: `http://w3processing.com/`

[6] GOOGLE INC. *Google Charts* [online]. 2012 [cit. 2014-03-11]. Dostupné z: `https://developers.google.com/chart`

[7] LUTEUS SARL. *Network Managment Software: Introduction to RRD - Round Robin Database* [online]. 2014 [cit. 2014-04-13]. Dostupné z: `http://rrd4j.googlecode.com/`

[8] KREUZER, Kai a Thomas ENGELEN. *Home Automation for Geeks* [online]. Antverpy (Belgie), 2012 [cit. 2013-11-16]. Dostupné z: `http://de.slideshare.net/xthirtynine/open-hab-devoxx-2012`.

[9] KYSILKA, Pavel. *Java na Webu III: Servlety* [online]. 2014 [cit. 2014-04-13]. Dostupné z: `http://www.linuxsoft.cz/`

[10] NEW RELIC, Inc. *Application Performance Management & Monitoring: New Relic* [online]. 2014 [cit. 2014-03-27]. Dostupné z: `http://newrelic.com`

[11] OPENHAB. *Empowering the smart home* [online]. 2011 [cit. 2013-11-16]. Dostupné z: `http://www.openhab.org/`

[12] OPENHAB. *Openhab samples* [online]. 2013 [cit. 2013-11-16]. Dostupné z: `https://code.google.com/p/openhab-samples/wiki/XSLTransforms`

[13] PAESSLER AG. *Network monitoring software, network performance management: Paessler* [online]. 2014 [cit. 2014-03-27]. Dostupné z: `http://www.paessler.com`

[14] PATEL, Ketan. *RRDtool: rrdtutorial* [online]. 2014 [cit. 2014-04-13]. Dostupné z: http://oss.oetiker.ch/

[15] RASPBERRY PI FOUNDATION. *Raspberry Pi* [online]. 2014 [cit. 2014-04-13]. Dostupné z: http://www.raspberrypi.org/

[16] SEN.SE. *Open.Sen.se* [online]. 2014 [cit. 2014-03-16]. Dostupné z: http://open.sen.se/

[17] SCOUT. *Hosted Server Monitoring: Scout* [online]. 2014 [cit. 2014-03-27]. Dostupné z: https://scoutapp.com/

[18] SIGAR - System Information Gatherer And Reporter. *Hyperic Support* [online]. 2010, 17. 9. [cit. 2014-02-26]. Dostupné z: https://support.hyperic.com/display/SIGAR/Home

[19] STROTMANN Carsten. *New DNS Technologies in the LAN* [online]. Iceland, 2007 [cit. 2013-11-02]. Dostupné z: http://meetings.ripe.net/ripe-55/presentations/strotmann-mdns.pdf

[20] TERRACOTTA, Inc. *Quartz Scheduler - Performance at Any Scale* [online]. 2012 [cit. 2013-11-02]. Dostupné z: http://quartz-scheduler.org/

[21] VIXIE, P. NETWORK WORKING GROUP. *Dynamic Updates in the Domain Name System (DNS UPDATE).* Woodside, 1997. Dostupné z: http://tools.ietf.org/rfc/rfc2136.txt

[22] VLASÁK, Filip a David ŠTARK. *RaspberryPi.cz – miniaturní levný počítač: Vše o mini počítači Raspberry Pi* [online]. 2014 [cit. 2014-04-13]. Dostupné z: http://raspberrypi.cz/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSI	Camera Serial Interface
CSS	Cascading Style Sheets
DDNS	Dynamic Domain Name System
DSI	Display Serial Interface
EPL	Eclipse Public License
DB4O	Database For Object
DNS	Domain Name System
FAT	File Allocation Table
GPIO	General Purpose Input Output
HDMI	High Definition Multimedia Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JAAS	Java Authentication and Authorization Service
JAR	Java ARchive
JSON	JavaScript Object Notation
mDNS	multicast Domain Name System
MIT	Massachusetts Institute of Technology
NAT	Network Address Translation
NFC	Near Field Communication
NOOBS	New Out of Box Software
NTSC	National Television System Committee

OSGi	Open Services Gateway initiative
PAL	Phase Alternating Line
RCP	Rich Client Platform
REST	Representational State Transfer
RRD4J	Round Robin Database For Java
RRs	Resource Records
RRset	Resource Record Set
SIGAR	System Information Gatherer And Reporter
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SoC	System on Chip
SSL	Secure Sockets Layer
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
TTL	Time To Live
UI	User Interface
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
USB	Universal Serial Bus
VML	Vector Markup Language
WBEM	Web Based Enterprise Management
WebApp	Web Application Micro Framework
WMI	Windows Management Instrumentation

XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XSL	eXtensible Stylesheet Language

SEZNAM PŘÍLOH

A OBSAH DVD

- Elektronická verze bakalářské práce – xstuse01_BP.pdf
- Elektronická verze bakalářské práce v LaTeX – ve složce Dokumentace_BP
- OpenHAB server pro zasílání informací – ve složce Sender
- OpenHAB server pro získávání informací – ve složce Receiver
- Zdrojové kódy pro databázi – ve složce RRD4J
- Zdrojové kódy pro Google Charts – ve složce Charts
- Zdrojové kódy programu pro nalezení stanic v síti – ve složce Config