**Jihočeská univerzita v Českých Budějovicích**
**Přírodovědecká fakulta**

# Metodika pro tvorbu vazeb mezi OpenHABem a inteligentním zařízením

Bakalářská práce

**Filip Oliva**

Školitel: Mgr. Jiří Pech, PhD.

České Budějovice
2017

# Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne 1. 4. 2017

Podpis:…….

# Poděkování

Tímto bych rád poděkoval vedoucímu mé bakalářské práce panu Mgr. Jiřímu Pechovi, Ph. D. za jeho rady, přístup, odborné vedení a především čas, který mi věnoval při zpracování této problematiky.

Oliva, F., 2017: Metodika pro tvorbu vazeb mezi openHABem a inteligentním zařízením. [Methodology for creating linkages between openHAB and intelligent devices. Bc. Thesis, in Czech.] – p., Faculty of Science, University of South Bohemia in České Budějovice, Czech Republic

# Abstrakt

Tématem mé bakalářské práce je sestrojení a popis vazby (binding) mezi softwarovým nástrojem pro ovládání inteligentního domu openHAB a inteligentním zařízením, v mém případě se jedná o tvorbu vazby mezi bezdrátovou zásuvkou AC-88 od firmy Jablotron, která je ovládána prostřednictvím systému OpenHAB.

Klíčová slova: inteligentní dům, řídící jednotky, Raspberry Pi, vazba, OpenHAB, automatizace, Turris gadgets, konfigurace

# Abstract

The topic of my bachelor thesis is creation and description of bind between software tool for control of smart home openHAB and intelligent device, in my case it is a creation of bind between control unit Raspberry Pi which is managed through systém openHAB and smart device of Jablotron company.

Keywords: smart home, control systém, automatization, Raspberry Pi, bind, OpenHAB, Turris gadgets, configuration

# Obsah

# 1 Úvod

S nástupem automatizace domácností dochází také k rozvoji inteligentních zařízení. Snahou výrobců je vytvořit zařízení s co možná největším počtem funkcí, ale zároveň musí být energeticky nenáročné. Domácí automatizace se zařazuje do množiny řešení, která se nazývá Internet of Things. V českém slovníku se pro tento výraz ustálil pojem Internet věcí. Můžeme ho také definovat jako množinu zařízení, která jsou součástí domácí sítě nebo internetové infrastruktury. Již několik let se tomuto tématu věnuje více a více pozornosti, protože s vývojem chytrých zařízení dochází k usnadňování každodenního života a tím i úspoře času. Podle firem, které se zabývají průzkumem trhu, by se měl do roku 2020 počet zařízení zapojených do sítě Internet of Things zvýšit o několik desítek miliard. Díky tomuto odhadu se stává využívání chytrých zařízení ve vlastní síti velmi zajímavé pro spoustu firem nejen z oblasti informatiky. (Svět hardware, 2016)

V tuto chvíli existuje nepřeberné množství inteligentních zařízení, a to od různých výrobců, přičemž každý využívá jiné způsoby komunikace a jiné uživatelské rozhraní. Zde vznikají problémy s kompatibilitou, které se projevují při rozšiřování chytré domácnosti. Například nemůžeme kombinovat chytrá zařízení od jednoho výrobce s ovládacím rozhraním vyvinutým jinou firmou.

Jedno z řešení tohoto problému je využití open-source projektu OpenHAB. Jeho cílem je integrace všech prvků chytrého domu do jedné aplikace, a to nezávisle na výrobci či použité technologii přenosu. Mezi výhody patří kompatibilita nejpoužívanějších operačních systému, mezi které patří Windows, MacOS X a Linux s nainstalovanou Javou 1.7 nebo s jeho novější verzí. Další výhoda je využití single board platformy Raspberry Pi, BeagleBone Black, Banana PI nebo Orange Pi. Možnost připojení chytrého prvku nám zprostředkovává vazba tzv. bindings.

Momentálně existuje více než čtyřicet hotových vazeb, které je možné si stáhnout ze stránek projektu. V této bakalářské práce se zabývám metodikou tvorby vazeb mezi OpenHAB a inteligentním zařízením, nejdříve přiblížím, co to je Internet of Things a na jaké bázi funguje tento open-source projekt a jak je možné postupovat při jeho nastavení a tvorby uživatelské aplikace. Ve vlastní části bakalářské práce popisuji tvorby vazby a následné připojení ke konkrétnímu zařízení.

# 2 Cíle a metodika práce

## 2.1 Cíle práce

Tato bakalářská práce je rozdělena do dvou částí, teoretické a praktické. Cíle práce jsou následující:

1. Přiblížit problematiku chytrých zařízení a Internet of Things
2. Představit systém OpenHAB a jeho architekturu
3. Popsat systém tvorby mezi programem OpenHAB a inteligentním zařízením
4. Tvorba konkrétní vazby

## 2.2 Metodika práce

Při zpracování mé bakalářské práce jsem vytvářel a následně také popisoval vazbu mezi řídícím systém OpenHAB a bezdrátovou zásuvkou od firmy Jablotron. Obě zařízení byla zapůjčena od Oddělení robotiky a embedded systémů na Přírodovědecké fakultě Jihočeské univerzity.

Základem pro tvorbu projektu je nastudování literárních pramenů a základní znalosti programování v jazyce Java a oblasti chytrých zařízení. Jelikož se jedná o sektor, který se stále rozvíjí, jsou informace čerpány především z internetových zdrojů. Vlastní práce představuje vytvoření vazby, která je následně popsána a interpretována v praxi.

Pro řešení této problematiky jsem použil následující hardware:

- Raspberry Pi 2
- SD karta 8GB
- Bezdrátová zásuvka Jablotron AC – 88
- Jablotron Turris Dongle
- Router TP-LINK WR841N

Dále jsem využíval tento software:

- Java 1.8.0_121
- Eclipse IDE for Java Developers
- Raspbian Jessie (kernel verze 4.4)
- OpenHAB 2– řídící systém
- Mozzilla Firefox
- SmartHome Designer – nástroj pro nastavení konfiguračních souborů

# 3 Literární rešerše

## 3.1 Internet of Things

### 3.1.1 Pojem Internet of Things

Internet of Things je poměrně nový termín, který byl poprvé zmíněn Kevinem Ashtonem v roce 1999. Tehdy byl použit ve spojení s radiofrekvenční identifikací (RFID) a v té době rozvíjejícím se pojmem internet. Tento nový pojem se skládá ze dvou slov a to ze slov „Internet“ a „Věci“. (Ashton, 2009)

Internet of Things sdružuje miliony soukromých, veřejných, akademických, podnikatelských a vládních sítí, které jsou propojeny za účelem výměny dat a poskytovaní služeb. Z technického hlediska, můžeme považovat věc za cokoli živého i neživého. Jsou to objekty, se kterými každý den setkáváme každý den např.: oblečení, nábytek, elektronické zařízení, zvířata a mnoho dalšího. To znamená, že věci mohou být živé i neživé a lze je identifikovat jako objekty ve fyzickém nebo hmotném světě. (Nunberg, 2012)

### 3.1.2 Definice IoT

Neexistuje žádná jedinečná definice pro Internet of Things, která by byla přijatelná pro celosvětovou komunitu uživatelů. Ve skutečnosti existuje mnoho různých skupin, včetně akademiků, vědců, vývojářů a firem, kteří tento termín definovali, i když jeho první použití je přičítáno Kevinu Ashtonovi, Britskému odborníkovy na digitální technologie. Existuje mnoho definic, ale všechny mají jedno společného a to, že první verze internetu má data tvořená lidmi a další verze bude mít data tvořená věcmi. Nejlepší definici pro Internet of Things bude:

*„Jedná se otevřenou a komplexní sít inteligentních objektů, které mají schopnost automaticky organizovat informace, data a zdroje, reagovat a konat vzhledem k situaci a pracovnímu prostředí.“* (Murer, 2010)

Internet of Thing se vyvíjí a je jedním z mnoha nejvíce medializovaných konceptů v IT ve světě, představuje vizi globální infrastruktury síťových fyzických objektů, která umožnuje kdykoliv a kdekoliv připojení na cokoliv. Internet věcí může být považován za globální síť, ve které má každý objekt jedinečnou identitu, která umožnuje komunikaci člověka s člověkem, člověka s věcí a věci s věcí. (E. A. Kosmatos, 2011)

Internet of Things popisuje svět, ve kterém může být cokoli připojeno do sítě a komunikovat inteligentním způsobem. Pod pojem „cokoli“, nejsou myšlena jen elektronické zařízení, jako jsou počítače, servery, tablety a chytré telefony. Ale i zařízení jako jsou čidla, senzory, termostaty, zásuvky, bezpečnostní systémy, ledničky a televize, které je možné ovládat a kontrolovat jejich činnost, po připojení prostřednictvím kabelových a bezdrátových sítí, často za použití TCP/IP protokolu do internetu. (E. A. Kosmatos, 2011)

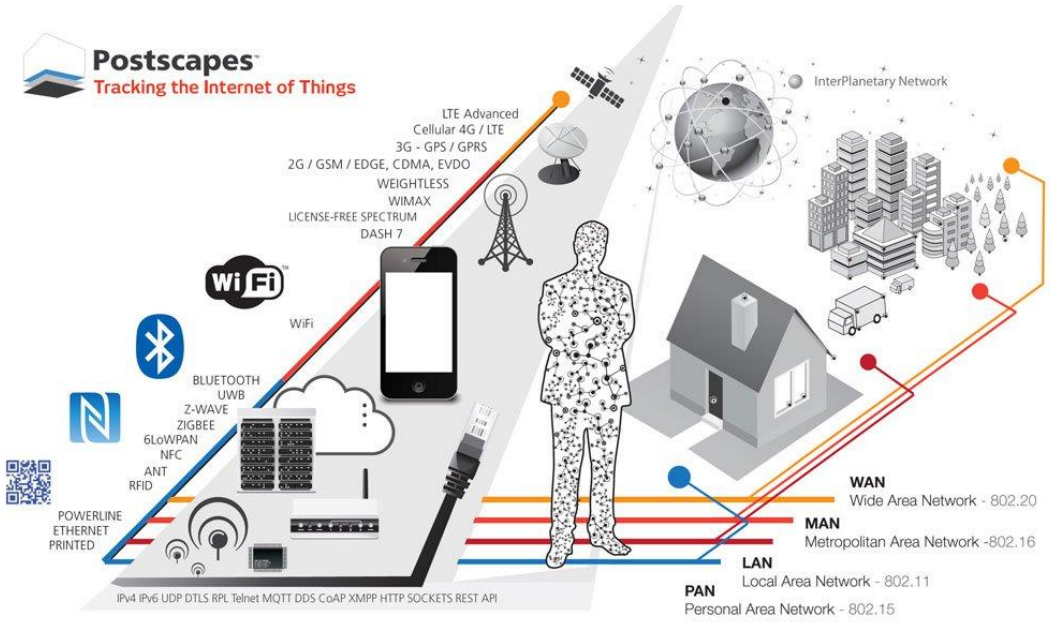
### 3.1.3 Architektura a technologie

Pokud chceme realizovat síť IoT, nejsme omezeni pouze na jednu domácnost, ale můžeme budovat propojení nezávisle na geografické rozlehlosti. Můžeme tedy využít prvky osobních (PAN), lokálních (LAN), městských (MAN) a celosvětových (WAN) sítí.

Připojení Internet of Things je možné s pomocí drátových i bezdrátových technologií. Mezi nejčastější drátové připojení patří norma ETHERNET, která je standardem pro vedení počítačových sítí. Další možností je využití bezdrátového připojení, kde největší podíl představuje WI-FI, tedy bezdrátová síť tvořená s pomocí mikrovln. Dále je možné využít technologie, které pracují na bázi rádiových vln. Jedná se technologie: RFID, NFC, Bluetooth, ZIGBEE, Z-WAVE a 6LoWPAN.

Výběr daných technologií je podmíněn jejich výhodami a nevýhodami v oblasti, kde chceme síť budovat, na finančních možnostech, na obtížnosti řešení a našich preferencích. Architektury a technologie využívají otevřených, standardních protokolů, které umožňují vytvořit nové internetové aplikace, včetně obchodních, podnikových a výrobních procesů. Každá věc, objekt, produkt nebo chytrý výrobek mají svůj ekvivalent v infrastruktuře IoT ve formě agenta. Optimalizovaná technologie Internetu věcí zahrnuje rozdělení inteligence mezi distribuované a více centralizované systémy typu ERP (Enterprise Resource Planning). Důležitou úlohu mají čidla, snímače a senzory, v rámci IoT je můžeme využívat například k měření teploty, rychlosti, zrychlení, spotřeby energie, zapnutí/vypnutí a dalším činnostem. Ze senzorů generovaná data se dále zpracovávají nejdříve agentem, který je virtuálním programovým reprezentantem věci, objektu, zařízení, z kterých se stávají chytré a inteligentní věci. V rámci průmyslových systémů komunikují agenty zpracovávající signály ve vertikálním nebo horizontálním směru, často v heterogenním prostředí, s pomocí jazyků typu XML.
(Ing. Pavel Burian, 2014)

Pro představu fungování Internet of Things a jeho realizace slouží následující schéma:

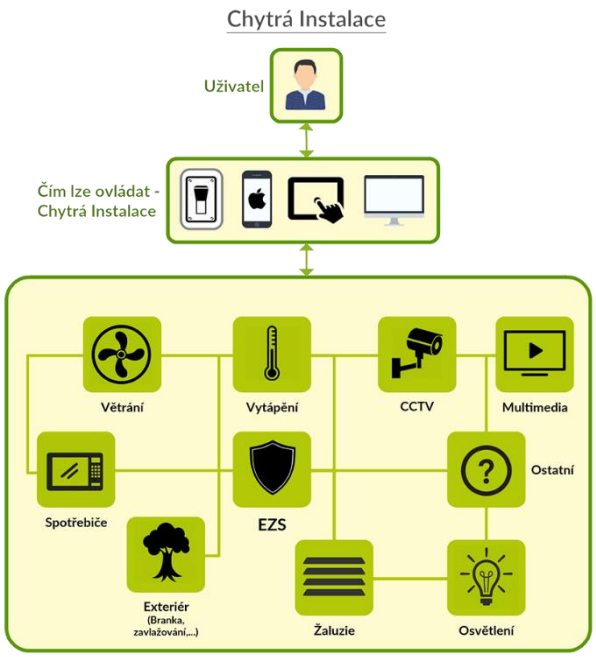



Obrázek 1: Schéma technologií (Postscapes, 2016)

# 3.2 Inteligentní dům

## 3.2.1 Pojem Smart Home System

Inteligentní dům je pojmem, který představuje domácí zařízení, které je částečně nebo kompletně automatizované. Jednotlivá zařízení jsou navzájem propojená, což umožňuje jejich uživatelům snazší ovládání domu a poskytuje informace o tom, co se v daný okamžik s jednotlivými zařízeními děje. Pod pojmem automatizace se nemyslí jen dálkové ovládání termostatu či vytápěcího kotle, ale obsahuje v sobě velkou spoustu funkcí, které nám zjednodušují každodenní činnost. Kvůli vysokým nákladům bylo její nasazení dříve využíváno jen v komerčních objektech, jako jsou obchodní centra, hotely nebo sídla společností. Ale s vývojem nových technologií a zároveň s poklesem jejich pořizovacích cen, se automatizace začíná postupně využívat v rodinných domech, bytech a v podobných nekomerčních objektech.




Obrázek 2: Funkce inteligentního domu (Chytrá instalace, 2015)

### 3.2.2 Podstata inteligentního domu

Základem každého inteligentního domu je řídící jednotka, která zprostředkovává komunikaci jednotlivých prvků mezi sebou a přístup uživatele k funkcím systému. Zařízení mohou být k jednotce připojena několika způsoby, využívá se drátové i bezdrátové řešení instalace. Obě řešení jsou vzájemně kompatibilní a není tak problém je mezi sebou kombinovat. (Trtík, 2009)

První zmínka o inteligentním domu byla prezentována v 60. letech 20. století v Japonsku, kdy centrem veškerých funkcí byl řídící počítač. Tento způsob se však nesetkal s velkým úspěchem. Za počátek zrodu jednotné koncepce inteligentních domů a elektroinstalační techniky vůbec lze považovat rok 1987, kdy založily firmy Berker, Gira, Merten a Siemens společnost Instabus Gemeinschaft. Cílem bylo vyvinout systém pro měření, řízení a sledování provozních stavů v budovách. (Trtík, 2009)

### 3.2.3 Řídící systém

Provoz inteligentních domů zajišťují řídící systémy. Jednotlivé prvky a zařízení inteligentního domu spolu musí komunikovat, aby bylo zajištěno jejich efektivní využití. O komunikaci a správnou funkci všech zařízení se stará řídící systém. Dnes existuje mnoho komerčních řídících systémů, které jsou vyvinuty společnostmi pro svá zařízení a není možné je rozšiřovat o zařízení jiných výrobců, protože tyto řídící systém nepodporuje. Řešením této nevýhody jsou open-source projekty jako je třeba OpenHAB, Domotiga, DomotiCZ, Home Assistant a mnoho dalších. (Turan, 2010)

### 3.2.4 Řídící jednotka

Nedílnou součástí řízení inteligentního domu, je také hardware, na kterém poběží řídící systém. Je možné použít jakékoli zařízení s operačním systém Linux, ale z hlediska spotřeby, velikosti a nízké pořizovací ceny jsou používány jednodeskové počítače, mezi nejznámější patří Raspberry Pi, ale existuje mnoho klonů či alternativ, například BeagleBone, UDOO apod. Pro účely tohoto projektu byla zvolena platforma RaspBerry pi 2, která díky své malé spotřebě umožní bez problémů celodenní provoz za velmi nízkých nákladů. (Raspberry Pi 2, 2015)

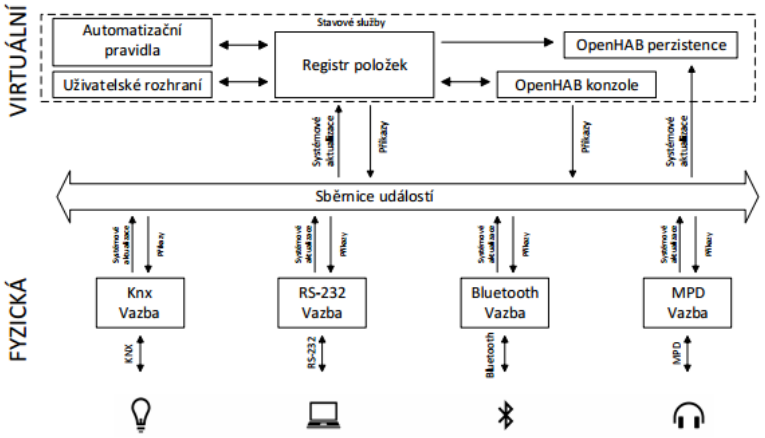
## 3.3 Systém OpenHAB

OpenHAB (open Home Automation Bus) je open source software, který slouží k ovládání chytrých domů. Jeho hlavní výhodou je, že sjednocuje různé systémy do jednoho řešení. Neřeší konfigurací jednotlivých zařízení, ale sjednocení jejich ovládání do jedné aplikace. Toto spojení je řešeno pomocí bindings (takzvaných vazeb), které řeší napojení komunikace zařízení s OpenHABem. Je napsán v jazyce Java za pomoci modulárního frameworku OSGi, který umožňuje přidávat či odebírat moduly a přitom není potřeba službu pozastavit. K ovládání dochází skrze webové rozhraní nebo aplikace pro mobilní zařízení, kterými jsou mobilní telefony s operačními systémy Android, iOS nebo Windows Mobile či tablety. Ke konfiguraci slouží nástroj SmartHome Designer. (OpenHAB empowering the smart home, 2017)

### 3.3.1 Architektura

Systém openHAB využívá OSGi, neboli Open Services Gateway initiative Framework, který umožňuje přidávání a odebírání modulů za běhu aplikace bez nutnosti zastavení služby. (Engelen, 2012)

Na obrázku č.3 je evidentní blokové schéma aplikace OpenHAB.




Obrázek 3: Schéma aplikace OpenHAB *(Slideshare, 2012)*

**Sběrnice událostí**

Sběrnice událostí je hlavním prvkem architektury OpenHAB, označována také jako Event Bus. Umožňuje komunikaci mezi moduly, které využívají sběrnici k oznámení událostí ostatním a aktualizují svůj stav na základě vnějších podnětů. Jedná se o dva typy událostí, kterými jsou systémové aktualizace, které informují o změně stavu zařízení a příkazy, které vykonávají činnost nebo mění stav zařízení. (Engelen, 2012)

Všechny vazby komunikují po sběrnici, což znamená, že je zde velmi nízká vazba mezi svazky, proto má OpenHAB dynamickou povahu. Většinou na centrálním serveru běží pouze jeden openhab-runtime, ale OSGi Event Admin může být využit i jako řídící služba, která umožní propojení více distribucí OpenHAB právě přes sběrnici událostí. (OpenHAB, 2011)

**Vazby**

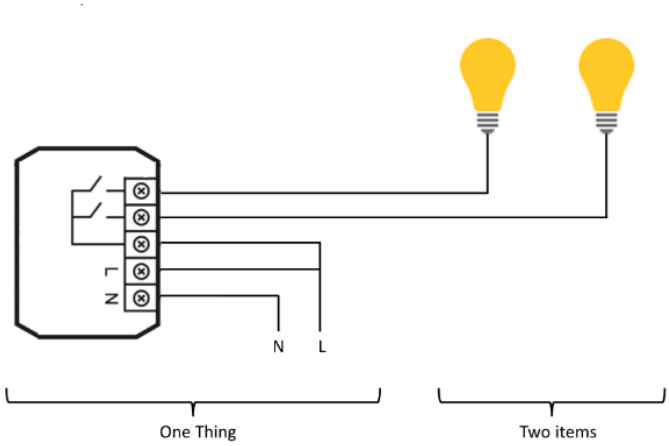
Anglicky bindings, česky vazby, představují balíčky, které jsou používány pro rozšíření rozhraní OpenHAB, lze s jejich pomocí získat přístup ke komunikačnímu softwaru nebo je také možné je připojit ke sběrnici domácí automatizace. Existuje asi 40 různých vazeb, které rozšiřují jednotlivé funkce. (OpenHAB empowering the smart home, 2017)

**Konfigurace**

Konfigurace OpenHABu je realizována textovými soubory, které je možné editovat v libovolném textovém editoru. Pokud vytváříme náročnější konfigurace, je výhodnější použít nástroj Eclipse SmartHome Designer, který je postaven na vývojovém prostředí Eclipse. (Kai Kreuzer, 2017)

Konfiguraci jako takovou lze rozdělit do několika souborů neboli skupin, které jsou následující:

- **Items (položky)** – jedná se o objekty, do kterých mohou být zapsány hodnoty, nebo které mohou být přečteny tak, aby došlo k interakci mezi nimi. Mohou být svázány také vazbou a definovány v souborech, které se nachází v **openhab_home/configurations/items**, příponu items musí mít všechny soubory dané definice položky. Pokud jsou položky typicky definovány pomocí OpenHAB Designeru, a to úpravou definičního souboru položky. Výhodou je možnost využití plné podpory vývojového prostředí, jako je například kontrola syntaxe.
- **Sitemaps (mapy stránek)** – používají se k vytvoření základů uživatelského rozhraní, aby mohly být jednotlivé položky OpenHABu dostupné u různých nadstaveb. Jde o deklarativní definice uživatelského rozhraní. Strukturou a obsahem obrazovky uživatelského rozhraní lze definovat také krátkým skriptem. Soubory jsou pak uloženy v **openhab_hobe/configurations/sitemaps**.
- **Things** – jsou zařízení jako celek v rámci OpenHABu. (Kai Kreuzer, 2017)




Obrázek 4: Příklad zapojení zařízení a jednotlivých položek *(Bip Philippe Gitbooks, 2015)*

- **Pravidla (Rules)** – jsou používány pro automatizaci procesů, každé pravidlo může být spouštěčem, který vyvolává skript, který provádí všechny druhy úkolů (například zapnou světla změnou své položky, dělat matematické výpočty v jazyce časovače atd.) OpenHAB funguje na bázi integrovaných a vysoce přesných pravidel. (Kai Kreuzer, 2017)

### 3.3.2 Uživatelské rozhraní

**Paper UI**

Uživatelské rozhraní Paper UI je webová aplikace vytvořená v HTML5, do které implementuje Google's Material Design a je velmi všestranné, takže plynule funguje na všech velikostech obrazovky. Všechny moderní prohlížeče, mezi které zařazujeme Safari, Chrome, Firefox, jsou podporovány v jejich nejnovější verzi. Jediný Internet Explorer postrádá podporu SSE. Paper UI se využívá především pro konfigurační účely a správu, nikoli pro provoz, na ty se odkazují základní UI rozhraní. (Kai Kreuzer, 2017)

**HABmin**

Moderní, profesionální a přenosné uživatelské rozhraní pro OpenHAB, to je HABmin. Poskytuje administrativní funkce (např. Sitemaps pro uživatele a konfigurační nástroje na podporu nastavení). (Kai Kreuzer, 2017)

Mezi hlavní funkce patří:

- Všestrannost – funguje na všech zařízeních
- Tématika – k dispozici je několik motivů, používají se bootswatch témata
- Mapování – moderní, rychle mapuje historická data
- Pravidla grafického editoru – není třeba se učit pravidla syntaxe
- Mezinárodní podpora – v současné době angličtina, němčina, francouzština, a další. (Kai Kreuzer, 2017)

**Basic UI**

Basic UI je základním uživatelským rozhraním, které je založené na Material Design Lite od Googlu. Funkcemi tohoto rozhraní jsou: všestranné dispozice vhodné pro různé velikosti obrazovek, navigace AJAX a živé aktualizace. (Kai Kreuzer, 2017)

**Classic UI**

Rozhraní Classic UI je původním pro OpenHAB , a proto je nejstabilnější a nejrozšířenější UI k dnešnímu dni. Nicméně vzhled neodpovídá moderním standardům, například Basic UI je brán za jeho nástupce. Klasické rozhraní je založeno na rámci WebApp.Net a mohou být použity v každém (WebKit bázi) webovém prohlížeči. Aplikace WebApp.Net se skládá hlavně z JavaScriptu a CSS souborů. (Kai Kreuzer, 2017)

### 3.3.3 Zabezpečení

Pro zabezpečení řídícího systému OpenHAB se využívají následující dva mechanismy:

- **HTTPS** – Hypertext Transfer Protocol Secure, jedná se o zabezpečovací verzi protokolu HTTP, která chrání například falšování dat nebo jejich odposlouchávání.
- **Autentizace** – požadavky pro autentizaci jsou odeslány na přístupový bod JAAS, tedy Java authentication & autorization Service, ten zajišťuje povolení autentizovat uživatele. (Kai Kreuzer, 2017)

### 3.3.4 Eclipse SmartHome Designer

Eclipse SmartHome Designer je nástroj pro správu konfiguračních souborů OpenHABu. Je vytvořeno na vývojovém prostředí Eclipse. Hlavní výhoda je v přehlednosti a zobrazování syntaxe. (Kai Kreuzer, 2017)

# 4 Praktická část

## 4.1 Návrh

V rámci své bakalářské práce jsem vytvořil vazbu neboli propojení mezi chytrým zařízením Jablotron AC-88 a řídícím systémem OpenHAB. Základem tvorby bylo seznámení se s fungováním OpenHABu jako takového, s jednotlivým zařízeními ze sady Turris Gadgets a zjišťoval jsem možnosti jejich použití pro můj projekt. Po domluvě s vedoucím práce jsem obdržel k vývoji zařízení Jablotron AC-88.

## 4.2 Turris Gadgets

Speciální projekt připravený ve spolupráci se společností Jablotron Alarms a.s. se nazývá Turris Gadgets, jedná se o sadu, která s pomocí routeru vytvoří řešení pro chytré domácnosti. Sada byla katedře Aplikované informatiky předána na testování od Cesnetu.

Základní komponentou je USB rozhraní Turris Dongle. Toto rozhraní zajišťuje komunikaci mezi periferiemi ze sady a řídící jednotkou, přes rádiové vlny na frekvenci 868,5 MHZ. Komunikace mezi hostitelem a modulem je řešena přes rozhraní USB jako virtuální sériový port (tzv. USB CDC). Toto zajišťuje integrovaný obvod od firmy FTDI - FT230X. (turris.cz, 2016)

Sériový port musí být inicializován v režimu 57600 8N1, což znamená:

- Komunikační rychlost 57600 Bd
- 8 bitů šířka slova
- Žádná parita
- 1 stop bit

Mezi řídící jednotkou a turris donglem jsou 3 základní typy komunikace:

- Příkaz (směr hostitel → modul)
- Odpověď (směr modul → hostitel)
- Zpráva (směr modul → hostitel)

Komunikační protokol je ve formátu ASCII. Každá komunikace musí být zahájena znakem `ESCAPE \x1B` a musí být zakončena znakem `\n`.

Správný formát příkazu je `\x1B OBSAH\n` (bez mezer mezi obsahem a značkami). Pokud Turris Dongle nepřijme platný příkaz, vrátí odpověď **ERROR.** (turris.cz, 2016)

### 4.2.1 Chytrá zásuvka Jablotron AC-88

Dále je potřeba registrovat zásuvku k Turris donglu. Každý výrobek, který vysílá, má svůj jedinečný čtrnáctimístný kód. Jeho posledních osm číslic představuje sériové číslo, jehož prostřednictvím se registrují periferie do modulu Turris Dongle. Čtrnáctimístný kód i sériové číslo se nachází na spodní straně zásuvky viz foto níže. (turris.cz, 2016)


Obrázek 5: Zásuvka AC-88

Modul přijímá pouze periferie, které jsou do něj registrované, která se nachází v tzv. slotech ve vnitřní flash paměti modulu. Ve výchozí sadě jsou všechny prvky v modulu již registrované a pro rychlý start toto není nutné řešit. Do modulu lze registrovat až 32 periferií.

Pro registraci slotu slouží příkaz `SET SLOT:XX [YYYYYYYY].` (turris.cz, 2016)

- Pole XX reprezentuje číslo slotu (rozsah 00..31).
- Pole YYYYYYYY je dekadická 24-bitová adresa periferie.

Odpověď na tento příkaz je OK.

Pro smazání všech slotů složí příkaz `ERASE ALL SLOTS`.

- Odpověď na tento příkaz je OK.

Zásuvka se ovládá s pomocí stavové věty: `TX ENROLL:E PGX:C PGY:Y ALARM:A BEEP:BBBB`, hodnoty této věty jsou následující:

**Pole E - registrační signál:**

- Hodnota 1 - vysílá se registrační signál pro přijímače
- Hodnota 0 - nevysílá se registrační signál pro přijímače

**Pole X - stav výstupu X:**

- Hodnota 1 - výstup je sepnutý
- Hodnota 0 - výstup je rozepnutý

**Pole Y - stav výstupu Y:**

- Hodnota 1 - výstup je sepnutý
- Hodnota 0 - výstup je rozepnutý

**Pole A - stav poplach:**

- Hodnota 1 - poplach je aktivní
- Hodnota 0 - poplach není aktivní

**Pole BBBB – stav zvukového výstupu:**

- Hodnota NONE - zvuk není aktivní
- Hodnota SLOW - pípání je pomalé
- Hodnota FAST - pípání je rychlé (turris.cz, 2016)

## 4.3 Konfigurace Raspberry Pi 2

### 4.3.1 Instalace Systému

Pro fungování Raspberry Pi 2 je nutné nejdříve nainstalovat operační systém, pro mé účely byl zvolen systém Raspbian. Obraz systému se stáhne z webových stránek raspberrypi.org, tento systém se přehraje na SD kartu pomocí programu WIN32DiskImager. Celý tento proces probíhá v PC nebo notebooku, po dokončení instalace se karta odpojí, vyjme se a vloží do Raspberry. To zapojíme do elektrické sítě a síťovým kabelem do routeru, následně se přihlásíme přes SSH.

### 4.3.2 Konfigurace sítové rozhraní

Aby mělo Rapsberry Pi vždy stejnou IP adresu je potřeba editovat soubor `/etc/network/interfaces`. Sítové rozhraní upravíme tímto způsobem

```
auto eth0
iface eth0 inet static
    address 192.168.0.2
    netmask 255.255.255.0
    gateway 192.168.0.1
    dns-server 192.168.0.1
```

Soubor uložíme a provedeme tento příkaz, který nám načte změny ze souboru:

```
sudo /etc/init.d/networking reload
```

### 4.3.3 Instalace OpenHABu

Instalace OpenHABu začíná načtením a aktualizací balíků ze zdrojů:

```
sudo apt-get update
sudo apt-get upgrade
```

Nainstalujeme tyto nástroje:

```
sudo apt-get install screen mc vim git htop
```

Nejdříve musíte nainstalovat poslední verzi Java, zadáme sérii těchto příkazů:

```
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu
xenial main" | sudo tee /etc/apt/sources.list.d/webupd8team-
java.list

echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu
xenial main" | sudo tee -a
/etc/apt/sources.list.d/webupd8team-java.list

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --
recv-keys EEA14886

sudo apt-get update
sudo apt-get install oracle-java8-installer
sudo apt-get install oracle-java8-set-default
```

Instalace OpenHABU může probíhat dvěma způsoby, a to skrze balík z repozitáře nebo ručně ze souboru. Pro své účely jsem zvolil první způsob, tedy repozitář.

Nyní musíme přidat repozitář openHAB 2 Bintray, použije se tento odkaz:

```
Wget-qO -
'https://bintray.com/user/downloadSubjectPublicKey?username=op
enhab' | sudo apt-key add -,

echo 'deb http://dl.bintray.com/openhab/apt-repo2 stable main'
| sudo tee /etc/apt/sources.list.d/openhab2.list
```

Následně aktualizujeme balíky :

```
sudo apt-get update
```

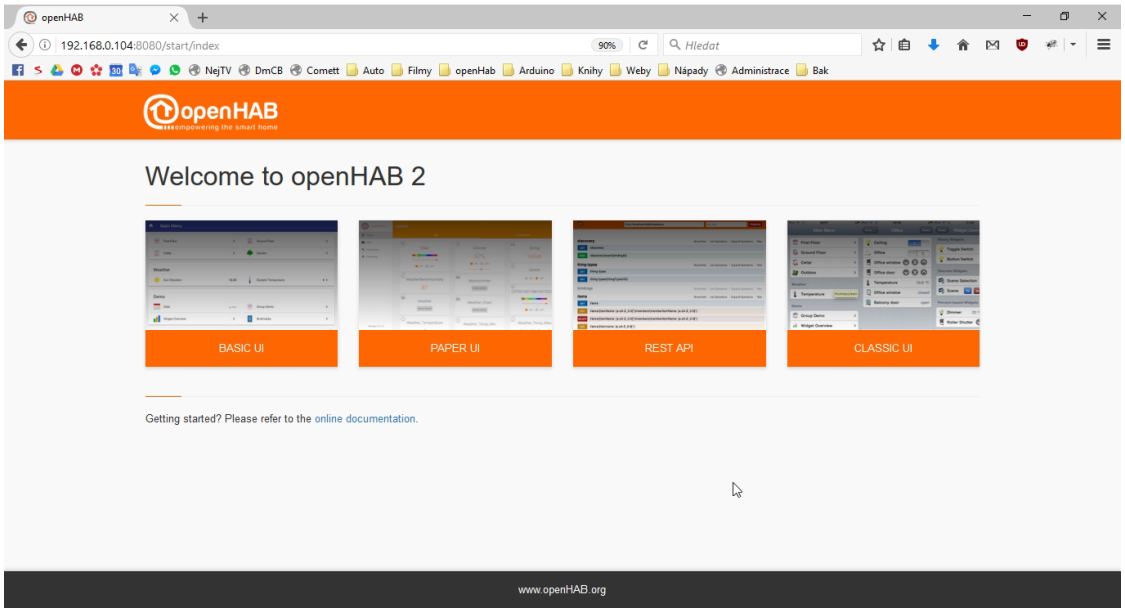
A provedete instalaci OpenHABu:

```
sudo apt-get install openhab2
```

Následně musíme zadat tyto příkazy, tím zapneme službu OpenHAB a zkontrolovali jsme, že je zapnutá a nastaví se automatické spuštění při startu systému.

```
sudo systemctl start openhab2.service
sudo systemctl status openhab2.service
sudo systemctl daemon-reload
sudo systemctl enable openhab2.service
```

Nyní je OpenHAB nainstalován a k jeho uživatelským prostředím přistoupíte na IP adrese Raspberry na portu 8080, v mém případě na této adrese: http://192.168.0.2:8080




Obrázek 6: Úvodní stránka OpenHABu

### 4.3.4 Nastavení USB portu

Pro nastavení komunikace Turris Donglu a řídící jednotkou je potřeba stáhnout a nainstalovat ovladač. Ten stáhneme z této stránky: www.ftdichip.com/Products/ICs/FT230X.html

Soubor rozbalíme a obsah adresáře `release/build` překopírujeme do `/usr/local/lib` a restartujeme Raspberry Pi.

Z tohoto odkazu stáhneme vzorové skripty: www.github.com/CZ-NIC/gadgets_demo

Pro ověření komunikace řídící jednotky s Turris Donglem použijeme tento skript `gadget_command.py`. Příkazem ve formátu `WHO AM I?` lze zjistit verzi firmwaru, případně vyzkoušet funkčnost spojení. (turris.cz, 2016)

Celý příkaz bude zadán takto: `sudo python gadget_command.py "WHO AM I?"`

Na tento příkaz pošle modul odpověď ve formátu `TURRIS DONGLE V#.#`, kde `#.#` představuje verzi firmware. (turris.cz, 2016)

Po připojení Turris Donglu do USB portu, je zařízení připojeno do `/dev/ttyUSB0`. Aby OpenHAB mohl komunikovat s tímto zařízením, je potřeba mu přidělit oprávnění. Toho docílíme změnou majitele přípojného bodu na `openhab`. Aby se nám po odpojení Turris Donglu z USB portu nebo restartu Raspberry neměnil přípojný bod nebo majitel je potřeba vložit příslušné pravidlo do souboru `/etc/udev/rules.d/30-usb-seriál.rules`.

Hodnoty pro pravidlo zjistíme z výpisu tohoto příkazu:

```
sudo udevadm info -a -p $(udevadm info -q path -n /dev/ttyUSB0)
```

V mém případě jsem přidal toto pravidlo:

```
SUBSYSTEM=="tty", ATTRS{serial}=="DJ002558",
SYMLINK+="turrisdongle",
OWNER="openhab", GROUP="dialout", MODE="0660"
```

Po uložení stačí pravidla načíst z konfigurace příkazem:

```
sudo service udev reload
```

## 4.4 Příprava vývojového prostředí

Pro vlastní tvorbu vazby si musíme připravit vývojové prostředí. Je potřeba nainstalovat tyto programy:

- Git
- Maven 3.x
- Oracle JDK 8

Vývoj probíhá v Eclipse IDE. Eclipse Installer automaticky připraví veškerá nastavení. Stačí postupovat podle těchto kroků:

1. Instalaci stáhneme zde: https://wiki.eclipse.org/Eclipse_Installer
2. Spustíme Eclipse Installer a v pravém menu přepneme do Advanced Mode
3. Zvolíme Eclipse IDE for Java Developers" a zadáme "Další"
4. Rozbalíme volbu GitHub Projects a zvolítem openHAB Development a openHAB2 Add-ons
5. Vybereme název adresáře a dáme Next
6. Instalaci spustíme kliknutím na Finish

Při tvorbě binding je možné použít dva jmenné prostory. Na výběr se nabízí tyto dvě možnosti:

- **org.eclipse.smarthome** - chcete-li přímo přispívat do projektu Eclipse Smarthome, vybereme tento jmenný prostor. Výhodou této varianty je, že tvorba je dostupná i pro jiné projekty než OpenHAB, které jsou založeny na Eclipse Smarthome. Nevýhodou je, že tento proces schvalování vašich příspěvků do projektu je přísnější, neboť zahrnuje i kontrolu duševního vlastnictví.
- **org.openhab** – je vhodný pokud chcete, aby jmenný prostor byl použit pouze pro openHAB. Jedná se o nejlepší možnost, pokud vaše vazba není zajímavá pro jiné řešení, anebo vyžaduje speciální knihovny. (OpenHAB, 2011)

## 4.5 Tvorba vazby

### 4.5.1 Vytvoření struktury vazby

Ve své práci jsem zvolil jmenný prostor org.openhab. Nejdříve je potřeba vytvořit kostru projektu. K tomu nám slouží skript `create_openhab_binding_skeleton.sh`. Ten nalezneme v adresáři, kam jsme nainstalovali Eclipse IDE v umítění `git/openhab2-addons/addons/binding`. Skript je volán s jedním parametrem, kterým je název naší vazby. V mém případě jsem použil:

```
./create_openhab_binding_skeleton.sh turrisgadgets
```

Tím se mi v umístění `git/openhab2-addons/addons/binding` vytvořila základní struktura projektu.

```
|- ESH-INF
|---- binding
|------- binding.xml
|---- thing
|------- thing-types.xml
|- META-INF
|---- MANIFEST.MF
|- OSGI-INF
|---- TurrisGadgetsHandlerFactory.xml
|- src
|---- main
|------- java
|---------- [...]
|- build.properties
|- pom.xml
```

Vytvořený projekt importujeme do Eclipse IDE. V horním menu zvolíme: `File-> Import->General->Existing Projects into Workspace` a vybereme složku s nově vytvořeným projektem. (Eclipse IoT project, 2016)

### 4.5.2 Příprava OpenHABu

Před psaním samotné vazby, vytvoříme testovací uživatelské prostředí v rámci Eclipse IDE. V levém menu Package Exploler přejdeme do umístění `Infrastructure-> distro-resources-> src-> main-> resources`.

Zde najdeme jednotlivé složky pro konfiguraci OpenHABu. Ve složce `items` upravíme soubor demo.items do této podoby:

```
Switch PGY "Switch"
{channel="turrisgadgets:turrisdongle:dongle01:pgy"}
Switch PGX "Switch"
{channel="turrisgadgets:turrisdongle:dongle01:pgx"}
Switch Alarm "Switch"
{channel="turrisgadgets:turrisdongle:dongle01:alarm"}
Switch Beep "Switch"
{channel="turrisgadgets:turrisdongle:dongle01:beep"}
```

Ve složce `sitemaps` upravíme soubor demo.sitemaps do této podoby:

```
sitemap demo label="Main Menu"
{
   Frame label=AC88{
                Switch item=PGY label="AC88 PGY Switch"
                Switch item=PGX label="AC88 PGX Switch"
                Switch item=Alarm label="AC88 Alarm Switch"
                Switch item=Beep label="AC88 BEEP Switch"
             }
```

A jako poslední věc upravíme soubor demo.things ve složce `things`. Záznam se zapisuje v tomto tvaru:

<binding-id>:<thing-type-id>:<thing-id> [parametry vazby]

```
turrisgadgets:turrisdongle:dongle01 [port="/dev/ttyUSB0"]
```

### 4.5.3 Definice vazby

Každá vazba potřebuje definovat v souboru binding.xml, který se nachází v adresáři `/ESH-INF/binding/`.V tomto souboru se uvádějí informace o autorovi a popis vazby. Nejdůležitější položkou je binding ID, který je unikátním identifikátorem vazby v rámci OpenHABu. (Eclipse IoT project, 2016)

Definice vazby v mém případě vypadá následujícím způsobem:

```
<?xml version="1.0" encoding="UTF-8"?>
<binding:binding id="turrisgadgets"
     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xmlns:binding=http://eclipse.org/smarthome/schemas/binding/v1.0.0
    xsi:schemaLocation="http://eclipse.org/smarthome/schemas/binding/v1.0.0
http://eclipse.org/smarthome/schemas/binding-1.0.0.xsd">
   <name>TurrisGadgets Binding</name>
   <description>This is the binding for TurrisGadgets.</description>
   <author>Filip Oliva</author>
</binding:binding>
```

### 4.5.4 Things

V OpenHABu jsou externí systémy reprezentovány jako Things. Ty jsou nadefinovány jako ThingType v souboru thing-types.xml, který se nachází v `/ESH-INF/thing/`. Každá funkce Things by měla být popsána jako Channel, česky kanál. Je důležité určit, jaký typ Items s nimi může být spojen. Níže je výpis ze souborů vazby Turrisgadgets.

```
<?xml version="1.0" encoding="UTF-8"?>
<thing:thing-descriptions bindingId="turrisgadgets"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:thing="http://eclipse.org/smarthome/schemas/thing-
description/v1.0.0"
        xsi:schemaLocation="http://eclipse.org/smarthome/schemas/thing-
description/v1.0.0 http://eclipse.org/smarthome/schemas/thing-description-
1.0.0.xsd">

    <!-- Sample Thing Type -->

    <thing-type id="turrisdongle">
        <label>Turris Dongle</label>
        <description>Thing for Turris Gadgets Binding</description>

        <channels>
            <channel id="pgx" typeId="pgx"/>
            <channel id="pgy" typeId="pgy"/>
            <channel id="alarm" typeId="alarm"/>
            <channel id="beep" typeId="beep"/>
        </channels>
```

```
    </thing-type>
    <!-- Sample Channel Type -->
    <channel-type id="pgx">
        <item-type>Switch</item-type>
        <label>Input PGX</label>
        <description>Change state on input PGX</description>
    </channel-type>

    <channel-type id="pgy">
        <item-type>Switch</item-type>
        <label>Input PGY</label>
        <description>Change state on input PGY</description>
    </channel-type>

    <channel-type id="alarm">
        <item-type>Switch</item-type>
        <label>Input  ALARM</label>
        <description>Change state on input ALARM</description>
    </channel-type>

    <channel-type id="beep">
        <item-type>Switch</item-type>
        <label>Input  BEEP</label>
        <description>Change state on input BEEP</description>
    </channel-type>
</thing:thing-descriptions>
```

### 4.5.5 Thing status

Každý objekt Things může měnit svůj stav. Ten nám může odhalit případné problémy se zařízením nebo službou.

- UNINITIALIZED - toto je počáteční stav Things, když je přidán nebo je OpenHAB spuštěn. Tento stav je také přiřazený v případě, když proces inicializace selhal nebo není vazba k dispozici. Příkazy odesílané na kanálech nebudou zpracovány.
- INITIALIZING - tento stav je přiřazen, když je načítán stav Things. Příkazy odesílané na kanálech nebudou zpracovány.
- UNKNOWN - příkazy odeslané na kanálech mohou být zpracovány, ale OpenHAB nemůže rozpoznat zda je zřízení ONLINE nebo OFFLINE. V případě, že není možné příkazy zpracovat, je přepnut stav na OFFLINE.
- ONLINE - zařízení bylo inicializováno a zpracovává příkazy
- OFFLINE - zařízení nepracuje správně, nezpracovává příkazy.
- REMOVING - odstranění Things ze systému.
- REMOVED - Things bylo smazáno z OpenHABu

### 4.5.6 Třídy projektu

V projektu jsou od základu vytvořeny 3 třídy, které jsou základní kostrou vazby. V mé práci jsem je upravil následujícím způsobem.

1. **TurrisGadgetsBindingConstants –** v této třídě jsou nadefinované hlavní proměnné projektu Binding ID, ThingTypeUID a seznam Channels ID.

```
public class TurrisGadgetsBindingConstants {

    public static final String BINDING_ID = "turrisgadgets";

    // List of all Thing Type UIDs
    public final static ThingTypeUID THING_TYPE_TURRISGADGETS = new
ThingTypeUID(BINDING_ID, "turrisdongle");

    // List of all Channel ids
    public final static String CHANNEL_PGX = "pgx";
    public final static String CHANNEL_PGY = "pgy";
    public final static String CHANNEL_ALARM = "alarm";
    public final static String CHANNEL_BEEP = "beep";

}
```

2. **TurrisGadgetsHandlerFactory -** tato třída vytváří ThingHandler instance. Každá vazba musí implementovat ThingHandlerFactory a registrovat jej jako OSGi službu tak, aby mohla být využita pro vytváření a manipulaci s Things. Pokud přidáváme novou Things, vždy nám ThingHandlerFactory zobrazí metodu supportsThingType. Tuto třídu nebylo nutné upravovat.

```
public class TurrisGadgetsHandlerFactory extends BaseThingHandlerFactory {

    private final static Set<ThingTypeUID> SUPPORTED_THING_TYPES_UIDS =
Collections.singleton(THING_TYPE_TURRISGADGETS);

    @Override
    public boolean supportsThingType(ThingTypeUID thingTypeUID) {
        return SUPPORTED_THING_TYPES_UIDS.contains(thingTypeUID);
    }

    @Override
    protected ThingHandler createHandler(Thing thing) {

        ThingTypeUID thingTypeUID = thing.getThingTypeUID();

        if (thingTypeUID.equals(THING_TYPE_TURRISGADGETS)) {
            return new TurrisGadgetsHandler(thing);
        }

        return null;
    }
```

3. **TurrisGadgetsHandler -** je hlavní třída projektu, která se stará o překládání příkazů mezi OpenHABem a externím zařízením a naopak. Skládá se ze dvou metod, první je `handleCommand`, která nejdříve zjistí, zda jsou všechny kanály zapnuté nebo vypnuté a dále zjišťuje, zda nějaký kanál změnil svůj stav a uloží hodnoty proměnných do řetězce `stateMessage`. Následně se příkaz odešle na zařízení.

```
public class TurrisGadgetsHandler extends BaseThingHandler {

    private static Logger logger =
LoggerFactory.getLogger(TurrisGadgetsHandler.class);

    private String portName;
    static int BAUD = 57600;
    private OutputStreamWriter output;
    private char ESC = (char) 27;
    private char LN = (char) 10;
    public int PGYState = 0;;
    private int PGXState = 0;
    private int AlarmState = 0;;
    private String BEEPState = "NONE";
    private String stateMessage;
    static NRSerialPort serialPort;
    static OutputStream outputStream;

    public TurrisGadgetsHandler(Thing thing) {
        super(thing);
    }

    @Override
    public void handleCommand(ChannelUID channelUID, Command command) {
        if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_PGY) &&
(command instanceof OnOffType)) {
            if (command == OnOffType.ON) {
                PGYState = 1;
            } else if (command == OnOffType.OFF) {
                PGYState = 0;
            }

        }

        if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_PGX) &&
(command instanceof OnOffType)) {
            if (command == OnOffType.ON) {
                PGXState = 1;
            } else if (command == OnOffType.OFF) {
                PGXState = 0;
            }
        }

        if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_BEEP) &&
(command instanceof OnOffType)) {
            if (command == OnOffType.ON) {
                BEEPState = "FAST";
            } else if (command == OnOffType.OFF) {
```

```
                BEEPState = "NONE";
            }
        }

        if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_ALARM) &&
(command instanceof OnOffType)) {
            if (command == OnOffType.ON) {
                AlarmState = 1;
            } else if (command == OnOffType.OFF) {
                AlarmState = 0;
            }
        }
        stateMessage = ESC + "TX ENROLL:0 PGX:" + PGXState + " PGY:" +
PGYState + " ALARM:" + AlarmState + " BEEP:"
                + BEEPState + LN;
        try {
            if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_PGX)
                    && (command instanceof OnOffType)) {
                if (command == OnOffType.ON) {
                    logger.info(stateMessage);
                    output.write(stateMessage);

                } else if (command == OnOffType.OFF) {
                    logger.info(stateMessage);
                    output.write(stateMessage);

                }
                updateState(channelUID, (OnOffType) command);
            }

            if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_PGY)
                    && (command instanceof OnOffType)) {
                if (command == OnOffType.ON) {
                    logger.info(stateMessage);
                    output.write(stateMessage);

                } else if (command == OnOffType.OFF) {
                    logger.info(stateMessage);
                    output.write(stateMessage);

                    updateState(channelUID, (OnOffType) command);
                }

                if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_BEEP)
                        && (command instanceof OnOffType)) {
                    if (command == OnOffType.ON) {
                        logger.info(stateMessage);
                        output.write(stateMessage);

                    } else if (command == OnOffType.OFF) {
                        logger.info(stateMessage);
                        output.write(stateMessage);
                    }
                    updateState(channelUID, (OnOffType) command);
                }
                if
(channelUID.getId().equals(TurrisGadgetsBindingConstants.CHANNEL_ALARM)
```

```
                    && (command instanceof OnOffType)) {
                if (command == OnOffType.ON) {
                    logger.info(stateMessage);
                    output.write(stateMessage);

                } else if (command == OnOffType.OFF) {
                    logger.info(stateMessage);
                    output.write(stateMessage);

                }
                updateState(channelUID, (OnOffType) command);
            }
            output.flush();
        }
    } catch (IOException e) {

        e.printStackTrace();
    }
}
```

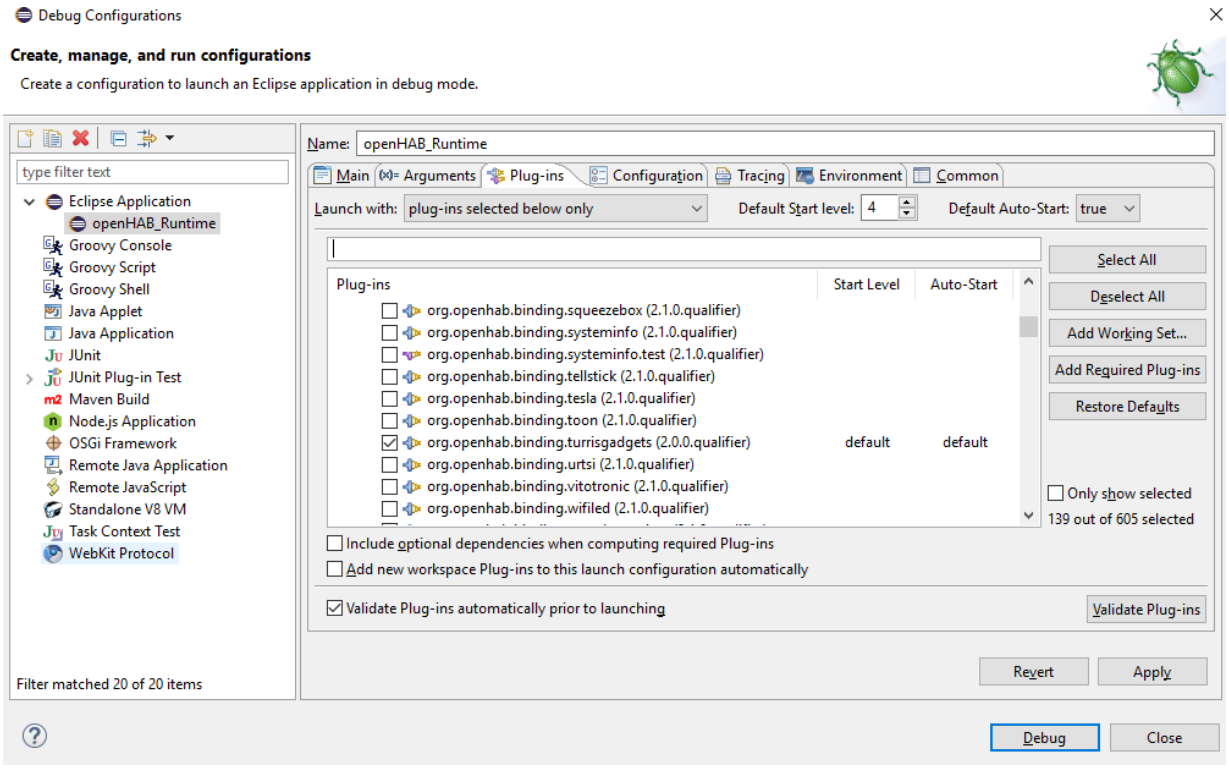
Druhá metoda se nazývá initialize a je vykonána při spuštění načtení vazby. Z nastavení Things zjistí port připojeného Turris Donglu, v mém případě se jedná o /dev/ttyUSB0 a naváže s ním komunikaci. V případě, že je port připravený navázat spojení, změní se stav ThingStatus na ONLINE.

```
@Override
    public void initialize() {
        portName = (String) getThing().getConfiguration().get("port");
        if (portName != null) {
            serialPort = new NRSerialPort(portName, BAUD);
            if (serialPort.connect()) {
                updateStatus(ThingStatus.ONLINE);
                output = new
OutputStreamWriter(serialPort.getOutputStream());
            } else {
                updateStatus(ThingStatus.OFFLINE,
ThingStatusDetail.COMMUNICATION_ERROR,
                        "Failed to connect to serial port " + portName);
                logger.debug("Failed to connect to serial port " +
portName);
            }
        } else {
            updateStatus(ThingStatus.OFFLINE,
ThingStatusDetail.CONFIGURATION_ERROR, "Serial port name not configured");
            logger.debug("Serial port name not configured");
        }

    }
}
```

### 4.5.7 Testování vazby

Abychom mohli otestovat funkčnost vazby zvolíme v Eclipse IDE v horním menu položku Debug configuration. Zde zvolíme OpenHAB_runtime a vybereme ze seznamu naši vazbu a dáme Debug.




Obrázek 7: Export vazby

Následně dojde ke spuštění OpenHABu a naší vazby. Průběh spouštění můžeme sledovat v záložce Console, kde jsou vypisovány údaje z logů aplikace. Nejdůležitější je pro nás zápis, zda byla načtena naše vazba a následné přepnutí stavu things `turrisgadgets` na ONLINE.

Nyní otevřeme internetový prohlížeč a zadáme adresu https://localhost:8080. Vybereme jedno z uživatelských prostředí a vyzkoušíme spínání zásuvky.

### 4.5.8 Export vazby

Aby bylo možné vytvořenou vazbu použít v OpenHABu nainstalovaném na Rapsberry Pi je ji nejprve potřeba vyexportovat z vývojového prostředí. V horním menu zvolíme `File -> Export -> Plug-in Development` následně vybere název naší vazby, místo, kam bude uložena a zvolíme `Finish`. Tím se v zadaném umístění vytvoří soubor s koncovkou .jar. Ten přesuneme na Raspsberry a vložíme do adresáře `/usr/share/openhab2/addons`.

Konfigurační soubory OpenHABu jsou umístěny v `/etc/openhab2`, ty upravíme stejně jako v odstavci 4.5.2 Příprava OpenHABu.

# 5 Závěr

Rozvoj inteligentních technologií v zaznamenal v posledních letech nebývalý růst. Vzhledem k velkému potenciálu tohoto odvětví, jsem se rozhodl vytvořit projekt se zásuvkou Jablotron a řídícím systémem OpenHAB. Tento projekt se stal podkladem pro tuto bakalářskou práci. Základem bylo nastudování jednotlivých pramenů, ze kterých se vazba mezi inteligentním zařízením a řídícím systémem skládala, dále bylo nutné sehnat potřebná zařízení, pro které byla tato vazba vytvořena.

Cílem mé bakalářské práce bylo vytvořit a následně popsat vazbu mezi inteligentním zařízením, zásuvkou Jablotron AC -88 a řídícím systémem OpenHAB, k jehož zprovoznění jsem využil řídící jednotku Raspberry Pi 2, která je díky svým vlastnostem úspornější než běžný notebook nebo počítač. Celý projekt vychází z teoretických znalostí jednotlivých částí inteligentních zařízení, známých také jako Turris Gadgets, řídícího systému OpenHAB od jeho instalace, přes uživatelská prostředí, rozhraní až po jeho spárování s inteligentním zařízením. Rapsberry Pi 2 sloužila jako řídící jednotka pro systém OpenHAB a zajišťovala tak fungování celého projektu.

Celý projekt, tedy vytvoření vazby mezi inteligentním zařízením a řídící jednotkou je možné dále rozšiřovat a přizpůsobovat pro funkce inteligentních domů tak, aby uspokojil široké spektrum uživatelů.

Vytvořená vazba je schopná ovládat dvě zásuvky Jablotron AC-88 nebo dvě dálkově spínaná relé AC-82 a interní sirénu JA-80L. Funkčnost řešení je závislá na správné konfiguraci OpenHABu a Turris Donglu, v OpenHABu je nutné definovat USB port, do kterého je Turris Dongle připojen. Konfigurace zařízení se systémem OpenHAB není pro běžného uživatele se základními znalostmi této problematiky náročná. Samotná tvorba vazby však vyžaduje i znalost programovacího jazyka JAVA, tím se stává řešení složitějším. Při tvorbě vazby se mohou objevit různé problémy spojené s komunikací mezi Turris Donglem a OpenHABem. V mém případě se jednalo o problém s přístupovým bodem /dev/ttyUSB0, který nebyl vlastněn uživatelem OpenHAB a při ručním nastavení majitele příkazem chmod openhab /dev/ttyUSB0 vše fungovalo, až do restartování ovládací platformy nebo odpojení Turris Donglu. Řešením se stalo trvalé nastavení majitele pravidlem v udev.

# 6 Citovaná literatura


1. Ashton, K. (22. Červen 2009). rfidjournal.com. Načteno z RFID Journal: http://www.rfidjournal.com/articles/pdf?4986
2. Bip Philippe Gitbooks. (2015). Načteno z bipphlippe.gitbooks.io: https://bipphilippe.gitbooks.io/openhab2-compendium-old/content/Things_items_and_Co/things,items_and_co_md.html
3. E. A. Kosmatos, N. T. (2011). Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture. Advances in Internet of Things: Scientific Research, str. http://dx.doi.org/10.4236/ait.2011.110022.
4. Eclipse IoT project. (2016). Eclipse SmartHome. Načteno z Eclipse.org: https://www.eclipse.org/smarthome
5. Engelen, K. K. (2012). Home Automation for Geeks. Načteno z http://de.slideshare.
6. Chytrá instalace. (2015). Načteno z chytrainstalace.cz: http://www.chytrainstalace.cz/chytra-elektroinstalace/
7. Ing. Pavel Burian, C. (2014). Internet inteligentních aktivit. Praha: Grada Publishing .
8. Kai Kreuzer, A. B.-E. (2017). OpenHAB empowering the smart home. Načteno z docs.openhab.org: http://docs.openhab.org/
9. Murer, R. (2010). Načteno z The Internet of Things: http://www.theinternetofthings.eu/content/internet-things-%E2%80%93-fundamentals-ricardo-murer
10. Nunberg, G. (2012). The Advent of the Internet. Courses.
11. OpenHAB. (2011). Empowering the smart home. str. http://www.openhab.org/.
12. Postscapes. (2016). Načteno z postscapes.com: : http://postscapes.com/internet-of-things-technologies#communication
13. Raspberry Pi 2. (2015). Načteno z https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

14. Slideshare. (2012). Načteno z de.slideshare.net: http.://de.slideshare.net/xthirtynine/open-hab-devoxx-2012

15. Svět hardware. (2016). Načteno z svethardware.cz: http://www.svethardware.cz/internet-of-things-propojena-budoucnost/39560

16. Trtík, J. (2009). Návrh elektroinstalace rodinného domu s využitím Inteligentních prvků. Brno: Bachelor Thesis, fakulta elektrotechniky a komunikačních technologií VUT v Brně.

17. Turan, P. (2010). Inteligentní RD 1. Načteno z DSpace UTB: http://dspace.k.utb.cz/bitstream/handle/10563/12066/turan_2010_dp.pdf?sequenc

18. turris.cz. (2016). Načteno z TURRIS:GADGETS: www.turris.cz/gadgets/start

# 7 Seznam obrázků