VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

INTEGRACE SENZORŮ A AKTORŮ DO SYSTÉMU BEEEON

INTEGRATION OF SENSORS AND ACTORS INTO THE BEEEON SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DAVID BEDNAŘÍK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN VIKTORIN**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Bednařík David**
Obor: Informační technologie
Téma: **Integrace senzorů a aktorů do systému BeeeOn**
Integration of Sensors and Actors into the BeeeOn System
Kategorie: Vestavěné systémy

Pokyny:

1. Nastudujte zařízení podporující technologie Belkin Wemo, Philips Hue, BeeeWi.
2. Seznamte se s architekturou brány systému BeeeOn.
3. Navrhněte způsob integrace vybraných technologií do systému BeeeOn.
4. Proveďte implementaci navrženého řešení a ověřte funkčnost na vhodném případu užití.
5. Diskutujte dosažené výsledky a možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Viktorin Jan, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

V této bakalářské práci bylo úkolem se seznámit s komponenty chytré domácnosti od výrobců Belkin WeMo, Philips Hue a BeeWi, nastudovat jejich komunikační protokol a vytvořit moduly do systému BeeeOn pro jejich ovládání. V teoretické části práce je popsané, jak funguje samotný systém BeeeOn a jak se chovají a komunikují jednotlivá zařízení od zmíněných výrobců. V praktické části je zmíněn způsob rozšíření aplikace BeeeOn Gateway o moduly pro řízení chytrých prvků domácnosti. Součásti této části je také způsob testování a ověřování správnosti implementace.

Abstract

In this bachelor thesis, the task was to acquaint with components of smart home from these vendors Belkin WeMo, Philips Hue, BeeWi, to study their communication protocol and create modules to BeeeOn system to control them. In the theoretical part of thesis is described how the BeeeOn system works and how the single device from mentioned manufacturers behaves and communicates. In the practical part, it is mentioned the way of extension of BeeeOn Gateway application to modules for control the smart home devices. This part also includes the methods of testing and verification of accuracy of implementation.

Klíčová slova

Inteligentní domácnost, BeeeOn, Belkin WeMo, Philips Hue, BeeWi, BeeeOn Gateway

Keywords

Smart Home, BeeeOn, Belkin WeMo, Philips Hue, BeeWi, BeeeOn Gateway

Citace

BEDNAŘÍK, David. *Integrace senzorů a aktorů do systému BeeeOn*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Viktorin

Integrace senzorů a aktorů do systému BeeeOn

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Viktorina. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Bednařík
7. května 2018

Poděkování

Zde bych rád poděkoval vedoucímu bakalařské práce Ing. Janu Viktorinovi za odborné vedení a konzultace jak při psaní této práce, tak při návrhu a implementaci praktické části práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Od počátku lidstva se člověk snaží svojí práci co nejvíce minimalizovat či ji úplně nahradit stroji, což je způsobeno touhou člověka objevovat nové věci a zjednodušit si práci. Toto je důvodem neustálého vývoje nových technologií. Není tomu jinak ani u chytré domácnosti, kde je podobně hlavním účelem ulehčit člověku život. Příkladem může být uvaření kávy, rozsvícení, naladění intenzity a barvy žárovky a zapnutí kotle na příjemných $24\,°C$ z pohodlí křesla v obývacím pokoji za pomocí mobilního zařízení. Takovéto ovládání chytrých zařízení je možné odkudkoliv, stačí pouze připojení k Internetu. Tato možnost může být velice přívětivá i v případě připojení žehličky do elektrické sítě. Velmi často se stává, že odejdeme z domova ve spěchu a nejsme si jisti, jestli jsme vypnuli žehličku. Když žehličku připojíme do sítě skrz chytrou zásuvku můžeme jednoduše odkudkoliv zjistit stav zásuvky, a tudíž zda je žehlička vypnuta nebo zapnuta. Další služby, které chytrá domácnost může nabídnout jsou například zabezpečení pomocí pohybových čidel a kamer proti zlodějům. Ale také umožňuje snížit náklady na chod domácnosti, kdy kotel běží pouze když platí určitá podmínka, například že teplota klesla pod určitou hodnotu, kterou získáme z teplotních čidel.

Realizací chytré domácnosti se zabývá projekt *BeeeOn*, který vyvíjí studenti a odborní pracovníci na Vysokém učení technickém, konkrétně na Fakultě informačních technologií v Brně. Hlavním cílem systému *BeeeOn* je poskytnout podporu velké škále zařízení výrobců třetích stran a jednoduché rozšiřitelnosti o nová zařízení. Hlavním cílem této práce je rozšířit projekt *BeeeOn* o další zařízení a to konkrétně chytrou zásuvku, žárovku a stmívač od firmy Belkin WeMo, chytrou žárovku od výrobce Philips Hue a také o chytrou zásuvku a žárovku vyráběnou společnosti BeeWi.

Obsah práce je strukturován do několika částí. První část popisuje systém *BeeeOn* jako celek. Účelem kapitoly je ukázat z jakých částí se systém skládá, co která část má za úkol a jaké mají vazby jednotlivé komponenty mezi sebou. Konkrétnější důraz je kladen na *BeeeOn Gateway*, protože tato část systému *BeeeOn* je rámci této práce modifikována.

Další část obsahuje popis jednotlivých chytrých zařízení. Součástí popisu daného zařízení jsou následující informace:

- Funkcionalita nabízená zařízeními.
- Způsob lokalizace zařízení v síti.
- Konkrétní komunikace s daným zařízením a způsob ovlivňování jeho chování a stavu.

Ve čtvrté a páté kapitole se zaměříme na návrh a implementaci modulů do *BeeeOn Gateway*, které zajistí řízení senzorů a aktorů. Řízením je myšleno hledání nových zaří-

zení, naparování zařízení, odpárování zařízení, změna stavu konkretního zařízení a zjištění aktuálního stavu zařízení.

Na závěr je ukázaný způsob, jak se ověřovala správnost implementace. Testování se provádělo převážně manuálně pomocí mobilní aplikace a pomocí testovacího modulu implementovaného v bráně. Součástí testování jsou také jednotkové testy, které zjišťují správnou funkčnost vybraných komponent.

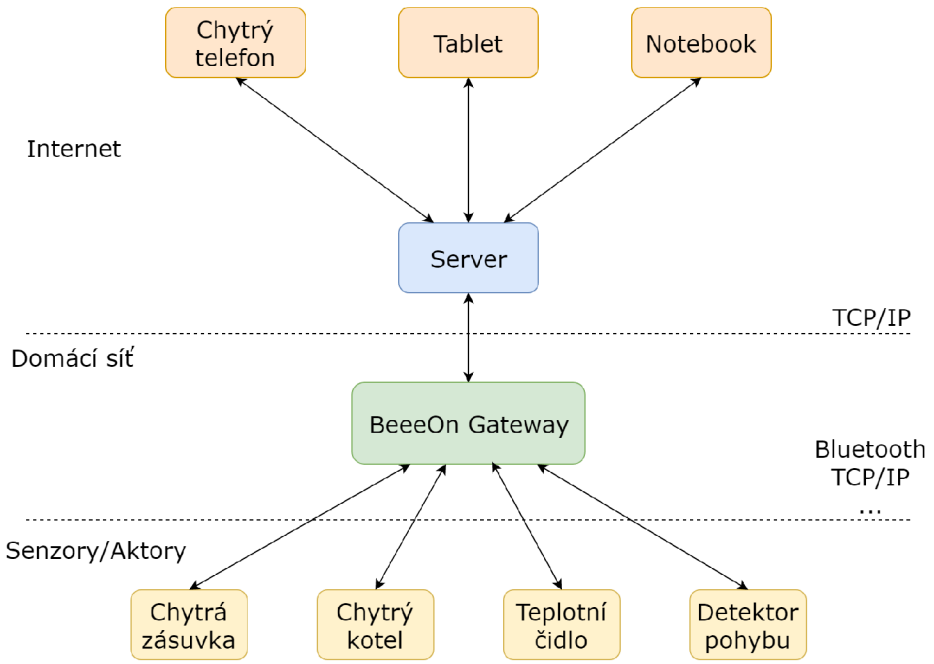
Kapitola 2

Systém BeeeOn

Předmětem kapitoly je architektura systému *BeeeOn*. Hlavní službou, kterou systém nabízí, jsou funkce inteligentní domácnosti. Tyto vlastnosti inteligentní domácnosti realizuje pomocí podpory velkého počtu chytrých zařízení od různých výrobců. Také se snaží sjednotit rozhraní k ovládání prvků inteligentní domácnosti, a to z jednoduchého důvodu. Každý výrobce vytváří vlastní aplikace k řízení svých zařízení, a pokud člověk chce mít výrobky od různých výrobců, musel by mít také velké množství aplikací pro jejich řízení, což je značně nekomfortní. Jak jsem již naznačil v úvodu, jsou zde představeny komponenty, které systém obsahuje a jejich vzájemné vztahy.

2.1 Architektura

Na obrázku 2.1 je znázorněno schéma architektury. Jsou zde zobrazeny části systému a jejich vazby. Vztahy mezi prvky jsou vizualizovány pomocí šipek, které také udávají směr komunikace.



Obrázek 2.1: Architektura systému BeeeOn

Na nejvyšší vrstvě jsou uživatelská rozhraní, která komunikují pomocí Internetu se serverem, což jim umožňuje zasílání požadavků na změnu stavu daného zařízení. Na druhou stranu rozhraní dokáže získávat stav určitého senzoru. Na další vrstvě máme server, který jak už bylo řečeno, vyřizuje požadavky od uživatelů a to díky propojení s bránou, jež je taktéž prováděno skrz počítačovou síť. S bránou také komunikují koncová zařízení (senzory, aktory). Zde je komunikace prováděna již za pomocí různých protokolů a technologií. Úkolem brány je sbírat senzorová data, posílat je na server a zároveň také přijímat příkazy od serveru a nastavovat stav zařízení.

2.2 Uživatelská rozhraní

Uživatelské rozhraní je hraniční prvek určený pro vstup uživatele do systému. Vstupem je míněno ovládání akčních částí systému. Hlavním cílem je umožnit uživateli manipulaci s jeho bránami a koncovými zařazeními. Nedílnou součástí je také vizualizace nasbíraných senzorových dat ve formě grafů.

Primárním uživatelským rozhraním je mobilní aplikace pro chytré telefony s operačním systémem *Google Android*. Nabízí uživateli všechny výše zmíněné funkce (správa bran, správa zařízení, zobrazení naměřených senzorových dat a nastavování stavu zařízení).

Jako další je vhodné zmínit webové rozhraní, které je ale zatím ve vývojové fázi. Hlavní výhodou oproti mobilní aplikace je v tloušťce klienta. Webovému rozhraní postačí velmi tenký klient, protože pro běh vyžaduje pouhý internetový prohlížeč, který je v dnešní době k nalezení relativně na každém zařízení.

2.3 Server

Server slouží jako prostředník v komunikaci mezi uživatelem a jeho bránou. Je důležitým prvkem systému, protože zajišťuje správu chytré domácnosti odkudkoliv z Internetu. Jeho úkolem je řídit správu uživatelů, správu připojených bran, správu senzorů, aktorů a ukládat senzorová data. Správa zahrnuje registraci uživatelů a zařízení, ale také odregistraci brány či senzorů nebo aktorů.

Architektura serveru je rozdělena do tří vrstev za účelem zjednodušení testování a rozšiřitelnosti. Mezi tyto vrstvy patří aplikační, servisní a datová vrstva.

Aplikační vrstva zajišťuje komunikaci jak s uživateli tak s bránami. V prvním kroku při vytváření spojení mezi bránou a serverem musí brána navázat toto spojení. Spojení musí být udržováno, protože brána může být zastíněná překladem adres (NAT). Toto zastínění způsobí, že samotný server nemůže navázat spojení s bránou. Udržování spojení umožňuje odesílání požadavků na změnu stavu aktorů, ale také tímto dokáže server detekovat výpadek brány a informovat o tom uživatele.

Další vrstvou je servisní vrstva, která implementuje vlastní logiku serveru. Pracuje tím způsobem, že vytvoří transakci pro každou servisní operaci. Ověří přístupová práva a následně kontaktuje datovou vrstvu, která vrátí požadovaná data. Servisní vrstva provede požadované operace nad daty a modifikovaná data předá zpět datové vrstvě, která je uloží. Na závěr se transakce potvrdí.

Nejnižší vrstvou je datová, která odstiňuje databázi od zbytku serveru. To zajišťuje, že logika serveru je nezávislá na použité databázi. Aktuálně je pro uložení dat použita databáze *PostgreSQL*.

2.4 Koncová zařízení

Pomocí koncových zařízení se zajišťují vlastnosti inteligentní domácnosti. Jedná se o fyzická zařízení, které nabízí určité služby. Může se skládat z více části, které jsou označovány jako moduly zařízení. Existují dva typy modulů:

- **Senzory** – modul, který měří většinou fyzikální veličiny. Příkladem může být teplota, vlhkost, obsah CO_2 ve vzduchu nebo atmosferický tlak. Pro tyto moduly je možné nastavovat četnost sběru naměřených hodnot.
- **Aktory** – na rozdíl od senzorů se nejedná o pasivní prvky, které pouze něco měří a odesílají. Tyto moduly dovolují měnit svůj stav, což může být například svítí/nesvítí, topí/netopí nebo zapnut/vypnut. Zařízení, která obsahují tyto moduly jsou například inteligentní zásuvky nebo žárovky.

Zařízení taktéž může obsahovat kombinaci těchto modulů. Názorným příkladem je regulátor VPT, který slouží k řízení topné soustavy a je implementována jeho podpora v projektu *BeeeOn*. Skládá se z několika modulů, jak senzorů, tak aktorů. Obsahuje teplotní čidla pro měření teploty, ale také spínač kotle, kterým se určuje zda se má topit.

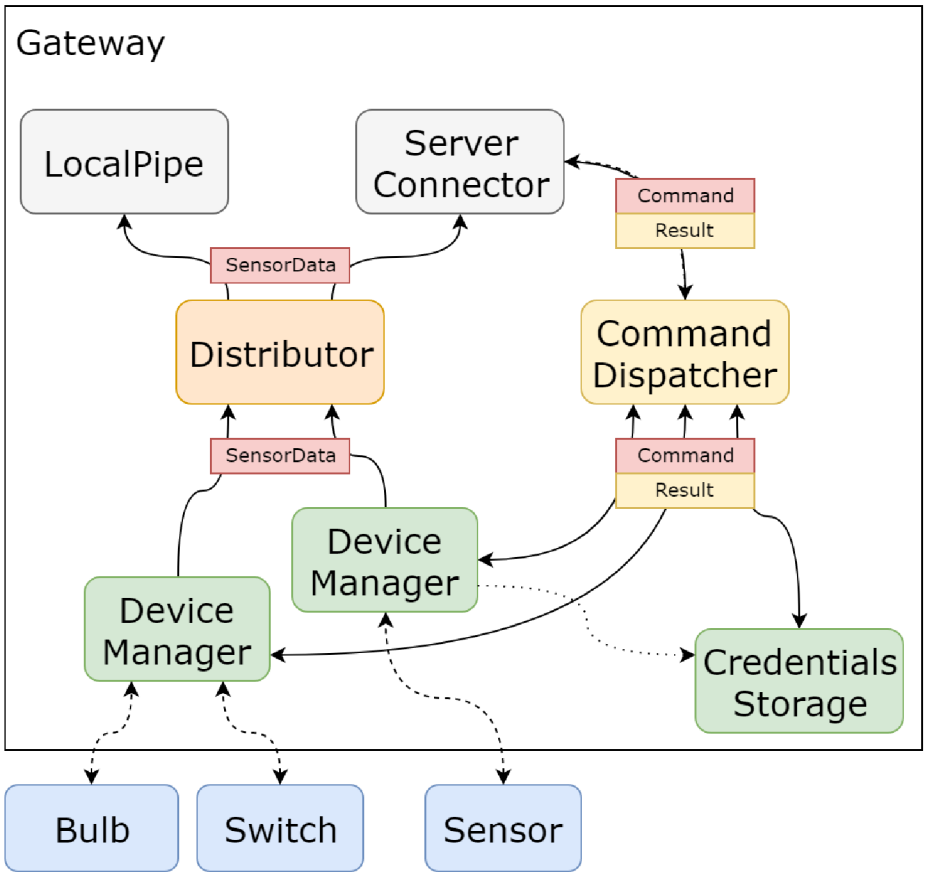
Součástí projektu *BeeeOn* byly také vyvinuty vlastní senzory, které nabízí měření teploty a vlhkosti. Brána obsahuje i senzor pro měření atmosferického tlaku.

2.5 Brána

Brána je určena k připojování různých zařízení do systému a jejich ovládání. Hlavním úkolem je sběr senzorových dat v určitých periodách a jejich exportování na server. Dokáže také přijímat od serveru příkazy na ovládání akčních prvků systému. Aplikace brány běží na open source hardware, konkrétně na vývojové desce A10-OLinuXino-LIME od výrobce Olimex. Desku tvoří 1 GHz Cortex-A8 procesor, 512 MB operační paměti, 100 Mbit Ethernet konektor a 160 GPIO pinů [1]. Součástí jsou také konektory pro komunikaci s dalšími periferiemi, mezi které patří SATA, HDMI a USB konektor. Pomocí pinů pro všeobecné použití je připojena rozšiřující deska obsahující rádio MRF89XA od společnost Microchip, která slouží ke komunikaci s *BeeeOn senzory*. Stejným způsobem je také připojeno čidlo pro měření atmosferického tlaku. Použitým operačním systémem, běžícím na desce, je linuxová distribuce sestavena pomocí překladového systému OpenEmbedded.

Obrázek 2.2: BeeeOn brána

Jádrem brány je program *BeeeOn Gateway*, který se stará, jak o komunikaci se serverem, tak s připojenými senzory. Architektura aplikace je znázorněna na obrázku 2.3. Z obrázku lze zjistit, z jakých objektů se program skládá, a možnou komunikaci mezi objekty. Objekty zde představují třídy programovacího jazyka *C++*.



Obrázek 2.3: Architektura brány

O sběr dat se starají manažeři zařízení (*DeviceManager*). Pro každého podporovaného výrobce inteligentních prvků domácnosti je vytvořena specifická implementace třídy *DeviceManager*, který se stará o správu těchto zařízení. Každá implementace třídy *DeviceManager*:

- řeší technologicky-specifickou komunikaci z připojených zařízení (senzorů)
- vyhledává a páruje dostupná zařízení na základě požadavků/příkazů z řídicího serveru
- definuje sadu konfigurovatelných vlastností dané technologie

Jedním z úkolů manažera zařízení je získávat data ze zařízení. V okamžiku přijetí senzorových dat se data transformují do unifikované podoby. Kromě naměřených hodnot je nutné znát identifikátor zařízení a číslo modulu. O každém zařízení je potřeba znát moduly, ze kterých se skládá, aby je bylo možné rozlišit. V aplikaci jsou tyto data reprezentována třídou *SensorData*.

Identifikátor zařízení je 64 bitové číslo, kde prvních 8 bitů je identifikátor manažera zařízení a zbytek je definován na základě určité vlastnosti zařízení (například MAC adresa).

Následně se vytvořená *SensorData* předají objektu *Distributor*, který je přepošle všem registrovaným exportérům. Mezi exportéry patří například unixová roura (*LocalPipe*) nebo klient pro komunikaci se serverem (*ServerConnector*). Komunikace mezi bránou a serverem je vysvětlená v sekci 2.5.1, kde jsou také popsány příkazy, které může server vyžadovat od brány a naopak.

Dalším případem komunikace brána-server je zasílání příkazů (*Command*). Tato komunikace je asynchronního typu, což znamená, že odesílatel příkazu není blokován čekáním na odpověď. Při odeslání příkazu serverem přijímá objekt *ServerConnector* na bráně tuto zprávu a předává ji objektu *CommandDispatcher*, který rozhodne jakým příjemcům zpráva náleží. Příjemcem může být libovolný objekt implementující rozhraní *CommandHandler*. Ke každému příkazu je potřeba vytvořit výsledek operace (*Result*), který definuje úspěšnost operace. Když odesílá požadavek brána na server, může například požadovat seznam napárovaných zařízení. Pak je tento příkaz předán opět objektu *CommandDispatcher*, který zprávu pošle objektu *ServerConnector* a ten se již postará o doručení požadavku serveru.

Pro případ potřeby uložení citlivých informací, jako uživatelské jméno a heslo, potřebné k ovládání inteligentních prvků, slouží třída *CredentialsStorage*. Objekt třídy *CredentialsStorage* je využíván manažery zařízení ke vkládání a čtení dat z úložiště chráněného proti neoprávněnému přístupu.

2.5.1 Komunikace server brána

Pro komunikaci mezi bránou a serverem je vytvořený zabezpečený kanál, kterým se posílají požadavky a odpovědi. Zprávy jsou přenášeny po síti v textovém kódováni JSON (JavaScript Object Notation). Všechny typy zpráv jsou vysvětleny v následujícím seznamu:

1. **Registrace brány** (gateway-register) – zpráva, která je zasílaná bránou po navázaní spojení se serverem za účelem její registrace na serveru.

2. **Seznam napárovaných zařízení** (device-list) – následně po startu brány a vytvoření manažerů zařízení, je velmi žádoucí zachovat konzistenci v systému *BeeeOn*. Při restartu nebo ukončení a opětovném spuštění brány jsou předchozí data uložena v operační paměti nenávratně ztracena. Mezi těmito daty jsou také seznamy napárovaných zařízení každého manažera zařízení, což znamená, že při startu brány jsou jejich seznamy prázdné. Tento problém vyřeší tato zpráva, která je odeslaná každým manažerem zařízení při startu. Odpovědí serveru na tento požadavek je seznam všech již napárovaných zařízení náležících danému manažerovi zařízení.

3. **Poslední hodnota zařízení** (last-value) – pomocí této zprávy se řeší podobný problém jako je tomu v minulé zprávě. Příkladem problému, který tento typ příkazu řeší může být například připojená chytrá žárovka k bráně, která je aktuálně ve stavu vypnuta. Neočekávaně vypadne dodávka elektřiny, která se opět obnoví, což způsobí zapnutí žárovky, která si ale nepamatuje svůj původní stav a je naprogramovaná se inicializovat se stavem zapnuta. Tudíž při startu brány a získání všech již napárovaných zařízení, je možné se dotázat na poslední stav každého zařízení a ten upravit na fyzickém zařízení.

4. **Párování** (listen) – požadavek odeslaný serverem na základě stisknutí tlačítka v uživatelském rozhraní uživatelem, který vyžaduje od systému vyhledání nových zařízení. Tento příkaz je určen všem manažerům zařízení. Zpráva obsahuje dobu trvání definující jak dlouho se má hledání nových zařízení provádět. Pro každé nově nalezené

zařízení posílá manažer zařízení zprávu informující o této události server, který zobrazí nově nalezené zařízení uživateli.

5. **Napárování zařízení** (device-accept) – aby mohl senzor posílat data na server nebo aby mohl být aktor ovládán uživatelem, je potřeba dané zařízení napárovat. K tomuto případu slouží tento příkaz. Když uživatel spustí párování, postupně se mu budou ukazovat nově nalezená zařízení. Následně si vybere, které zařízení chce přijmout a klikne na něj. Tím se odešle ze serveru tato zpráva na bránu a ta jej doručí správnému manažerovi zařízení, který se určí na základě posledních 8 bitů identifikátoru přijímaného zařízení. Operace je úspěšná, když je zařízení korektně napárováno. V případě, že manažer zařízení nevlastní záznam o zařízení obsaženého ve zprávě, končí příkaz chybou `FAILED` odeslanou na server ve formě objektu *Result*.

6. **Odpárování zařízení** (unpair) – jedná se o zprávu, která má opačný efekt než-li předchozí. V případě kdy uživatel již nechce odebírat data nebo ovládat dané zařízení, tak jej odebere (odpáruje) v uživatelském rozhraní a to způsobí odeslání tohoto příkazu ze serveru na bránu. Způsob výběru manažeru zařízení a vyhodnocení výsledku operace je stejná jako v předchozím případě.

7. **Nastavení hodnoty zařízení** (set-value) – účelem příkazu je umožnit uživateli nastavovat stav akčních prvků (aktorů) v systému. Obsahem zprávy je identifikátor zařízení, číslo modulu a hodnota, která určuje nový stav zařízení. Jakmile server přijme požadavek od uživatelského rozhraní na nastavení hodnoty, přepošle ho na bránu. Po úspěšném přijetí příkazu bránou je příkaz předán správnému manažerovi zařízení podle předaného ID zařízení. Manažer zařízení se snaží dané zařízení nastavit na požadovanou hodnotu. Nastavování může selhat pokud je dané zařízení například nedostupné. Pro tento případ končí operace chybou `FAILED`, která je předána serveru v podobě objektu *Result*. Doba, za kterou je nastavení hodnoty úspěšně provedeno, není často předem známa. Pokud tato doba překročí určitou hranici definovanou jako timeout, tak je nastavování považováno za neúspěšné a ukončeno taktéž chybou `FAILED`.

2.5.2 Sestavování aplikace

Aplikace se skládá z několika modulů mající mezi sebou určité závislosti, které již byli dříve popsány v 2.1. Aby nebylo nutné provést překlad aplikace pro každou možnou konfiguraci, je použita technika vkládání závislostí (Dependency injection). Příkladem může být, že je vyžadováno exportovat data pouze pomocí určitého exportéru. To lze provést velmi jednoduše díky této technice. Postačí pouze upravit konfigurační soubor, kde požadovaný exportér povolíme a ostatní zakážeme. Stejným způsobem se dají upravovat parametry objektů.

Po spuštění aplikace se jako první spustí poskytovatel vkládání závislostí (*DependencyInjector*), který načte konfigurační soubory a na základě toho vytvoří objekty a jejich závislosti. Příklad konfiguračního souboru je možné vidět v ukázce 2.1, který konfiguruje start aplikace v zjednodušené formě než-li je tomu na obrázku 2.3. Rozdílem je spouštění pouze jednoho manažera zařízení, jednoho exportéru a vynechání úložiště pro přihlašovací údaje.

```
<system>
  <factory>
```

```
    <instance name="main" class="BeeeOn::LoopRunner">
      <add name="loops" ref="gwServerConnector"/>
      <add name="runnables" ref="deviceManager"/>
    </instance>
    <instance name="distributor" class="BeeeOn::BasicDistributor">
      <add name="exporters" ref="gwServerConnector"/>
    </instance>
    <instance name="commandDispatcher" class="BeeeOn::PocoCommandDispatcher">
      <add name="handlers" ref="gwServerConnector"/>
      <add name="handlers" ref="deviceManager"/>
    </instance>
    <instance name="gwServerConnector" class="BeeeOn::GWServerConnector">
      <set name="commandDispatcher" ref="commandDispatcher"/>
    </instance>
    <instance name="deviceManager" class="BeeeOn::DeviceManager">
      <set name="distributor" ref="distributor"/>
      <set name="commandDispatcher" ref="commandDispatcher"/>
    </instance>
  </factory>
</system>
```

Výpis 2.1: Ukázka konfiguračního souboru

Kapitola 3

Bezdrátová chytrá zařízení

V této kapitole jsou představeny chytré prvky domácnosti, pro které je v rámci této práce vytvářená podpora do systému *BeeeOn*. Zaměření je vesměs na akční prvky, které je možné ovládat. Mezi tyto prvky patří zásuvky, žárovky a vypínače osvětlení. Analyzují se zde zařízení od tři různých společností:

- Belkin WeMo
- Philips Hue
- BeeWi

U každého ze zmíněných výrobců se setkáváme s jiným kódováním zpráv a použitou technologií pro komunikaci. Belkin WeMo a Philips Hue používají pro ovládání prvků počítačovou síť. Výjimku tvoří inteligentní žárovka, kde komunikace probíhá skrz prostředníka, se kterým ovládací stanice komunikuje pomocí počítačové sítě a prostředník následně zprávy přeposílá dané žárovce pomocí ZigBee. Poslední výrobce BeeWi využívá technologii Bluetooth.

3.1 Belkin WeMo

Jak již bylo naznačeno, komunikace se zařízeními je v případě Belkin WeMo prováděna skrz počítačovou síť s využitím WiFi. Protože se jedná o bezdrátové prvky komunikující pomocí technologie Wifi, je potřeba v prvotní části zařízení řádně nastavit. Nastavení spočívá v úspěšném přidání inteligentního zařízení do lokální sítě (LAN), ze které má být zařízení ovládáno. To znamená korektní připojení prvku k směrovači. Při prvním startu zařízení nebo po jeho resetování se začne chovat jako přístupový bod (access point). Na ten je potřeba se připojit například pomocí mobilního telefonu a spustit na něm mobilní aplikaci WeMo, která je dostupná, jak pro operační systém Android, tak iOS. Pomocí mobilní aplikace se předají zařízení přihlašovací údaje k směrovači, ke kterému se má připojit. Po úspěšném přidání chytrého zařízení do lokální sítě je možné jej ovládat a komunikovat s ním.

Pro nalezení prvků na síti je použit protokol SSDP (Simple Service Discovery Protocol), který je součástí sady protokolů UPnP (Universal Plug and Play). Protokol UPnP je blíže ukázán v sekci 3.1.2. Následná komunikace je založena na principu požadavek ⇄ odpověď (request ⇄ response). Zprávy pro ovládání zařízení jsou ve formátu SOAP (Simple Object Access Protocol), což je jazyk definovaný podle pravidel XML, který je blíže popsán v sekci 3.1.1.

3.1.1 SOAP (Simple Object Access Protocol)

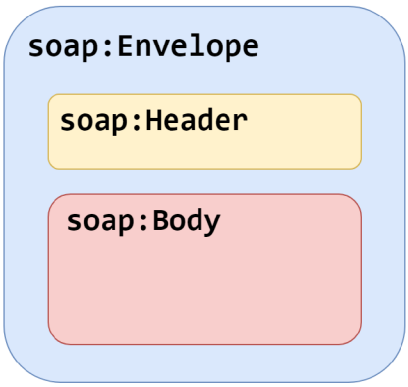
SAOP poskytuje jednoduchý a lehký mechanismus pro výměnu strukturovaných a typovaných informací mezi uzly v decentralizovaném a distribuovaném prostředí za pomocí XML. Pro přenos zpráv je obvykle využit HTTP aplikační protokol. Při přenosu SOAP zpráv pomocí HTTP protokolu, je přidána speciální hlavička `SOAPAction` do hlaviček HTTP. Hlavička může být použita pro definici účelu dané zprávy. Protokol se skládá ze tří částí [5]:

1. Konstrukce obálky SOAP definující z jakých částí se zpráva skládá, kdo by se měl kterou částí zabývat a také určuje které z částí jsou povinné a nepovinné. Struktura zprávy je znázorněna na obrázku 3.1. Zpráva SOAP je XML dokument, jehož kořenový element musí být obálka. Dále obálka může obsahovat hlavičku a tělo, které je povinnou částí tohoto prvku. Volitelná hlavička obálky může obsahovat informace o způsobu zpracování zprávy a komu je určena. Dále jsou také definovány elementy určené pro přenášení chybových či stavových informací. Pro tyto prvky platí, že se musí nacházet v elementu tělo a nesmí se zde vyskytnout vícekrát.

2. Pravidla kódování definující serializační mechanismus, které lze použít pro výměnu datových typů definovaných aplikací. Tím je možné přiřadit elementům datový typ a zajistit typovou kontrolu například, aby do elementu, v kterém je očekáváno celé číslo, se nepřiřadil například textový řetězec. Jednoduché datové typy SOAP přejímá z XML Schema[1]. Taktéž díky XML je možné vytvářet složitější datové typy pomocí jednoduchých základních datových typů.

3. RPC (remote procedure call) reprezentace definující konvenci, která může být použita k vzdálenému volání procedur a odpovědí. To umožňuje vykonat kód na jiném místě než je volající aplikace. Pro uskutečnění vzdáleného voláni je potřeba:

   - URI cílového objektu
   - jméno metody
   - volitelný podpis metody
   - parametry metody
   - volitelná hlavičková data



Obrázek 3.1: Formát zprávy SOAP

[1] https://www.w3.org/standards/xml/schema

Ukázka SOAP zprávy, vyňata z komunikace mezi mobilní aplikací respektive *BeeeOn Gateway* a inteligentní zásuvkou od Belkin WeMo, je dostupná v příkladu 3.1. Jejíž obsah říká zásuvce, aby se přepnula do stavu zapnuto. Na ukázce lze vidět, že kořenovým prvkem je obálka, která obsahuje tělo zprávy. Hlavička v tomto případě byla vynechána, což je dovoleno, protože hlavička je volitelným prvkem obálky.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:SetBinaryState xmlns:u="urn:Belkin:service:basicevent:1">
      <BinaryState>1</BinaryState>
    </u:SetBinaryState>
  </s:Body>
</s:Envelope>
```

Výpis 3.1: Ukázka SOAP zprávy

3.1.2 UPnP (Universal Plug and Play)

UPnP je sada protokolů definující architekturu pro *peer-to-peer* spojení mezi inteligentními přístroji, bezdrátovými zařízeními a počítači. Nejedná se o architekturu, která je rozšířením *Plug and Play*[2] technologie. UPnP je navrženo pro podporu *nulové konfigurace* a automatického vyhledání zařízení. To znamená, že zařízení se může připojit do sítě, dynamicky získat IP adresu a následně šířit informaci o tom, co za služby nabízí a zároveň získat přehled o již připojených zařízeních v síti a jejich vlastnostech. Zařízení také může síť automaticky opustit, aniž by za sebou zanechala nežádoucí stav. Příkladem může být situace, kdy v síti se nachází jeden kontrolní bod a jedno zařízení. Kontrolní bod o zařízení ví a počítá s ním. Aby zařízení při opuštění sítě nezanechalo nežádoucí stav, tak informuje všechny o tom, že se odpojuje. Nežádoucí stav tedy může znamenat, že kontrolní bod stále počítá s zařízením, ale to se již nevrátí.

Tato technologie využívá standardní protokoly jako IP, TCP, UDP, XML a HTTP. Toto je jedním z důvodů proč je prvním slovem jména univerzální. Jako další důvody je dobré zmínit, že není potřeba žádných ovladačů zařízení, není závislé na použitém médiu, implementace může být provedena pomocí jakéhokoliv programovacího jazyka pod jakýmkoliv operačním systémem s podporou IP komunikace.

Architektura UPnP rozlišuje dva typy zařízení, ovladatelná zařízení a kontrolní body. Ovladatelná zařazení jsou zde v roli serveru, což znamená, že odpovídají na požadavky od kontrolních bodů. Celá architektura se skládá z několika části [6]:

Adresování (Addressing) Základem pro vytváření UPnP architektury je IP adresování. Rozlišují se dva typy sítí spravovaná a nespravovaná. Spravovaná síť obsahuje DHCP server, který dynamicky přiděluje IP adresy připojovaným zařízením. Pokud není žádný DHCP server dostupný v síti, jedná se o nespravovanou síť, kde si zařízení IP adresu zvolí sami pomocí mechanismu Auto IP[3]. Ten definuje jak si zařízení inteligentně vybírá adresu IP ze skupiny rezervovaných adres.

[2] https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/introduction-to-plug-and-play
[3] https://tools.ietf.org/html/rfc3927

Vyhledávaní (Discovery) Po získání IP adresy zařízením je potřeba v případě ovladatelného zařízení svoji adresu propagovat (to advertise) a opačně u kontrolního bodu zařízení vyhledat (to search). V obou případech se přenášejí zprávy, které obsahují několik málo základních informací o daném zařízení, jako například jeho typ, identifikátor a ukazatel (URI) na podrobnější informace. Aby vyhledávací zprávy i propagační zprávy, byly doručeny všem zařízením podporujícím UPnP, je využita multicastová komunikace. Multicastová skupina UPnP naslouchá na IP adrese 239.255.255.250 a portu 1900. Formát zpráv určuje protokol SSDP (Simple Service Discovery Protocol), který pro přenos zpráv používá aplikační protokol HTTP nad transportním protokolem UDP.

Ukázka vyhledávací zprávy je zobrazena ve výpisu 3.2. Zpráva obsahuje hlavičky, které jsou za každé situace povinné. Důležitou hlavičkou je *ST*, pomocí které se definuje typ vyhledávaného zařízení. V této ukázce se vyhledávají všechna zařízení, která podporují SSDP protokol. Hlavička *MX* říká zařízení, jak dlouho bude čekat kontrolní bod na odpovědi. Po přijetí vyhledávací zprávy zařízením, zařízení čeká náhodnou dobu mezi 0 a hodnotou v hlavičce *MX*, aby vyvážilo zatížení kontrolního bodu při zpracování odpovědí. Každá odpověď 3.3 obsahuje hlavičku *Location* obsahující URL adresu, na které lze nalézt popis zařízení. Z URL adresy také dá zjistit IP adresa a port, na kterém zařízení naslouchá.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 5
ST: ssdp:all
```

Výpis 3.2: Vyhledávací zpráva

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=86400
EXT:
LOCATION: http://192.168.101.1/description.xml
SERVER: Unspecified, UP
ST: ssdp:all
```

Výpis 3.3: Odpověď na vyhledávací zprávu

Popis (Description) Jakmile kontrolní bod nalezne zařízení, tak o něm ještě zná velmi málo informací. Aby kontrolní bod získal více informací o zařízení, mezi které patří seznam vlastností zařízení, jak se zařízením interagovat, musí načíst popis zařízení z URL, které je obsaženo ve vyhledávací zprávě. UPnP popis zařízení je vyjádřen v XML a obsahuje jak informace přidělené výrobcem a seznam vestavěných zařazení a služeb, ze kterých se zařízení skládá, tak URL adresy pro ovládání, řízení událostí a prezentaci. Ukázkou takového popisu lze nalézt zde A.1, kde je popsána inteligentní zásuvka od Belkin WeMo. Pro každou službu existuje popis obsahující seznam příkazů, na které dokáže reagovat, a seznam parametrů pro každou akci. Popis jedné ze služeb na inteligentní zásuvce Belkin WeMo je dostupný v příloze A.2.

Ovládání (Control) Jakmile kontrolní bod již zná schopnosti ovladatelného zařízení, tak je schopen posílat požadavky na jeho služby. Pro zajištění ovládání zařízení, kontrolní bod posílá vhodné zprávy na URL služby, pomocí které se definuje akce, kterou má zařízení

provést. Kontrolní zprávy taktéž využívají XML s nadstavbou SOAP viz. 3.1.1. Podobně jako je tomu u volání funkcí, odpověď na kontrolní zprávu obsahuje informace o provedení akce.

Notifikace událostí (Event notification) Popis služby se mimo jiné skládá z proměnných, které modelují stav zařízení. Na základě změn těchto proměnných může zařízení vygenerovat událost, kterou rozešle kontrolním bodům. Zpráva o události obsahuje jméno proměnné a její novou hodnotu. Těchto dvojic může zpráva obsahovat více. První způsob rozesílání zpráv o události kontrolním bodům funguje na principu, kdy se kontrolní bod musí přihlásit k odběru těchto zprav u zařízení. V tomto případě se jedná o unicastovou komunikaci za použití spolehlivého transportního protokolu TCP. V druhém případě zařízení posílají zprávy o události všem zařízením připojeným do již zmíněné multicastové skupiny. Nevýhodou tohoto způsobu je využití UDP transportního protokolu, který nezaručuje doručení zprávy k příjemci. To může způsobit, že některé kontrolní body neobdrží zprávu o vzniklé události.

Prezentace (Presentation) Pokud má zařízení URL určenou k prezentaci, pak může kontrolní bod načíst stránku z této URL adresy, kterou lze zobrazit v prohlížeči a v závislosti na možnostech stránky umožnit uživateli ovládat zařízení a zobrazit jeho stav.

3.1.3 Belkin WeMo Switch

Prvním produktem od výrobce Belkin WeMo představuje chytrou elektrickou zásuvku. Funkcionalitu, kterou zásuvka nabízí je vzdálené ovládání spotřebičů připojených k ní. Zásuvku lze ovládat vzdáleně pomocí počítačové sítě, ale i pomocí hardwarového tlačítka určené k manuálnímu ovládaní. Fotografie inteligentní zásuvky lze vidět na obrázku 3.2.

Obrázek 3.2: Belkin WeMo inteligentní elektrická zásuvka

Relevantní akce se zásuvkou, které jsou potřeba pro základní řízení zařízení systémem *BeeeOn* jsou: získání MAC adresy zásuvky pro vytvoření identifikátoru zařízení, získání aktuálního stavu zásuvky a nastavení požadovaného stavu. Všechny zprávy jsou odesílány pomocí HTTP aplikačního protokolu metodou `POST` na URL `/upnp/control/basicevent1`.

Pro potřeby jednoznačné identifikace zařízení je nutné zjistit určitou vlastnost zařízení, která tuto podmínku pro odlišení zařízení bude splňovat aspoň v rámci jednoho manažera zařízení. V případě zásuvky se jedná o MAC adresu, která vyhovuje těmto podmínkám. Požadavek pro zjištění MAC adresy zásuvky je zobrazen ve výpisu 3.4. Z odpovědi zásuvky 3.5 lze vyčíst MAC adresu, která je obsažena v prvku `MacAddr`.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetMacAddr xmlns:u="urn:Belkin:service:basicevent:1" />
  </s:Body>
</s:Envelope>
```

Výpis 3.4: Zpráva k zjištění MAC adresy

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetMacAddrResponse xmlns:u="urn:Belkin:service:basicevent:1">
      <MacAddr>112233445566</MacAddr>
      <SerialNo>221618K1100220</SerialNo>
      <PluginUDN>uuid:Socket-1_0-221618K1100220</PluginUDN>
    </u:GetMacAddrResponse>
  </s:Body>
</s:Envelope>
```

Výpis 3.5: Odpověď k zjištění MAC adresy

Výpis 3.6 obsahuje zprávu odeslanou na zásuvku za účelem zjištění aktuálního stavu zásuvky. Odpověď 3.7 na požadavek obsahuje aktuální stav zásuvky v elementu `BinaryState`. Ukázková odpověď říká, že zásuvka je aktuálně ve stavu zapnuta.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetBinaryState xmlns:u="urn:Belkin:service:basicevent:1">
      <BinaryState />
    </u:GetBinaryState>
  </s:Body>
</s:Envelope>
```

Výpis 3.6: Zpráva k zjištění stavu

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetBinaryStateResponse xmlns:u="urn:Belkin:service:basicevent:1">
      <BinaryState>1</BinaryState>
    </u:GetBinaryStateResponse>
  </s:Body>
</s:Envelope>
```

Výpis 3.7: Odpověď k zjištění stavu

Zprávu pro nastavení stavu lze vidět v 3.8, která vyžaduje od zásuvky, aby změnila svůj stav na zapnuta. Odpověď zásuvky 3.9 nese informaci o úspěšnosti akce, která je obsažena v elementu `BinaryState`. Pokud je obsah elementu stav, na který se měla zásuvka přepnout, je akce úspěšná. Při chybě element nese textový řetězec `error`. Akce může být neúspěšná například v případě, že se zásuvka má přepnout do stavu, ve kterém již aktuálně je.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:SetBinaryState xmlns:u="urn:Belkin:service:basicevent:1">
      <BinaryState>1</BinaryState>
    </u:SetBinaryState>
  </s:Body>
</s:Envelope>
```

Výpis 3.8: Zpráva k nastavení stavu

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:SetBinaryStateResponse xmlns:u="urn:Belkin:service:basicevent:1">
      <BinaryState>error</BinaryState>
    </u:SetBinaryStateResponse>
  </s:Body>
</s:Envelope>
```

Výpis 3.9: Odpověď k nastavení stavu

3.1.4 Belkin WeMo Dimmer

Dalším produktem, pro který je vytvářena podpora do projektu *BeeeOn* je inteligentní vypínač osvětlení. Na rozdíl od předchozí inteligentní zásuvky obsahuje vypínač dva aktorové moduly. První modul má totožný s inteligentní zásuvkou. Modul umožňuje ovládat stav, čím se myslí, zda je vypínač ve stavu zapnut či vypnut. Druhým modulem lze ovládat jas osvětlení, které je vypínačem řízeno. Inteligentní vypínač lze taktéž ovládat buď vzdáleně skrz počítačovou síť nebo manuálně pomocí hardwarového tlačítka a dotykového panelu.

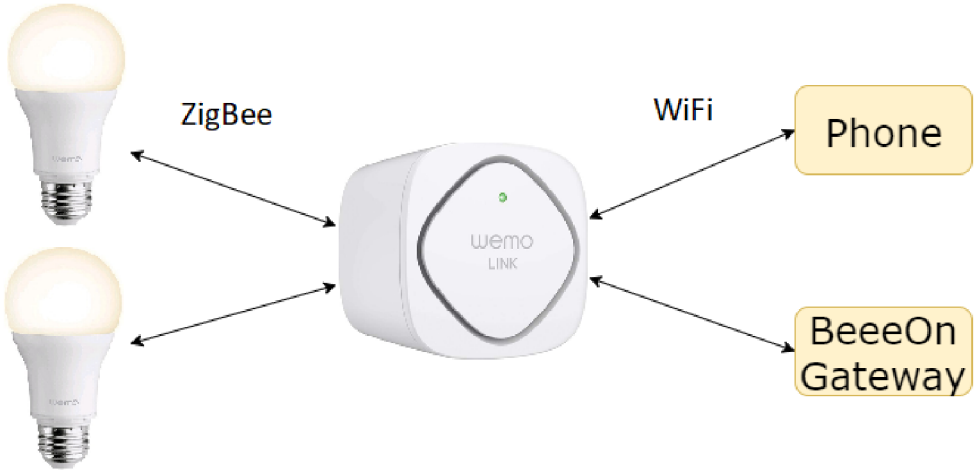
Obrázek 3.3: Belkin WeMo inteligentní vypínač osvětlení

Zprávy pro řízení vypínače osvětlení jsou téměř stejné jako u Belkin WeMo zásuvky 3.1.3. Z důvodu, že vypínač osvětlení obsahuje dva moduly, tak odpovědí na požadavek zjištění aktuálního stavu 3.7 obsahuje navíc element `brightness` na stejné úrovni jako prvek `BinaryState`. Element obsahuje aktuální nastavený jas vypínače osvětlení. Jas může být nastaven v rozmezí 0–100. V případě nastavování stavu vypínače osvětlení vypadá zpráva stejné jako u zásuvky 3.8. Zpráva se liší v případě, že má měnit jas. Místo elementu

`BinaryState` zpráva obsahuje prvek `brightness` obsahující úroveň nového jasu. V odpovědi 3.9 se v tomto případě nachází `Brightness` místo `BinaryState`.

3.1.5 Belkin WeMo Led Light

Součástí Belkin WeMo nabídky jsou také inteligentní žárovky. Chytrou žárovku je možné zapínat, vypínat a upravovat její jas. Komunikace s žárovkou je jistým způsobem rozdílná oproti předchozím zařízením od výrobce Belkin WeMo. Se samotnou žárovkou nelze komunikovat skrz počítačovou síť, ale pomocí technologie ZigBee. Aby bylo možné ovládat žárovku přes Internet, je k tomu potřeba prostředník, který se připojí do počítačové sítě. Odkud bude přijímat zprávy sloužící k ovládání žárovky a tyto zprávy přepošle konkretní žárovce pomocí ZigBee. Způsob komunikace je znázorněn na obrázku 3.4. Prostředník dokáže obsluhovat až 50 žárovek a je pojmenován Belkin WeMo Link.



Obrázek 3.4: Způsob komunikace s Belkin WeMo Led Light

Potřebné akce pro řízení inteligentních žárovek Belkin WeMo systémem *BeeeOn* jsou:

- zjištění MAC adresy Belkin WeMo Linku
- získání všech napárovaných žárovek na Belkin WeMo Linku
- získání aktuálního stavu žárovky
- nastavení stavu dané žárovky

Požadavek 3.4 a odpověď 3.5 pro získání MAC adresy zařízení jsou totožné jako v předchozích případech. Navíc z této odpovědi je potřeba uchovat obsah elementu `PluginUDN`, který je vyžadován ve zprávě pro získání všech napárovaných žárovek.

Všechny následující zprávy se odesílají na URL `/upnp/control/bridge1`. Pro získání všech napárovaných žárovek je potřeba odeslat zprávu 3.10 na Belkin WeMo Link. Odpověď odeslána Belkin WeMo Linkem na tento požadavek 3.11 obsahuje seznam napárovaných žárovek. O každé žárovce lze z odpovědi vyčíst její jméno, identifikátor a seznam vlastností. Vlastnosti žárovek jsou zde reprezentovány čísly. Vlastnost s číslem 10006 reprezentuje akotorvý modul, který umožňuje zásuvku zapnout či vypnout. Modul pro ovládání jasu je identifikován číslem 10008.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetEndDevices xmlns:u="urn:Belkin:service:bridge:1">
      <DevUDN>uuid:Bridge-1_0-221618K1100220</DevUDN>
      <ReqListType>PAIRED_LIST</ReqListType>
    </u:GetEndDevices>
  </s:Body>
</s:Envelope>
```

Výpis 3.10: Zpráva pro získání všech napárovaných žárovek

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetEndDevicesResponse xmlns:u="urn:Belkin:service:bridge:1">
      <DeviceLists>
        &gt;?xml version=&quot;1.0&quot; encoding=&quot;utf-8&quot;?&lt;
        &gt;DeviceLists&lt;
          &gt;DeviceList&lt;
            &gt;DeviceListType&lt;Paired&gt;/DeviceListType&lt;
            &gt;DeviceInfos&lt;
              &gt;DeviceInfo&lt;
                &gt;DeviceIndex&lt;0&gt;/DeviceIndex&lt;
                &gt;DeviceID&lt;1122334455667788&gt;/DeviceID&lt;
                &gt;FriendlyName&lt;Lightbulb&gt;/FriendlyName&lt;
                &gt;IconVersion&lt;1&gt;/IconVersion&lt;
                &gt;FirmwareVersion&lt;7E&gt;/FirmwareVersion&lt;
                &gt;CapabilityIDs&lt;
                  10006,10008,30008,30009,3000A
                &gt;/CapabilityIDs&lt;
                &gt;CurrentState&lt;,,,,&gt;/CurrentState&lt;
              &gt;/DeviceInfo&lt;
            &gt;/DeviceInfos&lt;
          &gt;/DeviceList&lt;
        &gt;/DeviceLists&lt;
      </DeviceLists>
    </u:GetEndDevicesResponse>
  </s:Body>
</s:Envelope>
```

Výpis 3.11: Odpověď pro získání všech napárovaných žárovek

Další akcí potřebnou k ovládání žárovek je získání aktuálního stavu konkretní žárovky. Pro provedení této akce slouží zpráva 3.12, v které se definuje identifikátor žárovky v elementu `DeviceIDs`. V odpovědi na tuto zprávu lze nalézt aktuální stav všech modulů. V ukázce 3.13 je žárovka ve stavu zapnuta a jas je na největší možné hodnotě 255. Tyto informace jsou obsaženy v prvku `CapabilityValue`.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
  <s:Body>
    <u:GetDeviceStatus xmlns:u="urn:Belkin:service:bridge:1">
      <DevUDN>uuid:Bridge-1_0-221618K1100220</DevUDN>
      <DeviceIDs>1122334455667788</DeviceIDs>
    </u:GetDeviceStatus>
  </s:Body>
</s:Envelope>
```

Výpis 3.12: Zpráva pro získání aktuálního stavu žárovky

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">'\
  <s:Body>
    <u:GetDeviceStatusResponse xmlns:u="urn:Belkin:service:bridge:1">
      <DeviceStatusList>
        &gt;?xml version=&quot;1.0&quot; encoding=&quot;utf-8&quot;?&lt;'\
        &gt;DeviceStatusList&lt;
          &gt;DeviceStatus&lt;
            &gt;IsGroupAction&lt;NO&gt;/IsGroupAction&lt;'\
            &gt;DeviceID available=&quot;YES&quot;&lt;1122334455667788&gt;/DeviceID&lt;
            &gt;CapabilityID&lt;10006,10008,30008,30009,3000A&gt;/CapabilityID&lt;
            &gt;CapabilityValue&lt;1,255:0,,,&gt;/CapabilityValue&lt;
          &gt;/DeviceStatus&lt;
        &gt;/DeviceStatusList&lt;
      </DeviceStatusList>
    </u:GetDeviceStatusResponse>
  </s:Body>
</s:Envelope>
```

Výpis 3.13: Odpověď pro získání aktuálního stavu žárovky

Poslední akcí, bez které se neobejde řízení inteligentních žárovek je nastavení konkretní žárovky do požadovaného stavu. Toto umožňuje zpráva v ukázce 3.14, která obsahuje identifikátor žárovky, identifikátor vlastnosti a novou hodnotu. Výsledkem příkladu zprávy by mělo být, že daná žárovka změní svůj jas na hodnotu 125. Belkin WeMo Link na základě úspěšnosti vygeneruje odpověď, která je zobrazena ve výpisu 3.15. Pokud element `ErrorDeviceIDs` obsahuje identifikátor žárovky, která měla měnit svůj stav, tak je změna stavu neúspěšná.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">'\
  <s:Body>
    <u:SetDeviceStatus xmlns:u="urn:Belkin:service:bridge:1">
      <DeviceStatusList>
        &gt;?xml version=\&quot;1.0\&quot; encoding=\&quot;UTF-8\&quot;?&lt;
        &gt;DeviceStatus&lt;
          &gt;IsGroupAction&lt;NO&gt;/IsGroupAction&lt;
          &gt;DeviceID available=\&quot;YES\&quot;&lt;1122334455667788&gt;/DeviceID&lt;
          &gt;CapabilityID&lt;10008&gt;/CapabilityID&lt;
          &gt;CapabilityValue&lt;125&gt;/CapabilityValue&lt;
        &gt;/DeviceStatus&lt;
```

```
      </DeviceStatusList>
    </u:SetDeviceStatus>
  </s:Body>
</s:Envelope>
```

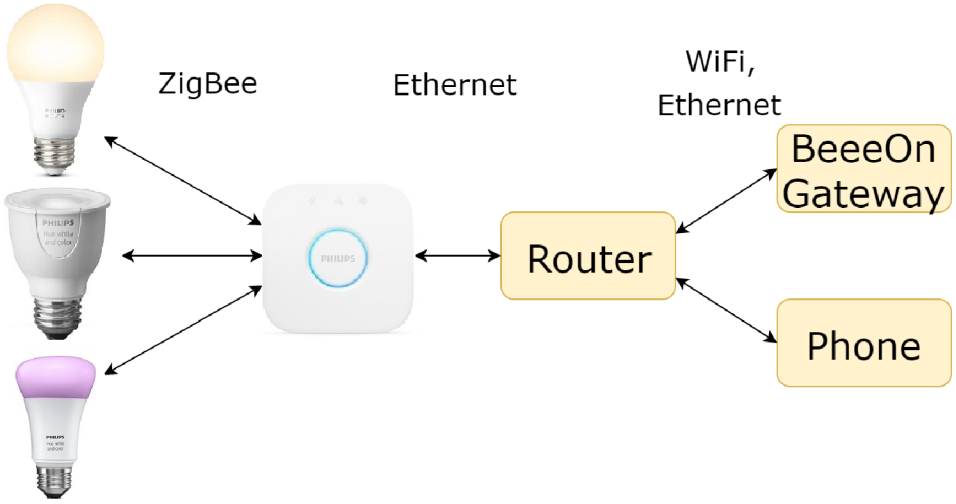
Výpis 3.14: Zpráva pro nastavení stavu žárovky

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:SetDeviceStatusResponse xmlns:u="urn:Belkin:service:bridge:1">
      <ErrorDeviceIDs></ErrorDeviceIDs>
    </u:SetDeviceStatusResponse>
   </s:Body>
</s:Envelope>
```

Výpis 3.15: Odpověď pro nastavení stavu žárovky

3.2 Philips Hue

Princip komunikace s inteligentními žárovkami je velmi podobný jako tomu bylo u Belkin WeMo chytrých žárovkách. Lampy a žárovky neobsahují hardware podporující komunikaci skrz počítačovou síť. S osvětlením si lze pouze vyměňovat zprávy pomocí technologie ZigBee. Z tohoto důvodu existuje Philips Hue Bridge, který vystupuje jako prostředník v komunikaci s žárovkami a lampami. Ten je připojen ke směrovači pomocí Ethernet kabelu, odkud přijímá zprávy určené k ovládání osvětlení, které případně přepošle dané žárovce skrz ZigBee. Graf znázorňující prvky podílející se na ovládání Philips Hue osvětlení a technologie použité pro komunikaci mezi zařízeními je zobrazen na obrázku 3.5.



Obrázek 3.5: Způsob komunikace s Philips Hue osvětlením

Způsob hledání Philips Hue Bridge na síti je totožný s vyhledáním Belkin WeMo zařízení, tudíž je použit protokol SSDP ze sady protokolů UPnP. Philips Hue zařízení na rozdíl od Belkin WeMo neposkytují UPnP popis jejich služeb. Pro komunikaci s Philips Hue

Bridge je použito RESTful aplikační rozhraní a přenášené zprávy jsou v JSON (JavaScript Object Notation) formátu.

3.2.1 Komunikace s Philips Hue Bridge

Za účelem řízení osvětlení jsou zde představeny příkazy a příslušné zprávy vyměňované mezi řídící stanicí (BeeeOn Gateway, mobilní aplikace) a prvkem Philips Hue Bridge.

Philips Hue na rozdíl od Belkin WeMo řeší určitým způsobem autorizaci zařízení přistupujícího k osvětlení. Zabezpečení funguje následujícím způsobem. Zařízení, které chce ovládat osvětlení, se musí při prvním přístupu k Philips Hue Bridge autorizovat. Proces autorizace začíná odesláním příkazu 3.16 zařízením na Philips Hue Bridge. Příjetím tohoto příkazu se Philips Hue Bridge přepne do stavu autorizace. Zpráva je odeslána na URL `/api` metodou `POST`. Element `devicetype` obsahuje řetězec ve formátu `jméno_aplikace#jméno_zařízení`. Ve stavu autorizace setrvá po dobu 30 sekund, kdy je vyžadováno zmačknutí hardwarového tlačítka na Philips Hue Bridge pro úspěšnou autorizaci. Po zmáčknutí tlačítka je potřeba opětovně poslat požadavek na autorizaci, kdy odpověď 3.17 bude obsahovat v prvku `username` vygenerovaný identifikátor. Tento unikátní identifikátor musí být použit v URL, která je obsažena v HTTP požadavcích odesílaných na něj [3].

```
{
  "devicetype":"BeeeOn#gateway"
}
```

Výpis 3.16: Požadavek na autorizaci

```
[
  {
    "success": {
      "username": "YTV2PIPXrtrnHFLafGQlcVyrcxSgNo8wv-NQPmVk"
    }
  }
]
```

Výpis 3.17: Úspěšná odpověď na autorizaci

Pro získání základních informací o konfiguraci Philips Hue Bridge je potřeba odeslat na URL `/api/<username>/config` HTTP požadavek metodou `GET`. Důležitou informací obsaženou v odpovědi 3.18 je MAC adresa Philips Hue Bridge, na základě které lze jednoznačně daný Philips Hue Bridge identifikovat.

```
{
  "name": "Philips hue",
  "datastoreversion": "63",
  "swversion": "1709131301",
  "apiversion": "1.21.0",
  "mac": "00:17:88:29:12:17",
  "bridgeid": "001788FFFE291217",
  "factorynew": false,
  "replacesbridgeid": null,
  "modelid": "BSB002",
  "starterkitid": ""
```

```
}
```

Výpis 3.18: Odpověď na konfiguraci

Za účelem získání všech žárovek obsluhovaných konkretním Philips Hue Bridgem, je potřeba na něj odeslat `GET` HTTP požadavek na URL `/api/<username>/lights`. V odpovědi na požadavek 3.19 se nachází seznam všech napárovaných žárovek. O každé žárovce se v odpovědi lze dozvědět její stav, typ, jméno, identifikátor a další. Taktéž důležitou informací o žárovce je pořadí v seznamu, protože se toto číslo využívá k identifikaci žárovky při získávání jejího aktuálního stavu a při nastavování nového stavu.

```
{
  "1": {
    "state": {
      "on": true,
      "bri": 51,
      "alert": "none",
      "reachable": true
    },
    "swupdate": {
      "state": "readytoinstall",
      "lastinstall": null
    },
    "type": "Dimmable light",
    "name": "Hue white lamp 1",
    "modelid": "LWB006",
    "manufacturername": "Philips",
    "uniqueid": "00:17:88:01:10:5c:34:5f-0b",
    "swversion": "5.38.1.15095"
  }
}
```

Výpis 3.19: Odpověď pro získání všech napárovaných žárovek

Pro zahájení hledání nových žárovek Philips Hue Bridgem slouží HTTP požadavek s metodou `POST` odeslaný na URL `/api/<username>/lights`. Úspěšnou odpověď na tento požadavek lze vidět ve výpisu 3.20. Hledání trvá 40 sekund, takže po uplynutí této doby se lze dotázat na seznam napárovaných žárovek, kde již budou v odpovědi obsaženy nově nalezené žárovky.

```
[
  {
    "success": {
      "/lights": "Searching for new devices"
    }
  }
]
```

Výpis 3.20: Odpověď pro zahájení hledání nových žárovek

Další příkaz důležitý pro řízení žárovek je zjištění aktuálního stavu konkretní žárovky. K tomu slouží HTTP požadavek `GET` odeslaný na URL `/api/<username>/lights/<id>`. `id` obsažené v URL je pořadové číslo žárovky, které lze zjistit z odpovědi na příkaz zjištění

seznamu napárovaných žárovek 3.19. Informace obsažené v odpovědi o žárovce jsou totožné jako v odpovědi na příkaz zjištění seznamu napárovaných žárovek.

K ovládání chování žárovky slouží příkaz na změnu stavu. Tato akce je provedena pomocí HTTP požadavku `PUT`, který je odeslán na URL `/api/<username>/lights/<id>/state`. Zpráva obsažena v požadavku obsahuje jména modulů a hodnoty, na které se mají dané moduly nastavit. Ukázka zprávy 3.21 požaduje od žárovky, aby změnila svůj jas na hodnotu 155. Na základě úspěšnosti akce Philips Hue Bridge vygeneruje odpověď. Příklad úspěšné odpovědi na změnu jasu žárovky je zobrazen ve výpisu 3.22 .

```
{
  "bri":155
}
```

Výpis 3.21: Zpráva pro nastavení jasu žárovky

```
[
  {
    "success": {
      "/lights/1/state/bri": 155
    }
  }
]
```

Výpis 3.22: Odpověď pro nastavení jasu žárovky

3.2.2 Philips Hue White

Jedním z nabízených produktů od Philips Hue je právě žárovka Philips Hue White. Jedná se o nejzákladnější žárovku, která i přes svoji jednoduchost nabízí možnost vzdáleného ovládání stavu (zapnuta/vypnuta) a jasu.

3.3 BeeWi

Posledním výrobcem inteligentních zařízení, která jsou analyzována a pro která je vytvářena podpora v rámci této práce je francouzská firma BeeWi. BeeWi pro komunikaci s inteligentními zařízeními zvolilo technologii Bluetooth Low Energy.

3.3.1 Blutooth Low Energy

Bluetooth Low Energy, známé také jako Bluetooth Smart, je technologie nabízející bezdrátovou komunikaci mezi zařízeními. Velikou výhodou této technologie je velmi nízká spotřeba energie při vytváření spojení, udržování spojení a výměně dat mezi zařízeními.

GAP (Generic Access Profile) Vrstva GAP z Bluetooth Low Energy zásobníku definuje a kontroluje způsob spojení dvou zařízení a propagaci informací o zařízení. GAP umožňuje, aby zařízení bylo viditelné okolnímu světu a také definuje způsob interakce mezi dvěma zařízeními. V rámci GAP se zařízení dělí na dva typy:

- **Centrální zařízení** – typicky se jedná o mobilní zařízení nebo počítač, který ovládá periferní zařízení. Ovládáním může být sběr naměřených dat nebo přepnutí zařízení do jiného stavu.

- **Periferní zařízení** – malá, zdrojově omezená zařízení, která nabízí určité služby. Může se jednat o teplotní senzor, inteligentní žárovku a tak podobně.

Propagaci informací o přítomnosti zařízení na síti lze provést dvěma způsoby. Při prvním způsobu periferní zařízení propaguje samo sebe v předem určených časových intervalech. Delší časový interval šetří energii, ale za to je menší šance, že tuto zprávu zachytí vyhledávající centrální zařízení. Druhým způsobem je, že centrální zařízení si vyžádá propagující zprávu o každém zařízení na síti. Když periferní zařízení příjme skenovací zprávu, tak odešle stejná data jako v prvním případě, ale již pouze konkretnímu centrálnímu zařízení. Následně po nalezení periferního zařízení centrálním zařízením, s ním může vytvořit spojení a začít vyměňovat zprávy pomocí GATT.

Pro situace, kdy je vyžadováno předávání aplikačních dat více zařízením najednou, je možné do propagačních zpráv periferního zařízení tyto data vložit. To není možné provést předchozím způsobem, protože tehdy dochází k výměně dat až při vytvoření konkretního spojení mezi dvěma zařízeními. Jakmile se vytvoří spojení mezi periferním zařízením a centrálním zařízením, propagační proces se obecně zastaví [8].

GATT (Generic Attribute Profile) GATT definuje způsob jakým si dvě Bluetooth LE zařízení přenášejí data tam a zpět za pomocí konceptu nazývaným *Služby a Vlastnosti*. Každé periferní zařízení se skládá z profilu, který obsahuje určitou množinu služeb.

Služba má za úkol rozdělit konkretní vlastnosti do logických entit. Každá služba může obsahovat více vlastností a právě jeden identifikátor nazývaný UUID, na základě kterého se odlišuje od jiných služeb. Na nejnižší úrovni je vlastnost zapouzdřující samotná data. Součástí vlastnosti je popis, datový typ, hodnota, přístupová práva, UUID a identifikátor nazývaný *handle*, pomocí kterého lze adresovat danou vlastnost.

GATT je postaven na datovém protokolu Attribute Protocol (ATT), který slouží k ukládání služeb, vlastností a souvisejících dat do jednoduché vyhledávací tabulky pomocí 16 bitového identifikátoru (handle). Také definuje způsob výměny zpráv mezi dvěma zařízeními a jejich formát.

Zpráva se skládá z 8 bitového identifikátoru příkazu, 16 bitového identifikátoru vlastnosti a případně obsahu zprávy. Komunikace mezi zařízeními je založena na principu klient/server. V roli GATT serveru účinkuje periferní zařízení obsahující ATT vyhledávací tabulku se službami a vlastnostmi, které přijímá a vyřizuje požadavky GATT klienta. Klientem v komunikaci může být mobilní telefon, tablet nebo právě *BeeeOn Gateway*.

Existují zde tři typy komunikace. První dvě možnosti iniciuje klient. Jedná se o požadavek na objevení služeb a vlastností periferního zařízení, a o požadavek pro manipulaci s daty na periferním zařízení. Třetí možností je odeslání zprávy periferním zařízení na základě výskytu určité události za účelem notifikovat centrální zařízení [2].

3.3.2 BeeWi Smart temperature & humidity sensor

Uvedený senzor slouží k měření teploty a vlhkosti vzduchu. Senzor obsahuje tři senzorové moduly pro měření teploty, vlhkosti a kapacity baterie, kterou je napájen. Fotografie senzoru je zobrazena na obrázku 3.6.

Obrázek 3.6: BeeWi teplotní a vlhkostní senzor

Potřebné požadavky na senzor pro řízení jsou: získání čísla modelu a získání aktuálních hodnot senzorových modulů. Po vyhledání Bluetooth LE zařízení, známe pouze jejich MAC adresu, která se případně použije pro vytvoření identifikátoru zařízení v systému *BeeeOn*. Aby bylo možné zjistit, že konkrétní zařízení je právě tento senzor, je potřeba jej požádat o číslo modelu. Tyto data jsou uložena na zařízení ve vlastnosti s UUID `00002a24-0000-1000-8000-00805f9b34fb`. Pro získání čísla modelu je nutné odeslat čtecí požadavek na zmíněnou vlastnost. V úspěšné odpovědi lze nalézt požadované modelové číslo. Očekávané číslo modelu pro senzor je `BeeWi BBW200`, které je v odpovědi kódováno v *Unicode*.

Aktuální naměřené hodnoty jsou obsaženy ve vlastnosti identifikovatelnou pomocí UUID `a8b3fb43-4834-4051-89d0-3de95cddd318`. Získat se dají stejným způsobem jako číslo modelu. Odpověď na požadavek je deseti bytový blok, který obsahuje všechny tři hodnoty senzorových modulů. Na druhém a třetím bytu se nachází teplota. Kladné a záporné hodnoty jsou zakódovány rozdílně. Pokud se druhý byte rovná 255, tak se jedná o zápornou hodnotu. Pro kladné hodnoty lze teplotu vypočítat tímto způsobem:

$$\frac{\text{byte}[1] + (\text{byte}[2] \ll 8)}{10}$$

V opačném případě se záporná teplota získá za pomocí tohoto výpočtu:

$$\frac{\text{byte}[1] - \text{byte}[2]}{10}$$

Vlhkost se nachází na pátém bytu a úroveň baterie na desátem. Oba moduly jsou vyjádřeny v procentech.

3.3.3 BeeWi Smart LED Light Bulb

Mezi akční prvky inteligentní domácnosti nabízenými BeeWi patří chytrá žárovka. Žárovka disponuje aktorovými moduly pro ovládání stavu (zapnuta/vypnuta), jasu, barvy a teploty bílé. Oproti předchozím inteligentním žárovkám se s žárovkou BeeWi komunikuje přímo, bez žádného prostředníka.

Obrázek 3.7: BeeWi LED žárovka

Nezbytnými akcemi pro řízení žárovky jsou: získání čísla modelu, získání aktuálního stavu aktorů a nastavení nové hodnoty všech aktorů. Důvod potřeby čísla modelu je stejný jako u BeeWi senzoru. Slouží k zjištění, zda konkretní nalezené zařízení je BeeWi inteligentní žárovka. Číslo modelu charakterizující žárovky, pro které je vytvářena podpora je `BeeWi BLR07`.

Pro získání aktuálního nastavení aktorů je nutné odeslat čtecí požadavek na vlastnost s UUID `a8b3fff2-4834-4051-89d0-3de95cddd318`. Odpověď se skládá z pěti bytů obsahující aktuální hodnoty všech aktorových modulů. První byte obsahuje informaci o stavu žárovky, to znamená, jestli je vypnutá respektive zapnutá. V druhém bytu se nachází aktuálně nastavená hodnota jasu a teploty bílé. První čtyři bity druhého bytu nesou hodnotu jasu a následující čtyři bity představují hodnotu teploty bílé. Poslední tři byty obsahují hodnoty postupně jednotlivých RGB složek.

Všechny příkazy na změnu hodnoty aktorů jsou posílány pomocí příkazu zápisu na vlastnost s UUID `a8b3fff1-4834-4051-89d0-3de95cddd318`. Obsah zprávy je rozdělen na 4 části. Tyto části a formát zprávy je zobrazen na obrázku 3.8.

prefix (0x55)	kód příkazu	parametry	suffix (0x0D0A)

Obrázek 3.8: Zpráva pro ovládání žárovky

Tabulka 3.1 obsahuje všechny příkazy pro nastavování aktorů. Ke každému příkazu je doplněn jeho kód a možné parametry.

Příkaz	Kód	Parametry
Nastavení stavu (zapnuta/vypnuta)	0x10	1 byte s hodnotou v intervalu 0-1.
Nastavení teploty bíle	0x11	1 byte s hodnotou v intervalu 2-11.
Nastavení jasu	0x12	1 byte s hodnotou v intervalu 2-11.
Nastavení barvy	0x13	3 byty, kde postupně každý byte představuje jednu složku RGB. Hodnota každé složky může být v intervalu 0-255.

Tabulka 3.1: Seznam příkazů pro ovládání BeeWi inteligentní žárovky

Kapitola 4

Návrh integrace vybraných technologií do systému BeeeOn

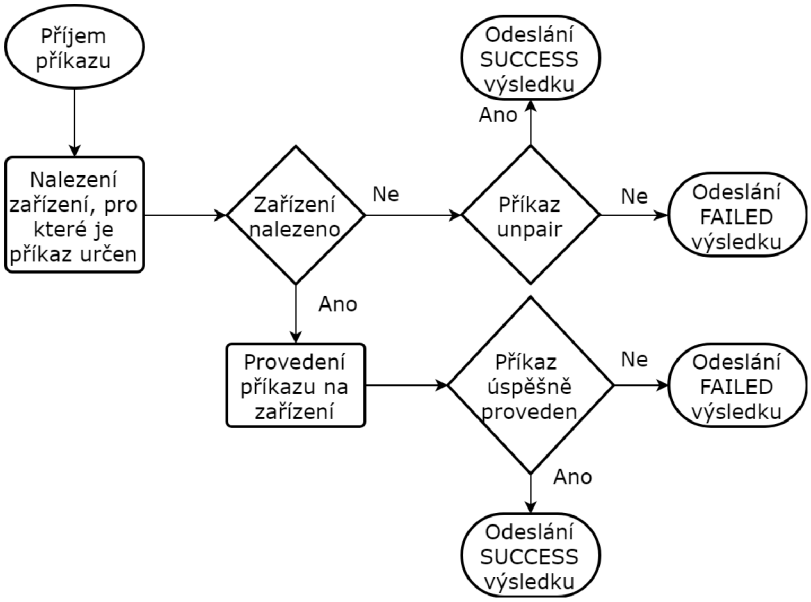
Zaměřením této kapitoly je na návrh nových modulů. V prvé řadě je popsán manažer zařízení, který slouží k zajištění řízení inteligentních prvků. Součástí popisu manažeru zařízení jsou vyjmenovány jeho funkce a způsob jakým je realizuje. Následně je popsán návrh nových manažerů zařízení zajišťujících podporu zmíněných zařízení popsaných v kapitole 3. Při jejich návrhu byl kladen velký důraz na jednoduchost přidávání nově podporovaných zařízení.

4.1 Manažer zařízení

Hlavními úkoly manažera zařízení jsou:

- přijímání příkazu od uživatele k ovládání akčních prvků
- komunikace s koncovými zařízeními
- sběr aktuálních hodnot na zařízení

Modul manažera zařízení je spuštěn v samostatném vlákně. V něm typicky manažer zařízení provádí sběr dat ze zařízení. Příkazy směřující na manažera zařízení jsou realizovány v jiných oddělených vláknech. Toho lze využít především při komunikaci s fyzickými zařízeními, která můžou být náchylná na nepředvídatelná zpoždění. To umožňuje manažerovi zařízení zpracovat několik příkazu najednou. Algoritmus zpracování příkazů, pracující se zařízením v manažerovi zařízení, je znázorněn ve vývojovém diagramu na obrázku 4.1.



Obrázek 4.1: Vývojový digram popisující zpracování příkazu manažerem zařízení

V počáteční fázi se manažer zařízení nestará o žádné zařízení. Při startu aplikace manažer zařízení zjišťuje, jestli v předchozím běhu již neměl napárovaná nějaká zařízení. To provede odesláním příkazu *device-list* na server, který mu vrátí identifikátory již napárovaných zařízení. Pokud je seznam napárovaných zařízení po této akci neprázdný, manažer zařízení se pokusí tyto zařízení vyhledat. Součástí tohoto procesu může být také zjištění poslední naměřené hodnoty zařízením a jeho navrácení do tohoto stavu.

Hledání nových zařízení manažerem zařízení se spouští po přijetí příkazu *listen*. O každém nalezeném zařízení musí manažer zařízení odeslat zprávu *new-device* obsahující jeho identifikátor, jméno, výrobce a moduly, kterými disponuje. Díky této zprávě lze informovat uživatele o nalezeném zařízení, které je pak schopen přidat do své chytré domácnosti. O každém objeveném zařízení si manažér zařízení vytvořit záznam, na základě kterého bude schopen zařízení spravovat.

Aby bylo možné zařízení ovládat a sbírat aktuální hodnoty jeho modulů, je nutné dané zařízení napárovat. Proces párování se provede po přijmutí příkazu *device-accept* manažerem zařízení. Na základě identifikátoru zařízení obsaženém v příkazu, manažer zařízení zjistí, zda již takové zařízení bylo objeveno. Pokud o přidávaném zařízení existuje záznam, tak si manažer zařízení dané zařízení registruje jako napárované. V této chvíli může manažer zařízení dané zařízení ovládat a zjišťovat aktuální hodnoty jeho modulů.

Když se uživatel rozhodne, že už nemá zájem o ovládání a sběr dat ze zařízení, provede odpárování. V takovém případě manažer zařízení příjme příkaz *unpair*, na základě kterého odregistruje zařízení a přestane s ním komunikovat.

4.1.1 Třída DongleDeviceManager

DongleDeviceManager je abstraktní třída, která kromě výše zmíněných funkcí také řeší akce spojené s připojitelným zařízením. Připojitelným zařízením může být například USB dongle. Pomocí připojení USB donglu k bráně je možné přidat podporu dalších komunikačních technologií. Akce a situace, které je nutné řešit kvůli donglu jsou: dongle chybí, dongle připojen a dongle odpojen.

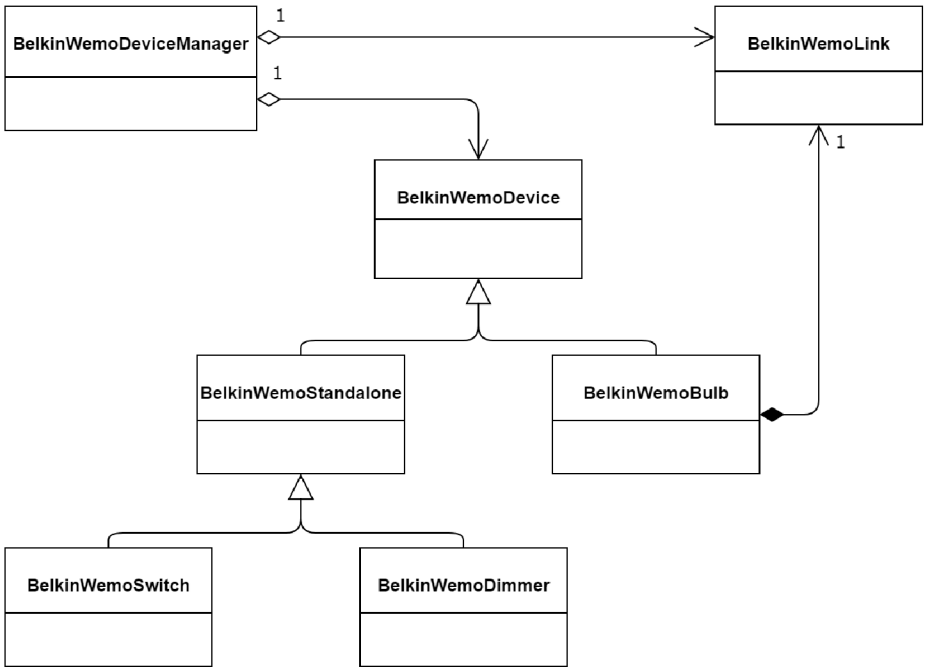
4.2 Návrh podpory Belkin WeMo

Objektově orientovaný návrh pro podporu Belkin WeMo zařízení lze vidět na obrázku 4.2. Obrázek obsahuje třídy, které zajistí řízení zmíněných Belkin WeMo inteligentních zařízení.

Logiku manažera zařízení realizuje třída *BelkinWemoDeviceManager*, která dědí abstraktní třídu *DeviceManager*. Pro každé fyzické zařízení existuje třída, která má za úkol umožnit jeho ovládaní. Každá třída představující fyzické zařízení dědí třídu *BelkinWemoDevice*, aby manažer zařízení mohl se všemi typy zařízení pracovat jednotně. Tato třída nabízí jednotné aplikační rozhraní k práci se zařízením. Toto je umožněno díky polymorfismu.

Pro ovládání inteligentních žárovek jsou určeny třídy *BelkinWemoLink* a *BelkinWemoBulb*. *BelkinWemoLink* slouží ke komunikaci s fyzickým Belkin WeMo Linkem. Každá instance žárovky musí mít odkaz na příslušný Belkin WeMo Link, ke kterému je fyzicky připojena. Manažer zařízení pro získání a změnu stavu žárovky kontaktuje její příslušnou instanci, a ta požadovaný příkaz předá instanci Belkin WeMo Linku, který operaci provede. Manažer zařízení navíc spravuje všechny instance třídy *BelkinWemoLink*, pomocí kterých se provádí hledání nových žárovek.

Inteligentní zásuvka a vypínač osvětlení spolu sdílí většinu komunikačních zpráv. Proto se zde nachází abstraktní třída *BelkinWemoStandalone*, která obsahuje metody pro ovládání těchto zařízení. Komunikace se liší ve zpracování některých odpovědí. Proto jsou zde pro obě zařízení vlastní třídy, kde je implementováno zpracováni rozdílných odpovědí.



Obrázek 4.2: Objektový návrh pro podporu Belkin WeMo

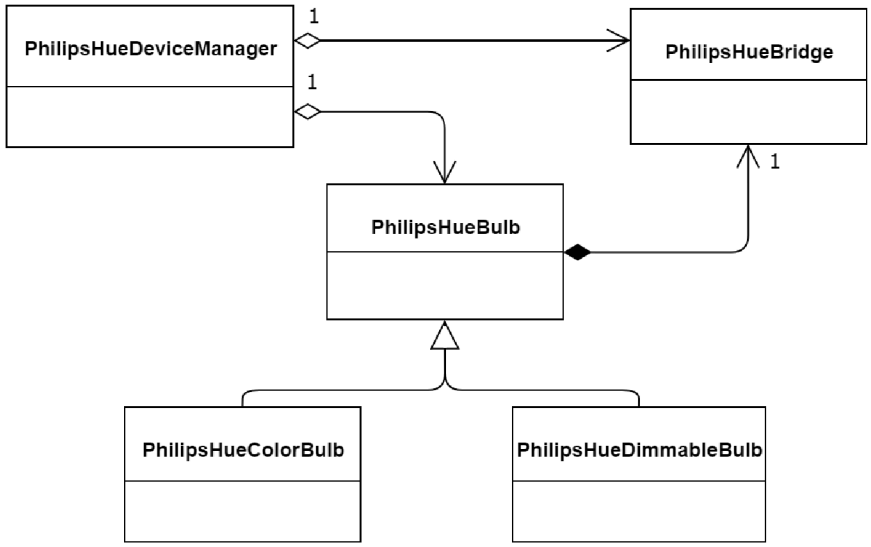
4.3 Návrh podpory Philips Hue

Návrh podpory Philips Hue je zobrazen na obrázku 4.3, který zobrazuje objektově orientovaný návrh a následně vysvětlen dále v textu.

PhilipsHue používá obdobnou architekturu jako Belkin Wemo. Třída *PhilipsHueDeviceManager* implementuje chování manažera zařízení. Pro zachování jednotného přístupu

manažeru zařízení k žárovkám existuje abstraktní třída *PhilisHueBulb*, kterou každá varianta žárovky dědí. Komunikace s žárovkami není přímá, ale vyžaduje prostředníka – Philips Hue Bridge. V návrhu se proto nachází třída *PhilipsHueBridge* zapouzdřující komunikaci s její fyzickou podobou. Každá instance žárovky obsahuje právě jeden odkaz na *PhilipsHueBridge*, který se využívá při volání metody žárovky pro zjištění jejího stavu nebo pro změnu jejího stavu. Kromě instancí žárovek se manažer zařízení také stará o instance třídy *PhilipsHueBridge*, protože pomocí nich objevuje nové žárovky.

Odlišné varianty žárovek disponují různými aktorovými moduly, což se projevuje ve zprávách pro nastavení stavu a pro získání aktuálního stavu. Z tohoto důvodu existuje pro každý typ žárovky vlastní třída, která implementuje vytváření a zpracování zpráv, speciální pro každý typ žárovky.



Obrázek 4.3: Objektový návrh pro podporu Philips Hue

4.4 Návrh podpory BeeWi

Způsob rozšíření aplikace *BeeeOn Gateway* o řízení BeeWi zařízení, je znázorněn na obrázku 4.4, který obsahuje objektově orientovaný návrh. Součástí návrhu jsou třídy a vazby mezi nimi, které zajišťují podporu inteligentních zařízení BeeWi.

Manažer zařízení je v tomto případě realizován třídou *BeeWiDeviceManager*. Oproti předchozím případům dědí tato třída *DongleDeviceManager*, protože komunikace se zařízeními je prováděna technologií Bluetooth. Bluetooth řadič je připojen k bráně pomocí USB donglu, protože hardware brány nedisponuje Bluetooth modulem. Aby mohl manažer zařízení zacházet se všemi BeeWi zařízení uniformně, existuje v návrhu třída *BeeWiDevice* definující rozhraní, které každá třída reprezentující zařízení implementuje. Pro potřeby hledání nových zařízení má třída manažera zařízení asociační vazbu s třídou *DBusHciInterface*, která zajišťuje komunikaci s Bluetooth řadičem.

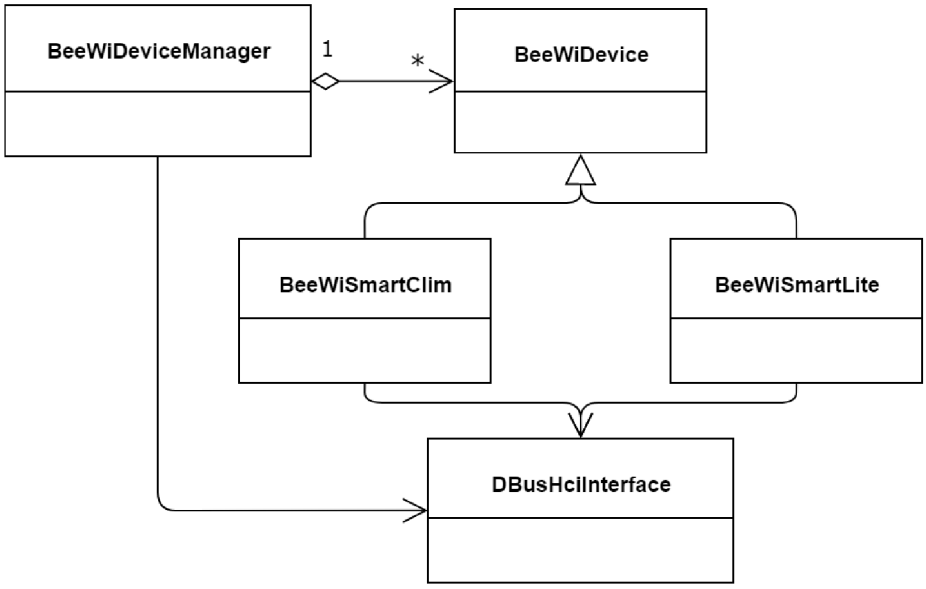
Bluetooth komunikace je v architektuře *BeeeOn* oddělena třídou *HciInterface* poskytující metody:

- pro hledání nových Bluetooth Classic zařízení
- pro detekci Bluetooth Classic zařízení

- pro hledání nových Bluetooth Low Energy zařízení
- pro získání statistik Bluetooth rozhraní

Díky této abstraktní třídě je možné modulům, které využívají Bluetooth, vyměnit implementaci Bluetooth bez větších změn. Pro řízení BeeWi zařízení je potřeba kromě hledání nových zařízení také odesílání čtecích a zapisovacích požadavků na ně. Proto je vytvořena třída *DBusHciInterface* implementující komunikaci s Bluetooth řadičem pomocí sběrnice DBus. Tato třída implementuje abstraktní třídu *HciInterface* a také poskytuje metody pro odesílání požadavků. Z tohoto důvodu má třída manažera zařízení a jednotlivá zařízení, vazbu s třídou *DBusHciInterface*. Důležitým požadavkem na třídu *DBusHciInterface* je možnost používat Bluetooth komunikaci ve více modulech zároveň.

Pro každý typ zařízení je vytvořena samostatná třída, která dědí a implementuje abstraktní třídu *BeeWiDevice*. Nezbytnými metodami u třídy zařízení *BeeWiDevice*, jsou získání aktuálního stavu a nastavení aktorového modulu na danou hodnotu. Třída *BeeWiSmartClim* reprezentuje senzor pro měření teploty a vlhkosti. Poslední nezmíněnou třídou je *BeeWiSmartLite*, která představuje inteligentní žárovku BeeWi. Úkolem těchto tříd je komunikace s jejich fyzickou podobou, proto mají vazbu s třídou *DBusHciInterface*.



Obrázek 4.4: Objektový návrh pro podporu BeeWi

Kapitola 5

Implementace podpory chytrých zařízení

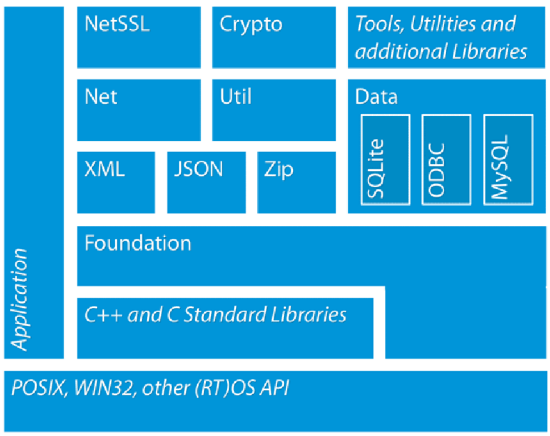
Tato kapitola se věnuje implementaci podpory a rozdílům od návrhu podpory zmíněných inteligentních prvků domácnosti. Součástí kapitoly je část zmiňující technologie, které byly využity pro zjednodušení implementace. Závěr této kapitoly se věnuje způsobu překladu nově vytvořených modulů a možnosti jejich rozšíření.

5.1 Použité technologie

Implementace aplikace *BeeeOn Gateway* je napsána v jazyce C++. Jedná se o jazyk kompilovaný a disponuje objektově orientovaným paradigmatem, které je primárně využito v projektu *BeeeOn*. Kromě standardních knihoven jazyka C++ jsou použity C++ a C knihovny, které jsou podrobněji popsány níže.

5.1.1 Knihovna Poco

Knihovna Poco je kolekce open-source knihoven, zaměřující se na zjednodušení a urychlení vývoje aplikací, které využívají komunikaci skrz počítačovou síť. Knihovnu lze použít, jak pro vývoj počítačové aplikace, tak pro mobilní či vestavěné aplikace. Architektura knihovny Poco je rozdělená do několika modulů, které je možné vidět na obrázku 5.1 [4].



Obrázek 5.1: Přehled modulů C++ knihovny Poco

Kromě základního modulu Foundation, který implementuje jádro knihovny Poco, byly použity moduly XML, JSON a Net. Z modulu Foundation byly použity třídy pro vytváření a správu vláken, pro zpracování čísel reprezentovaných v textovém řetězci, pro uchování a vytváření časových značek, pro vytváření výjimek a další. Modul XML je využit pro vytváření a zpracování zpráv vyměňovaných mezi bránou a Belkin WeMo zařízeními. Pro stejné účely byl použit modul JSON, za pomocí kterého se konstruují a analyzují zprávy odesílané a přijímané od Philips Hue Bridge. Koncová zařízení od Belkin WeMo a Philips Hue používají pro zajištění komunikace počítačovou síť. Z tohoto důvodu je využit modul Net, který zajistí spojení a výměnu dat mezi bránou a koncovým zařízením.

5.1.2 Knihovna GLib

GLib je balík nízkoúrovňových systémových knihoven napsaných v jazyce C. Knihovna je převážně vyvíjená vývojáři GNOME. Tato knihovna poskytuje pokročilé datové struktury, správu vláken, aplikační rozhraní k virtuálním souborovým systémům, vytváření callbacků, časovačů a další [7].

V nově vytvořených modulech se z této knihovny využívá DBus aplikační rozhraní, pomocí kterého se komunikuje se systémovou službou zajišťující komunikaci pomocí technologie Bluetooth. Další použitou funkcionalitou je hlavní smyčka, která umožňuje zpracování callbacků připojených k časovačům nebo signálům.

5.2 Popis implementace jednotlivých modulů

Stejně jako pro již existující moduly, tak pro nově vytvořené moduly v aplikaci *BeeeOn Gateway* musí existovat část v konfiguračním souboru, která specifikuje jejich nastavení. Současně také existuje konfigurační soubor určující, kterým třídám je umožněno vytvářet logovací výpisy a definovat úroveň logovacích zpráv, které se od dané třídy mají zaznamenávat. Díky těmto konfiguračním souborům je možné změnit nastavení aplikace bez nutnosti překladu programu. V následujících podsekcích je popsána implementace jednotlivých modulů založených na jejich návrhu.

Pro každý modul popsaný níže platí, že jádrem je manažer zařízení. Pro potřeby uchování objektů reprezentujících chytrá zařízení v manažerovi zařízení je využit kontejner ze standardní knihovny C++ *std::map*. Prvkem tohoto kontejneru je dvojice klíč a hodnota. Klíčem je identifikační číslo zařízení (*DeviceID*) a hodnotu představuje příslušná instance objektu reprezentující dané zařízení. Zařízení obsažená v manažerovi zařízení můžou být buď napárovaná nebo nenapárovaná. Pro rozlišení stavu zařízení si manažer zařízení ukládá identifikátory napárovaných zařízení v kontejneru *std::set*. Díky tohoto kontejneru může jednoduše zařízení napárovat, odpárovat a zjistit, o kterých zařízeních se mají sbírat data.

V každém modulu také existuje abstraktní třída, která definuje atributy a metody, jež musí každá třída reprezentující fyzické zařízení obsahovat. Smyslem této třídy je jednotná práce se zařízeními a možnost uchování všech typů zařízení v jednom kontejneru. Metody, které musí každá podřazená třída implementovat jsou:

- získání aktuálních hodnot všech modulů
- vykonání požadavku na změnu hodnoty aktorů
- vrácení jména zařízení
- vrácení všech modulů, kterými zařízení disponuje

5.2.1 Modul pro řízení Belkin WeMo zařízení

Účelem modulu je zajistit řízení inteligentních zařízení od firmy Belkin WeMo. Skutečná implementace modulu vychází z návrhu dostupného v sekci 4.2 beze změn. Jádrem modulu je manažer zařízení, který je implementován ve třídě *BelkinWemoDeviceManager*.

Kromě inteligentních zařízení si třída *BelkinWemoDeviceManager* uchovává také objekty reprezentující Belkin WeMo Link. Pro uložení těchto objektů je použit stejný kontejner jako v případě uchování instancí zařízení, s tím rozdílem, že pro identifikaci prvku kontejneru je použit jiný vyhledávací klíč. Protože Belkin WeMo Link není inteligentní zařízení samo o sobě, nevytváří se mu jeho identifikační číslo. Z tohoto důvodu je každý Belkin WeMo Link identifikován pomocí jeho MAC adresy.

Vyhledávání nových zařízení

Aby bylo možné řídit nějaká zařízení, je v prvé řadě nutné je nalézt na síti. K tomu slouží třída *UPnP*, která implementuje vyhledávání zařízení za pomocí SSDP protokolu. Pro vytváření a odeslání UDP datagramů je použitá třída *Poco::Net::DatagramSocket* z Poco knihovny. Výsledkem vyhledávání je seznam IP adres a portů, na kterých nalezená zařízení naslouchají.

Vyhledávaní probíhá v samostatném vlákně, protože objevování nových zařízení může trvat až několik minut. Následně se pro každé nalezené zařízení vytvoří instance příslušné třídy a zjišťuje se, jestli už dané zařízení manažer zařízení zná. Pokud již manažer zařízení obsahuje záznam o zařízení, zkontroluje jestli se mu nezměnila IP adresa a případně ji aktualizuje. Pro objevení nových chytrých žárovek je nutné provést ještě nad každým nalezeným Belkin WeMo Linkem akci vrácení všech napárovaných žárovek. O každém nově nalezeném zařízení, kromě Belkin WeMo Linku, se vytvoří instance *NewDeviceCommand* třídy, která je odeslaná na server pro jeho informování o novém zařízení.

Práce s zařízeními

Stejně jako v návrhu, je pro každé zařízení implementovaná třída, která má za úkol komunikovat s její fyzickou podobou. Tyto třídy implementují abstraktní třídu *BelkinWemoDevice*. Mezi atributy, které definuje abstraktní třída *BelkinWemoDevice*, patří číselný identifikátor zařízení (*DeviceID*) a zámek (*Poco::FastMutex*), který zajistí výlučný přístup k zařízení. Zámek je nutný z důvodu možné aktualizace IP adresy při hledání.

Třída představující chytrou žárovku přímo dědí a implementuje tuto abstraktní třídu. Instance žárovky, stejně jako ve skutečnosti, nemůže existovat bez prostředníka, který umožňuje ovládání žárovek skrz počítačovou síť. V tomto případě se jedná o Belkin WeMo Link, pro který je implementována třída *BelkinWemoLink*. Aby manažer zařízení mohl rozhodnout, která instance *BelkinWemoLink* se nepoužívá, a může být tedy odstraněna, obsahuje *BelkinWemoLink* počítadlo, počítající kolik instancí žárovek na něj vlastní referenci. Počítadlo se vždy aktualizuje, když vzniká nebo zaniká instance žárovky s ukazatelem na daný Belkin WeMo Link.

U inteligentní zásuvky a vypínače osvětlení existuje ještě jedna mezivrstva, a to *BelkinWemoStandaloneDevice*. Tato dvě zařízení se liší pouze ve zpracování odpovědi na požadavek získání aktuálních hodnot všech modulů. Tím pádem jsou v abstraktní třídě *BelkinWemoStandaloneDevice* implementovány metody pro získání MAC adresy, pro změnu stavu aktorových modulů a pro získání aktuálních hodnot všech modulů. Metoda pro zpra-

cování odpovědi s aktuálními hodnotami vrací celou odpověď, protože zpracování se liší pro obě zařízení, takže si jej implementují sami.

Pro vytváření SOAP zpráv odesílaných na inteligentní zařízení je definována třída *SOAPMessage*, která implicitně připraví nezbytné části zprávy. Mezi nezbytné části patří kořenový element envelope a v ní vnořený element body. Následně je možné pomocí *Poco::XML::XMLWriter*, který *SOAPMessage* poskytuje, vytvořit obsah zprávy. Pro analýzu příchozích zpráv jsou použity třídy z modulu XML z knihovny Poco. O odeslání HTTP požadavků obsahující SOAP zprávu se stará statická třída *HTTPUtil*, která využívá *Poco::Net::HTTPClientSession* pro přenos požadavků a příjem odpovědí.

Konfigurace modulu

Část konfiguračního souboru nastavující parametry Belkin WeMo modulu je zobrazen ve výpisu 5.1. Pomocí parametru `enable` lze celý modul zakázat, respektive povolit. Parametrem `upnp.timeout` je možné nastavit dobu, po kterou se maximálně čeká na odpovědi po odeslání vyhledávací SSDP zprávy. Pro podobné potřeby existuje parametr `http.timeout`, který určuje časový limit pro HTTP komunikaci. Časový interval mezi dvěma sběry dat z napárovaných zařízení je definován v parametru `refresh`.

```
[belkinwemo]
enable = yes
upnp.timeout = 5
http.timeout = 3
refresh = 10
```

Výpis 5.1: Konfigurace Belkin WeMo modulu

5.2.2 Modul pro řízení Philips Hue zařízení

Tento modul do aplikace *BeeeOn Gateway* přidává podporu řízení inteligentních zařízení nabízený firmou Philips Hue. Implementace modulu zcela odpovídá návrhu podpory Philips Hue v sekci 4.3. Hlavní části modulu je třída *PhilipsHueDeviceManager*, která zajišťuje povinnosti manažera zařízení.

Podobně jako u Belkin Wemo, se také uchovávají všechny instance reprezentující Philips Hue Bridge, které jsou nutné pro hledání nových žárovek a komunikaci s žárovkami skrz počítačovou síť. Tyto objekty se ukládají ekvivalentně s instancemi inteligentních zařízení, tudíž v *std::map*. Jediný rozdíl je ve vyhledávacím klíči, který je v tomto případě MAC adresa Philips Hue Bridge, protože pro něj není vytvářen číselný identifikátor.

Vyhledávání nových zařízení

Vyhledávání nových žárovek se skládá ze dvou kroků. V prvním kroku je nutné nejdříve najít prostředníka v komunikaci s žárovkami Philips Hue Bridge, a až poté v druhém kroku je možné se jej zeptat na dostupné žárovky. K objevení Philips Hue Bridgů slouží třída *UPnP* stejně jako u hledání Belkin WeMo zařízení. Pro každý nalezený Philips Hue Bridge se vytvoří instance třídy *PhilipsHueBridge*.

Následně se zkontroluje, zda už daný Philips Hue Bridge neexistuje v manažerovi zařízení. Pokud ano, tak se ověří, jestli se mu nezměnila IP adresa a případně se upraví.

U nově nalezeného Philips Hue Bridge je potřeba od něj získat unikátní identifikátor, bez kterého není možné provádět se zařízením všechny akce. Pokud se jedná o Philips

Hue Bridge, který byl nalezen v předchozím běhu aplikace, tak se identifikátor vytáhne z úložiště chráněného proti neoprávněnému přístupu (CredentialsStorage). Jinak se provede autorizace brány k Philips Hue Bridge, kdy je na něm nutné zmáčknout tlačítko uživatelem. Po úspěšné autorizaci se vygenerovaný identifikátor uloží do *CredentialsStorage*.

Když je správně nastaven identifikátor v instanci Philips Hue Bridge, tak se na něj odešle požadavek na hledání nových žárovek a následně se pošle požadavek na získání všech nalezených žárovek. Pro každou nalezenou žárovku se vytvoří instance příslušné třídy, reprezentující daný typ žárovky. Na základě existence záznamu o nalezené žárovce v manažerovi zařízení se rozhodne, jestli se do něj žárovka přidá. O nově objevených žárovkách se odešle příkaz *new-device* na server.

Práce s zařízeními

Pro každý typ žárovky je implementovaná vlastní třída, která zná její vlastnosti a formát některých vyměňovaných zpráv. Oproti návrhu je reálně implementovaná pouze *PhilipsHueDimmableBulb* třída, protože barevná varianta žárovky nebyla k dispozici, tudíž bylo nemožné vyzkoušet její chování.

Každá třída přestavující určitý typ Philips Hue osvětlení dědí abstraktní třídu *PhilipsHueBulb*, což umožní manažerovi zařízení jednotnou práci se všemi variantami žárovek. Abstraktní třída definuje, že každá podřazená třída obsahuje tyto atributy: číselný identifikátor zařízení, pořadové číslo žárovky na Philips Hue Bridge a ukazatel na příslušnou instanci *PhilipsHueBridge*.

Aby mohl manažer zařízení zjistit, která instance *PhilipsHueBridge* může být odstraněna, definuje její třída počítadlo obsluhovaných žárovek. Toto počítadlo je aktualizováno vždy, když se vytváří nebo zaniká instance žárovky s referencí na daný Philips Hue Bridge. Pro zpracování zpráv v JSON formátu jsou použity třídy z modulu JSON z knihovny Poco. O HTTP komunikaci se stejně jako u Belkin WeMo modulu stará statická třída *HTTPUtil*.

Konfigurace modulu

Konfigurace modulu odpovídá konfiguraci Belkin WeMo modulu 5.1 s rozdílem prvního řádku, který je v tomto případě `philipshue`.

5.2.3 Modul pro řízení BeeWi zařízení

Posledním vytvořeným modulem v rámci této práce je modul pro ovládání inteligentních zařízení od firmy BeeWi. Implementace modulu vychází z objektově orientovaného návrhu popsaného v sekci 4.4. Modul je řízen manažerem zařízení, který je implementován ve třídě *BeeWiDeviceManager*. V tomto případě se musí manažer zařízení starat kromě sběru aktuálních hodnot všech napárovaných zařízení a zpracovávání příkazu od serveru, také o problémy, které můžou nastat s Bluetooth USB donglem. Proto třída implementující manažera zařízení dědí abstraktní třídu *DongleDeviceManager*, která problémy s připojitelným zařízením řeší.

Vyhledávání nových zařízení

O prohledání Bluetooth sítě a následnou komunikaci s nalezenými zařízeními se stará třída *DBusHciInterface*. Tato funkcionalita třídy je realizována pomocí meziprocesové komu-

nikace se systémovou službou zajišťující komunikaci pomocí technologie Bluetooth skrz DBus[1]. Aplikační rozhraní k DBus je použito z výše zmíněné knihovny GLib.

Bluetooth daemon v roli DBus serveru se skládá z objektů, které se dále skládají z metod, vlastností, signálů a rozhraní. Relevantními objekty jsou *Adapter1*[2], *Device1*[3] a *GattCharacteristic1*[4]. *Adapter1* představuje Bluetooth rozhraní nabízející metody začni/skonči prohledávat síť, zapni/vypni rozhraní a další. Objekt *Device1* reprezentuje Bluetooth zařízení a obsahuje metody pro připojení/odpojení zařízení, získání jeho vzdáleného jména, získání jeho služeb a tak dále. Služby zařízení se skládají z charakteristik, které je možné číst nebo modifikovat pomocí *GattCharacteristic1* DBus objektu.

Popis těchto objektů je v XML formátu, jejichž ukázka je k dispozici v příloze B.1. Pro zjednodušení komunikace, s jednotlivými DBus objekty, jsou pomocí programu **gdbus-codegen**[5] vygenerovány C moduly používající DBus aplikační rozhraní z knihovny GLib, které obsahují funkce pro veškerou manipulaci s daným objektem. Toto generování se provádí až při překladu, tudíž součástí modulu jsou XML popisy používaných DBus objektů.

Jedním z požadavků na třídu *DBusHciInterface* byla možnost ji použít ve více modulech najednou a to z důvodu existence modulu pro zjištění prezence Bluetooth zařízení. Tento požadavek je třídou splněn díky dvěma zámkům. První zámek (*Poco::FastMutex*) slouží k výlučnému přístupu ke stavu Bluetooth rozhraní. Druhý zámek (*Poco::RWLock*) zabezpečuje situace:

- pouze jedno vlákno může v jednom okamžiku provádět objevování nových zařízení
- objevování nových zařízení a odesílání požadavků se provádí výlučně
- více vláken může odesílat požadavky v jednom okamžiku

Práce s zařízeními

Každá podporovaná varianta BeeWi inteligentního zařízení má svou třídu, která zná její moduly a implementuje způsob jejich ovládání. Kvůli jednotnému přístupu manažeru zařízení ke všem typům zařízení, existuje třída *BeeWiDevice*, která definuje jaké komponenty musí mít každá třída reprezentující BeeWi zařízení. Mezi atributy patří číselný identifikátor zařízení, který se vytváří z MAC adresy zařízení a číselného prefixu manažera zařízení, MAC adresa zařízení a časový limit, který se využívá například při připojování k zařízení nebo čtení hodnoty ze zařízení.

Modulem jsou aktuálně podporovány dvě BeeWi zařízení. Prvním z nich je senzor teploty a vlhkosti, pro který je vytvořena třída *BeeWiSmartClim* a druhým zařízením je inteligentní žárovka, kterou reprezentuje třída *BeeWiSmartLite*.

Konfigurace modulu

Mezi nastavitelné parametry modulu patří povolení modulu, kterým lze modul povolit či zakázat. Dalším parametrem `scan.timeout` lze definovat časovou dobu hledání nových zařízení na síti. Parametr `device.timeout` určuje časový limit při interakci se zařízením. Mezi interakce patří připojení k zařízení, čtení hodnot ze zařízení nebo zápis hodnot na

[1] https://www.freedesktop.org/wiki/Software/dbus/
[2] https://git.kernel.org/pub/scm/bluetooth/bluez.git/tree/doc/adapter-api.txt
[3] https://git.kernel.org/pub/scm/bluetooth/bluez.git/tree/doc/device-api.txt
[4] https://git.kernel.org/pub/scm/bluetooth/bluez.git/tree/doc/gatt-api.txt
[5] https://developer.gnome.org/gio/stable/gdbus-codegen.html

zařízení. Posledním parametrem `refresh` je, jako v předchozích modulech, časový interval, v kterém se mají sbírat data z napárovaných zařízení. Ukázka konfigurace modulu, vyňata z konfiguračního souboru aplikace, je k dispozici ve výpisu 5.2.

```
[beewi]
enable = yes
scan.timeout = 10
device.timeout = 5
refresh = 30
```

Výpis 5.2: Konfigurace BeeWi modulu

5.3 Překlad a spuštění

Pro překlad *BeeeOn Gateway* aplikace se používá nástroj **cmake**[6], který se také stará o překlad nově vytvořených modulů. Tento nástroj k překladu vyžaduje konfigurační soubor *CMakeLists.txt*, který byl rozšířen o záznamy nově vytvořených tříd. V rámci překladu aplikace lze definovat, které moduly se mají přeložit a které nikoliv. Pro úspěšný překlad nově vytvořených modulů je nutné mít korektně nainstalovány zmíněné knihovny.

Pro spuštění nově vytvořených modulů, bylo potřeba rozšířit konfigurační soubory aplikace o nově vytvořené manažery zařízení. Tyto konfigurační soubory předané aplikaci při spuštění jsou zpracovány poskytovatelem závislostí (*DependencyInjector*), který zajistí vytvoření všech potřebných vláken a objektů.

5.4 Možnosti rozšíření

Všichni zmínění výrobci inteligentních zařízení nabízí další chytré prvky domácnosti, pro které je možné vytvořit podporu v systému *BeeeOn*. U Philips Hue to může být například inteligentní lampa nebo chytrý LED pásek. Pro integraci podpory nového zařízení je potřeba vytvořit třídu, která bude reprezentovat nové zařízení a přidat způsob jeho hledání do manažera zařízení. Každá třída představující chytré zařízení musí implementovat abstraktní třídu zařízení pro daný modul, což umožní manažeru zařízení pracovat se všemi zařízeními stejně.

Dalším možným rozšířením je vylepšení UPnP vyhledávání, které momentálně umožňuje objevování pouze jednoho typu zařízení v jednom okamžiku. Schopnost vyhledávat více typů zařízení najednou by se hodilo do Belkin WeMo manažera zařízení, kde se hledají tři typy zařízení. S tímto vylepšením by se jedno prohledání sítě zrychlilo trojnásobně.

[6] https://cmake.org

Kapitola 6

Testování

Tato kapitola se věnuje způsobu ověřování správnosti implementace nově vytvořených modulů. Součástí kapitoly je popis případů užití, za pomocí kterých se validovala funkčnost nových manažerů zařízení.

V prvotní fázi, kdy ještě chybělo propojení brány se serverem, bylo možné bránu testovat pouze pomocí komponenty *TestingCenter*. Komponentě je při spuštění brány vytvořené vlastní vlákno, v kterém naslouchá na nakonfigurované IP adrese a portu, a čeká na nová připojení. Připojit se k běžící instanci *TestingCenter* lze například pomocí programu **nc**. Po připojení je na něj možné odesílat příkazy, a to všechny dříve zmíněné v této práci 2.5.1. Následně, po zprovoznění komunikace mezi bránou a serverem, bylo možné testovat implementaci brány pomocí mobilní aplikace.

K testování nově vytvořených manažerů zařízení byla využita komponenta brány *TestingCenter* a mobilní aplikace. Testování chování manažerů zařízení bylo prováděno převážně manuálně. Testovanými případy užití, kterými musel každý manažer zařízení projít, byli následující:

1. Spustit aplikaci brány a odeslat na ní příkaz *listen*. Ověřuje implementaci hledání nových zařízení a informování o jejich nalezení.

2. Odeslat příkaz *listen*, když už se objevování nových zařízení provádí.

3. Napárovat a odpárovat nalezené zařízení. Testuje zpracování příkazu *device-accept* a *unpair* manažerem zařízení.

4. Napárovat nalezené zařízení a modifikovat hodnotu všech jeho aktorových modulů. Ověřuje schopnost manažera zařízení přijmout příkaz *set-value* a zároveň testuje komunikaci brány s inteligentním zařízením.

5. Napárovat nalezené zařízení a sbírat data z jeho senzorových modulů. Ověřuje manažera zařízení jak v komunikaci s chytrým zařízením, tak v odesílání nasbíraných dat.

6. Restartovat bránu, která měla už napárovaná nějaká zařízení. Zde se testuje, jestli se pokusí manažer zařízení při startu tato zařízení nalézt.

Pro snadnější testování chování manažera zařízení, bez nutnosti fyzické přítomnosti inteligentního zařízení, byly napsány skripty emulující daná zařízení. Emulující skripty nebylo

možné napsat pro zařízení, které komunikují pomocí Bluetooth, protože Bluetooth nepodporuje lokální smyčku. Skripty jsou napsány v jazyce Python verze 3. Emulační skript byl vytvořen pro Belkin WeMo zásuvku, Belkin WeMo vypínač osvětlení, Belkin WeMo Link a Philips Hue Bridge. Každý skript se při spuštění nejprve přihlásí do multicastové skupiny UPnP, kde vyčká než ho aplikace brány najde. Po přijmutí SSDP vyhledávací zprávy a odeslání odpovědi, se skript přepne do role HTTP serveru. HTTP server je realizován pomocí tříd *http.server.HTTPServer* a *http.server.BaseHTTPRequestHandler* ze standardní knihovny jazyka Python. V tomto stavu skript čeká na HTTP požadavky, které při přijetí zpracuje a vytvoří k nim příslušné odpovědi. Skripty emulující jednotlivá zařízení dokáží reagovat na zprávy, které byli analyzovány dříve v kapitole 3. Aktorové moduly zařízení jsou realizovány skriptem pomocí proměnných, což umožňuje nastavovat hodnoty modulům a získávat jejich aktuální hodnoty.

Pro vybrané části, jako například metody pro analýzu příchozích zpráv z BeeWi zařízení nebo pro převod hodnot modulů na hodnoty předepsané konvencí v systému *BeeeOn* a zpět, jsou napsány jednotkové testy, které kontrolují jejich správnost. Pro vytvoření těchto testů byl použit framework **CppUnit**.

Kapitola 7

Závěr

V rámci této práce byli úspěšně vytvořeny tři nové moduly do systému *BeeeOn*. Úkolem těchto modulů je řídit určitý typ inteligentních zařízení. Celkem do množiny podporovaných zařízení systému *BeeeOn* přibylo 6 zařízení od 3 různých výrobců. Každý z vytvořených modulů je navržen pro jednoduché přidání podpory dalších zařízení. Toto je velmi žádoucí, protože existují stále zařízení, která zmíněné firmy nabízejí a není pro ně vytvořena podpora v systému *BeeeOn*.

V prvé řadě bylo potřebné nastudovat architekturu systému *BeeeOn*, konkrétněji aplikaci brány, která byla v rámci této práce rozšiřována o novu funkcionalitu. Následně bylo nutné analyzovat chování a komunikační protokol vybraných inteligentních zařízení, pro která byla vytvořena podpora. Byly zkoumány tři typy zařízení podle použité technologie pro komunikaci. Jako první byla zařízení, které používají počítačovou síť. Dalšími byla inteligentní zařízení, která potřebují prostředníka (Belkin WeMo Link a Philips Hue Bridge) v komunikaci s nimi. Tento prostředník je připojen do počítačové sítě a jeho úkolem je přeposílat požadavky na konkretní zařízení technologií ZigBee. Poslední typ zařízení používá pro komunikaci technologii Bluetooth.

Navrhl jsem nové moduly, které zajistí řízení konkretního druhu zařízení. Poté byly podle tohoto návrhu nové moduly implementovány a integrovány do aplikace *BeeeOn Gateway*. Část textu věnující se implementaci také popisuje, co je potřeba implementovat, při přidávání podpory nového zařízení do daného modulu. Nakonec byly nově vytvořené moduly otestovány, kdy každý modul musel úspěšně vykonat množinu případů užití popsaných v kapitole o testování.

Literatura

[1] BeeeOn - Adaptér [online]. https://antdev.fit.vutbr.cz/redmine/projects/adapter/wiki, [cit. 2018-01-18].

[2] GATT Overview [online]. https://www.bluetooth.com/specifications/gatt/generic-attributes-overview, [cit. 2018-02-04].

[3] Philips Hue API Documentation [online]. https://www.developers.meethue.com/philips-hue-api, [cit. 2018-01-22].

[4] Applied Informatics Software Engineering GmbH and Contributors: A Guided Tour Of The POCO C++ Libraries [online]. https://pocoproject.org/docs/00100-GuidedTour.html, [cit. 2018-02-19].

[5] BOX, Don, et al.: Simple Object Access Protocol (SOAP) 1.1 [online]. https://www.w3.org/TR/soap11/, 2000, [cit. 2018-01-18].

[6] PRESSER, Alan, et al.: UPnP Device Architecture 1.1 [online]. http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf, 2008, [cit. 2018-01-18].

[7] The GNOME Project: Referenční příručka k API [online]. https://developer.gnome.org/references, [cit. 2018-04-18].

[8] TOWNSEND, Kevin: Introduction to Bluetooth Low Energy [online]. https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction, [cit. 2018-02-04].

Přílohy

Příloha A

Ukázka UPnP popisu

```
<?xml version="1.0"?>
<root xmlns="urn:Belkin:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
<deviceType>urn:Belkin:device:controllee:1</deviceType>
<friendlyName>WeMo Switch</friendlyName>
    <manufacturer>Belkin International Inc.</manufacturer>
    <manufacturerURL>http://www.belkin.com</manufacturerURL>
    <modelDescription>Belkin Plugin Socket 1.0</modelDescription>
    <modelName>Socket</modelName>
    <modelNumber>1.0</modelNumber>
    <modelURL>http://www.belkin.com/plugin/</modelURL>
<serialNumber>221332K1100068</serialNumber>
<UDN>uuid:Socket-1_0-221332K1100068</UDN>
    <UPC>123456789</UPC>
    <iconList>
      <icon>
        <mimetype>jpg</mimetype>
        <width>100</width>
        <height>100</height>
        <depth>100</depth>
         <url>icon.jpg</url>
      </icon>
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:Belkin:service:WiFiSetup:1</serviceType>
        <serviceId>urn:Belkin:serviceId:WiFiSetup1</serviceId>
        <controlURL>/upnp/control/WiFiSetup1</controlURL>
        <eventSubURL>/upnp/event/WiFiSetup1</eventSubURL>
        <SCPDURL>/setupservice.xml</SCPDURL>
```

```
      </service>
      <service>
        <serviceType>urn:Belkin:service:basicevent:1</serviceType>
        <serviceId>urn:Belkin:serviceId:basicevent1</serviceId>
        <controlURL>/upnp/control/basicevent1</controlURL>
        <eventSubURL>/upnp/event/basicevent1</eventSubURL>
        <SCPDURL>/eventservice.xml</SCPDURL>
      </service>
    </serviceList>
   <presentationURL>/pluginpres.html</presentationURL>
</device>
</root>
```

Výpis A.1: Ukázka části UPnP popisu inteligentní zásuvky Belkin WeMo

```
<?xml version="1.0"?>
<scpd xmlns="urn:Belkin:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>SetBinaryState</name>
      <argumentList>
        <argument>
          <retval />
          <name>BinaryState</name>
          <relatedStateVariable>BinaryState</relatedStateVariable>
          <direction>in</direction>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetFriendlyName</name>
      <argumentList>
        <argument>
          <retval />
          <name>FriendlyName</name>
          <relatedStateVariable>FriendlyName</relatedStateVariable>
          <direction>in</direction>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetMacAddr</name>
    </action>
    <action>
      <name>GetSerialNo</name>
```

```
  </action>
  <action>
    <name>ChangeFriendlyName</name>
    <argumentList>
      <argument>
        <retval />
        <name>FriendlyName</name>
        <relatedStateVariable>FriendlyName</relatedStateVariable>
        <direction>in</direction>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>GetBinaryState</name>
    <argumentList>
      <argument>
        <retval/>
        <name>BinaryState</name>
        <relatedStateVariable>BinaryState</relatedStateVariable>
        <direction>out</direction>
      </argument>
    </argumentList>
  </action>
</actionList>
<serviceStateTable>
  <stateVariable sendEvents="yes">
    <name>BinaryState</name>
    <dataType>Boolean</dataType>
    <defaultValue>0</defaultValue>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>FriendlyName</name>
    <dataType>string</dataType>
    <defaultValue>0</defaultValue>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>MacAddr</name>
    <dataType>string</dataType>
    <defaultValue>0</defaultValue>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>SerialNo</name>
    <dataType>string</dataType>
    <defaultValue>0</defaultValue>
  </stateVariable>
</serviceStateTable>
```

```
</scpd>
```

Výpis A.2: Ukázka části UPnP popisu jedné služby inteligentní zásuvky Belkin WeMo

Příloha B

Ukázka popisu DBus objektu

```
<!DOCTYPE node PUBLIC "-//freedesktop//DTD D-BUS Object Introspection 1.0//EN"
  "http://www.freedesktop.org/standards/dbus/1.0/introspect.dtd">
<node>
  <interface name="org.bluez.Device1">
    <method name="Disconnect"></method>
    <method name="Connect"></method>
    <method name="ConnectProfile">
      <arg name="UUID" type="s" direction="in"/>
    </method>
    <method name="DisconnectProfile">
      <arg name="UUID" type="s" direction="in"/>
    </method>
    <method name="Pair"></method>
    <method name="CancelPairing"></method>
    <property name="Address" type="s" access="read"></property>
    <property name="Name" type="s" access="read"></property>
    <property name="Alias" type="s" access="readwrite"></property>
    <property name="Class" type="u" access="read"></property>
    <property name="Appearance" type="q" access="read"></property>
    <property name="Icon" type="s" access="read"></property>
    <property name="Paired" type="b" access="read"></property>
    <property name="Trusted" type="b" access="readwrite"></property>
    <property name="Blocked" type="b" access="readwrite"></property>
    <property name="LegacyPairing" type="b" access="read"></property>
    <property name="RSSI" type="n" access="read"></property>
    <property name="Connected" type="b" access="read"></property>
    <property name="UUIDs" type="as" access="read"></property>
    <property name="Modalias" type="s" access="read"></property>
    <property name="Adapter" type="o" access="read"></property>
    <property name="TxPower" type="n" access="read"></property>
    <signal name="PropertiesChanged">
      <arg name="interface" type="s"/>
      <arg name="changed_properties" type="a{sv}"/>
      <arg name="invalidated_properties" type="as"/>
```

```
    </signal>
  </interface>
</node>
```

Výpis B.1: Popisu DBus objekty reprezentující Bluetooth zařízení