VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

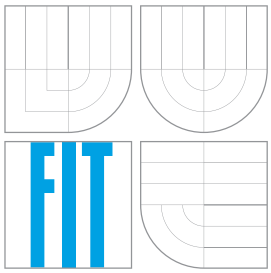
EMULÁTOR KONCOVÝCH PRVKŮ INTELIGENTNÍ DOMÁCNOSTI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE HANA BRYCHTOVÁ
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EMULÁTOR KONCOVÝCH PRVKŮ INTELIGENTNÍ DOMÁCNOSTI

EMULATOR FOR INTELLIGENT HOME END STATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — HANA BRYCHTOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVOL KORČEK
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce popisuje návrh a implementaci aplikace s grafickým rozhraním emulující činnost koncových zařízení systému inteligentní domácnosti. Tato zařízení slouží k měření veličin a ovládání spínačů. Aplikace je řešena pomocí grafických a síťových knihoven Qt v jazyce C++. Výsledkem je program, který umožňuje jednoduché testování v uživatelsky přívětivém prostředí. Přínosem této práce je umožnění paralelního vývoje serverové aplikace a jejího testování s vývojem hardwarových koncových zařízení.

Abstract

This thesis covers design and implementation of an application with a graphic user interface emulating the behaviour of end stations for intelligent home. These devices are used for measuring physical quantities and controlling switches. The application is implemented in C++ using Qt Framework. The final product is an application which provides a simple way of testing in a friendly user interface. The main benefit is in enabling a parallel developement of the server application and the developement of the hardware end station devices.

Klíčová slova

síťová aplikace, emulátor, inteligentní domácnost, grafické rozhraní, testování síťové komunikace

Keywords

network application, emulator, intelligent home, graphic user interface, testing of communication over network

Citace

Hana Brychtová: Emulátor koncových prvků inteligentní domácnosti, bakalářská práce, Brno, FIT VUT v Brně, 2014

Emulátor koncových prvků inteligentní domácnosti

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Pavola Korčeka. Uvedla jsem všechny zdroje a publikace, ze kteých bylo při psaní čerpáno.

.....
Hana Brychtová
20. května 2014

Poděkování

Ráda bych poděkovala svému vedoucímu práce Ing. Pavolu Korčekovi za jeho odborné vedení, trpělivost a pohotové odpovědi. Dále bych chtěla poděkovat celé výzkumné skupině, pod jejíž záštitou byl tento projekt řešen.

Obsah

Kapitola 1

Úvod

Internet poskytuje mnoho nových možností a prostor pro vývoj nových technologií. Stále více objektů reálného světa získává svou virtuální variantu. Napojením na Internet získáváme stálý přísun informací a dává nám větší možnosti manipulace s realitou. Vývoj postupuje rychle a vznikají celé systémy napojené do sítě a jedním takovým je i dále popsaný projekt.

Tato práce je řešena pro výzkumnou skupinu zabývající se projektem inteligentní domácnosti. V současnosti se do sítě zapojuje čím dál více zařízení, ať už jde o domácí spotřebiče, osvětlení nebo vytápění. Pomocí různých senzorů můžeme o celém objektu získávat mnoho informací a z nich počítat statistiky či další údaje zlepšující systém. Tato idea je stále rozšířenější, protože má mnoho výhod. Za prvé můžeme hovořit o úspoře energií a tím zlepšením výdajů domácnosti. Moderní technologie nám umožňují například automatizaci vytápění, regulaci intenzity světla a další. Další z výhod je komfort uživatele hned v několika směrech. První je skutečnost, že v současnosti je většina prvků bezdrátových a umožňují jednoduché zapojení. Také je díky připojení na Internet možné ovládat více zařízení současně nebo vzdáleně např. bude možné zapnout vytápění domu před naším příjezdem nebo vypnout spotřebič, který zůstal zapnutý. Lze také dosáhnout většího zabezpečení celého objektu například díky okamžitého upozornění od senzorů detekující výjimečné stavy (např. kouř, voda, ...).

Přenos informací samotných je však spíše bezpečnostním rizikem, a proto musí být veškerá komunikace šifrována, aby nemohl výhody systému někdo jednoduše zneužít. Stejně tak má toto řešení mnoho problémů způsobených použitými technologiemi a prostředím. Je třeba minimalizovat rušení, jelikož je většina komunikujících prvků bezdrátových. S tímto souvisí zajištění dostatečného dosahu a kvality signálu pro všechny zapojené prvky, aby se dal systém použít v různě velkých objektech.

Jelikož bude mnoho zařízení napájených bateriemi, musíme zajistit jejich včasnou výměnu, aby byl systém stále funkční a stejně tak optimalizovat používání baterie tak, aby měla co nejdelší životnost (např. mechanismus buzení u senzorů). Kvůli těmto a dalším problémům je třeba systém před jeho nasazením dostatečně otestovat, což je i hlavním tématem této bakalářské práce. Výsledná aplikace testuje komunikaci mezi koncovými prvky a serverem a tím urychlí vývoj serveru, který by byl jinak možný vyvíjet až v době, kdy budou dostupné hardwarové prvky. Grafické rozhraní emulátoru také oproti konzolové aplikaci urychlí a zjednoduší testování.

Kapitola 2

Inteligentní domácnost

Pojem Inteligentní domácnost, neboli automatizace domácnosti, znamená integraci technologií do domácnosti, správu všech jejích automatizovaných prvků, ale také pokrývá služby, které lze díky automatizaci do domu přidat. [3]

Tato kapitola se zabývá srovnáním dvou systémů inteligentní domácnosti. Jedním z nich je jeden z komerčních systémů, který je nabízen na trhu a druhým systém výzkumné skupiny na FIT VUT. Popsány jsou oblasti architektury systému, použitých technologií i možností využití. Nakonec jsou u každého zhodnoceny jeho výhody a nevýhody.

2.1 Aktuální stav

V současné době existuje mnoho českých i zahraničních společností a projektů zabývajících se tématem inteligentní domácnosti. Pro další výzkum a vývoj v této oblasti je proto důležité srovnání a znalosti aktuální situace a nabídky. Existuje i několik opensource – tzv. otevřených, projektů, které mohou sloužit nejen pro inspiraci, ale také se nabízí možnost spolupráce v různých oblastech.

V této sekci je popsán systém jedné z těchto společností – jde o českou firmu zabývající se inteligentními domy. Toto řešení je zde detailně popsáno a následně srovnáno se systémem vyvíjeným výzkumnou skupinou na FIT VUT.

Komerční systém inteligentní domácnosti od společnosti CannyNet

Jde o systém s velkou modularitou a možnosti rozšiřitelnosti. Ovládání je řešeno přes webovou aplikací, čímž je dosaženo přenositelnosti, a je možné sdružovat různá nastavení jednotlivých prvků. Na stránkách je o systému řečeno, že s jeho používáním lze dosáhnout úspory až 30% [2]. Konkrétně např. snížením intenzity osvětlení či pomocí systému regulace vytápění, které umí vyhodnotit otevřené/zavřené okno v místnosti a podle toho sepnout topení. Systém je ovladatelný i vzdáleně, je-li tak nastavený, a nabízí i grafy spotřeby energií.

Architektura

Systém je řízen jedinou řídící jednotkou, která je napájena přímo z elektroinstalační sítě, a i při selhání této jednotky je systém ovladatelný manuálně. Prvky jsou k ní připojeny standardním elektroinstalačním rozvodem nebo nezávislou dvoudrátovou sběrnicí. To dovoluje podstatně větší dosah než pomocí bezdrátových technologií a komunikace je odolnější proti

rušení. Multimediální prvky či kamery instalované v domácnosti aj. mohou být připojeny přes *Ethernet* či přes domácí *WiFi* síť.

Nejvyšší vrstvu, tj. koncové stanice tvoří chytré telefony, tablety, PC a ostatní zařízení s přístupem na Internet. Ovládání je řešeno webovou aplikací, takže je třeba mít přístup chráněn proti zneužití. Nacházímeli se přímo v objektu, k zabezpečení komunikace postačí šifrování komunikace přes *WiFi*, které je tu řešeno mechanismem *WPA2PSK/AES*.

Při vzdáleném přístupu je používán protokol *HTTPS*. Webová aplikace je navíc chráněna uživatelským jménem a heslem a dále při provádění nejdůležitějších operací z hlediska zabezpečení (což je např. deaktivace alarmu) je vyžadován navíc ještě čtyřmístný PIN.

Protokoly

V rámci elektroinstalační sítě je používán protokol *PLCBUS*, který podporuje *Full Duplex*, rychlou odezvu a možnost adresovat velkého množství modulů. Pomocí něj se získávají informace o stavu zařízení vypnuto/zapnuto. Každý modul má pak svou unikátní adresu.

Protokol *MODBUS* je protokol umožňující přenos po různých sítích a sběrnicích, definuje strukturu zprávy nezávisle na typu komunikační vrstvy. Výsledná zpráva pak udává operaci, kterou má server provést.

Na transportní a síťové vrstvě je používáno *TCP/IP*, na aplikační pak protokol *HTTPS*. Přenášená data jsou dále šifrována pomocí *SSL* nebo *TLS*.

Instalace systému

Instalace systému je nabízena jak do novostaveb, tak do stávajících zástaveb a to bez nutnosti přidat mnoho další kabeláže. Samotná instalace probíhá umístěním řídící jednotky do rozvaděče nebo technické místnosti a jejím připojením do elektrického okruhu. Do každého obvodu sítě se poté nainstalují mikromoduly starající se o spínání a stmívání. Je nabízeno několik variant řešení instalace termostatů a to buď základní stanici s bezdrátovými termostaty, nebo standardní termostat připojený vodiči. Speciální variantou je pak virtuální termostat ovládaný pomocí koncové stanice, například tabletu a neruší interiér v místnosti jsou pouze miniaturní čidla připojená vodiči. Poté je třeba jednotlivé mikromoduly spárovat s hlavní řídící jednotkou pomocí stisku tlačítka mikromodulu a poté tlačítka ve webovém rozhraní.

Variabilita

Systém nabízí celkem pět modulů, které můžeme jakkoliv kombinovat a přidávat.

Prvním je modul zahrnující osvětlení domu a ovládání elektrických zařízení. Každý světelný okruh v domě je vybaven mikromodulem který se stará o spínání, stmívání či nastavování intenzity světla. Může být nainstalován přímo pod standardní vypínač nebo samostatně v podobě krystalového vypínače. Mikromodul je poté třeba spárovat v aplikaci s příslušným tlačítkem ve webové aplikaci. Tyto tlačítka je možné poté sdružovat pomocí funkce scéna. Tento modul zahrnuje i ovládání elektrických zařízení jako např. žaluzií či garážových vrat, které fungují podobně.

Dalším modulem je regulace a ovládání teploty. Dům je vybaven sítí senzorů, které odesílají pravidelně data (na stránkách je uvedeno 24 hodin denně, 356 dní v roce) a je možné zobrazit informace o aktuální spotřebě energie s pohledem do historie. Dále systém disponuje i aktivními termostaty, které se starají o spínání. Tyto prvky jsou připojeny pomocí technologie *Ethernet*. Komunikace probíhá tak, že jakmile systém zaznamená změnu

hodnoty od přednastavené, je vyslán příkaz pro sepnutí daného tepelného okruhu a po jejím dosažení se okruh opět vypne.

Třetím modulem jsou multimédia a jeho základem je centrální úložiště dat, které je automaticky zálohováno. Tento server lze využít i pro služby jako je *FTP*, email či webhosting. Samozřejmostí je ovládání zařízení určených k přehrávání. Ve stavu, kdy nedochází k zápisu či čtení dat, je úložiště uvedeno do stavu hibernace. K přehrávání multimediálního obsahu je využívána technologie *XBMC*[1].

Modul zabezpečení se stará o alarmy a čidla, která detekují různá ohrožení domácnosti (např. kouř, únik plynu a další). Je také možné nastavit systém tak, aby byla simulována přítomnost majitele náhodným rozsvěcením a zhasínáním světel při dlouhodobější nepřítomnosti v objektu.

Posledním modulem je vizuální zabezpečovací systém. Kamery připojené přes *Ethernet* nebo *WiFi* posílají záznamy na centrální úložiště a zároveň je dostupný streaming. Lze zajistit i záložní napájení kamer v případě výpadku elektřiny.

Výhody a nevýhody

Velká výhoda systému spočívá v jeho variabilitě, lze nainstalovat jen některé moduly, nebo jsou nabízeny i balíčky. Jeho výhodou je i to, že je to systém velmi komplexně pokrývající všechny možnosti moderní inteligentní domácnosti a je funkční i případě poruchy všechny vzdáleně ovladatelné prvky jsou ovladatelné i manuálně. Samotná webová aplikace je výhodná svou přenositelností. Nepotřebujeme instalovat žádné programy či aplikace na koncové stanice, stačí webový prohlížeč. Další výhodou jsou i možnosti samotné aplikace sdružování různé funkcionality do scén nebo nastavení různých režimů.

Problémem je vyžadování neustálého připojení k Internetu a veškeré komunikace přes něj. Takže i zařízení nacházející se v lokální síti nebudou schopny ovládat systém v případě přerušení připojení na Internet. Další nevýhodu je přístup na webovou aplikaci přes veřejnou adresu, kterou musíme znát.

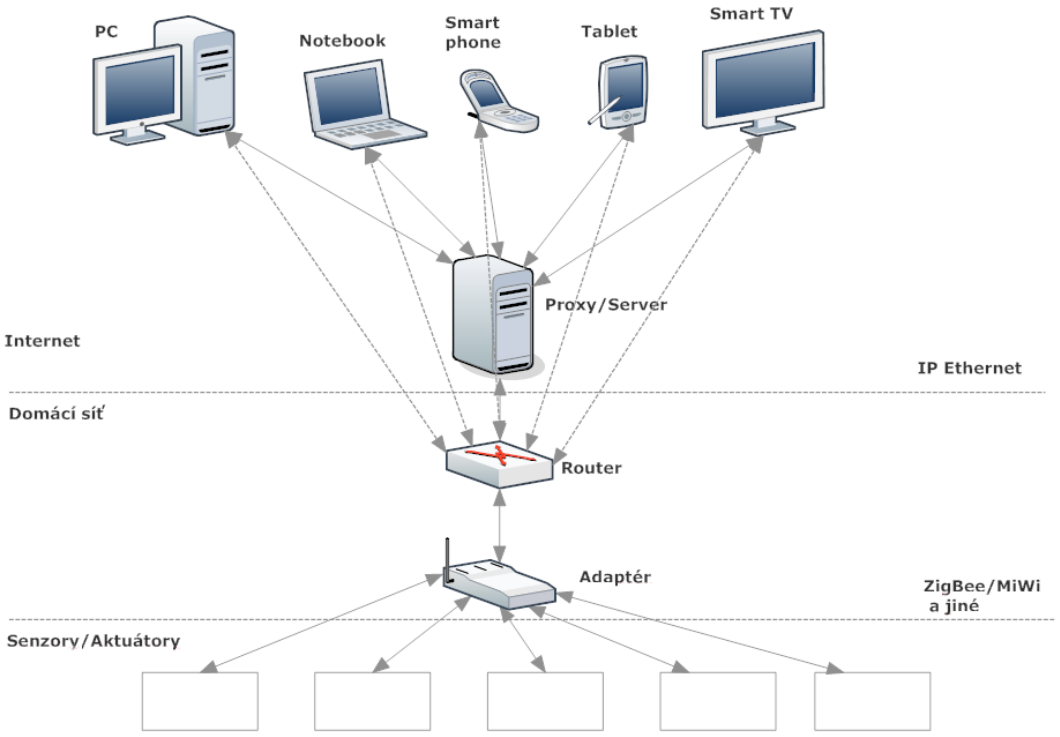
2.2 Systém na FIT VUT

Systém má vícevrstvou architektu [7] – jak je vyobrazeno na obrázku 2.1. Nejnižší vrstvu tvoří koncové bezdrátové prvky, které jsou dvojího typu: senzory a aktuátory. Senzory jsou prvky, které zaznamenávají nějakou veličinu ta je používána vyšší vrstvou systému k dalšímu zpracování. Aktuátory jsou prvky, které určitým způsobem zasahují do prostředí tak, že v určitých nastavených intervalech dojde k jejich buzení a provádění jejich činnosti.

Komunikace probíhá pomocí bezdrátového protokolu *MiWi* od společnosti *Microchip*, ale tyto prvky se musí nejdříve spárovat s adaptérem, který tvoří další vrstvu. Spárování prvků se může dít i opakovaně a mělo by být z hlediska uživatele jednoduché, například pomocí stisku tlačítka. Při spárování jsou po síti posílána data s informací o jednoznačné identifikaci prvku ale předpokládá se, že do sítě budou připojeny i komerční prvky, které mohou mít komunikaci řešenou jinak, což bude řešeno na úrovni adaptéru. Komunikace musí být kvůli možnému odposlechu šifrována.

Další informací odesílanou na adaptér je aktuální stav baterie, aby mohlo dojít k její včasné výměně. Lze předpokládat, že některé prvky s malou spotřebou budou mít baterii vestavěnou, nebude se tedy moct vyměnit, ale i v tomto případě je vhodné být o stavu informován. Jedinou výjimkou jsou jedině prvky přímo napájené z elektrického zdroje.

[1] *XBMC* je multimediální opensource software, který je zdarma ke stažení. [1]



Obrázek 2.1: Architektura systému inteligentní domácnosti vyvíjené na FIT VUT.

Další vrstvou je adaptér. Je také jednoznačně identifikován, protože můžeme předpokládat, že jich může být v rámci jedné domácnosti více, a ve vyšší vrstvě musí být rozlišitelné. Adaptér má za úkol přijímat data od senzorů a přeposílat je na router, případně nějaká data uchovávat pro případ, že dojde k výpadku připojení na Internet a při následném znovupřipojení je ze své paměti odeslat. Komunikace ale probíhá výhradně s těmi prvky, které byly s adaptérem spárovány. Adaptér neřeší, kdy dostane od prvků další data, musí být tedy neustále schopný je přijímat.

Propojení adaptéru a routeru může být řešeno několika způsoby za prvé může být vestavěný přímo v routeru, to však znamená zásahy jak do hardware tak do software routeru, což je problém, protože ne každý výrobce dovoluje přidání jisté funkcionality. Toto můžeme vyřešit pomocí kompletní přeinstalace operačního systému, tam ovšem nastává problém s možnou ztrátou původní funkcionality routeru. Druhou variantou je připojení adaptéru přes rozhraní *USB*. Toto řešení je lepší, protože oproti předchozímu nevyžaduje zásah do hardware routeru, ale stále vyžaduje minimálně přidání ovladače do systému v routeru. Třetí variantou je pak připojení adaptéru přes *Ethernet* nebo *WiFi*, kde není třeba do routeru nijak zasahovat. Při připojení přes WiFi ale musíme nastavit SSID a WPA2 heslo, jeli vyžadováno. V tomto případě je ale nutné ještě vyřešit napájení adaptéru. Adaptér bude ale muset mít v každém případě možnost aktualizovat software, jelikož se předpokládá přidávání nových koncových prvků od komerčních výrobců.

Bránou do Internetu je adaptér společně s routerem. K routeru se jde připojit pomocí veřejné IP adresy ze všech koncových stanic, které tvoří vrchní vrstvu systému. Nevýhodou

je fakt, že musíme vždy tuto adresu znát. Dalším problémem je, že při výpadku připojení k Internetu se stanice nepřipojí ani k zařízením, které jsou s nimi v lokální síti. Musí používat další adresu, kterou je třeba si zapamatovat. Stejně tak je používání domácí adresy výhodnější pro stanice trvale umístěné v domácí síti. To všechno znamená další režii na straně koncové aplikace, která musí rozpoznat ve které síti se stanice nachází. Poslední nevýhodou je fakt, že veškeré informace a logika domácnosti i s databází prvků musí být uložena přímo v routeru, což by se řešilo opět zásahem do jeho software.

Router nemusí být na Internetu také reprezentován veřejnou adresou, stanice přistupující k domácnosti zvenku by tedy neměly šanci se připojit. Jako řešení bylo zvolení proxy serveru jako mezivrstvy mezi routerem a koncovými stanicemi. Tento server disponuje veřejnou adresou a je možné se k němu připojit neřeší ovšem problémy zmíněné předtím.

Řešením je přesunutí logiky domácnosti na tento server, stejně tak databázi. Dále může poskytovat další služby, které se odvíjí od získaných dat. Největší výhodou je určitě fakt, že dále není třeba pamatovat si veřejnou adresu, ale koncové stanice jsou pouze přihlášeny do distribuované služby. Problémem ale zůstává nutnost připojení k Internetu, ošetření zabezpečení přenášených dat či možná duplicita uložení certifikátu, který bude nejen v routeru/adaptéru, ale také na serveru.

Kapitola 3

Koncová zařízení

Aplikace, kterou popisuje tato bakalářská práce, emuluje bezdrátové koncové prvky tvořící nejnižší vrstvu systému. Některá z těchto hardwarových zařízení budou vyvíjena výzkumnou skupinou, ale také se počítá s nasazením prvků od komerčních výrobců, což znamená vysokou variabilitu protokolů a parametrů. Už z tohoto důvodu jsou emulovaná zařízení od reálných koncových prvků značně abstrahována. Dalším důvodem je, že jelikož je hlavním účelem této aplikace simulace síťové komunikace, nezaměřujeme se na jiné detaily a vlastnosti hardwarových prvků.

Tato kapitola se zabývá rozdělením koncových prvků, míry abstrakce emulovaných zařízení vůči fyzickým prvkům a generováním dat posílaných po síti.

3.1 Typy koncových prvků

Aplikace má za úkol emulovat koncová zařízení – v systému rozlišujeme dva jejich typy – a to senzory a aktory či aktuátory. V návrhu aplikace pak rozlišujeme typů více, ale jde pouze o kombinace těchto dvou základních typů. U těchto typů budeme předpokládat, že každý z nich měří právě jednu veličinu či ovládá právě jednu jednotku – více měřených typů budeme chápat jako dva koncové prvky. To se odlišuje od návrhu aplikace, kde chápeme typ jako množinu těchto koncových zařízení. V této sekci se ale zabýváme reálnými fyzickými zařízeními zapojenými do sítě.

Senzor

Senzor je bezdrátový prvek v síti, jehož úkolem je měřit veličinu či snímat pozici spínače a tuto informaci posílat na server. Jelikož tato zařízení nebudou trvale připojena v síti a z ní napájena, největší důraz je v hardwarovém návrhu kladen na výdrž baterie. Tyto senzory se přednastaveném intervalu zapínají a odesílají hodnoty na server, a po přijetí odpovědi se vypnou. Přičemž budou asi čekat jen určitou dobu, aby se baterie příliš nezatížila, došlo by-li například k výpadku připojení. Interval, ve kterém se budou zapínat neboli budit, bude možné měnit odpovědí na jejich zprávy.

Aktor

Aktor nebo aktuátor je koncový prvek, který umožňuje vzdálenou správu určitého zařízení. Jde o možnost vzdáleně nastavit například polohu elektricky ovladatelných rolet či zapnutí

vytápění – tedy nastavení nějakého přepínače. Tyto prvky jsou napájeny ze sítě, protože potřebují být neustále k dispozici na přepínání ze serveru – respektive od uživatele.

3.2 Rozdíly emulovaných prvků a hardwarových zařízení

V emulátoru jsou koncové prvky řešeny softwarově pomocí objektů. Tento objekt je množina koncových prvků vystupující v síti jako jedno zařízení. Podle koncových prvků, které množina obsahuje je pak objekt typu senzor, aktor či senzor/aktor, jak je popsáno v sekci 4.3.

Zatímco fyzické koncové prvky budou veličinu doopravdy měřit a ukládat do své paměti, u emulovaných prvků proběhne výpočet náhodné hodnoty z rozsahu zadaného uživatelem. Stejně tak nebude probíhat měření stavu baterie a kvality signálu daného prvku.

Buzení a připojování prvků je řešeno pomocí mechanismů knihoven Qt – pomocí signálů a slotů, nedojde tedy přímo k odpojení prvku ze sítě – v případě senzoru, ale pouze k pozastavení činnosti. Emulované prvky také neobsahují mechanismy pro reset či odpojení ze sítě.

3.3 Generovaní náhodných hodnot

Tato sekce se zabývá hodnotami, které emulované koncové prvky posílají. Z hlediska testování bylo nutné, aby prvky posílaly různé hodnoty, protože je potřeba sledovat změny na jednotlivých zařízeních. Z tohoto důvodu bylo zvoleno generování hodnot z určitého uživatelem zvoleného rozsahu, který by odpovídal hodnotám, který by daný reálný koncový prvek mohl naměřit.

Protože jsou mezi odesíláním zpráv několikasekundové pauzy, bylo zvoleno pouze generování pseudonáhodných čísel pomocí funkce `qrand()` z knihoven Qt a konstanty `RAND_MAX`. Generování probíhá pomocí metod `getRanVal()` a `getRanBool()` abstraktní třídy `TType`, která je více rozebrána v sekci 4.4.

Kapitola 4

Objektový návrh aplikace

Celá aplikace byla navržena objektově – tato kapitola popisuje hlavní třídy objektů a jejich rozhraní. Popis objektů grafického rozhraní a objekty pro síťovou komunikaci jsou rozebírány v dalších kapitolách. Také se zabývá jejich funkcí v programu. Je zde uveden jejich popis a nejvýznamnější metody a třídní proměnné. Dále jsou také uvedeny třídy, které od nich dědí.

V příloze A je schéma návrhu tříd, v němž jsou obsaženy třídy z této kapitoly – představují logickou strukturu programu.

4.1 Třída domácnosti

Třída `home` obsahuje seznam všech objektů adaptéru v rámci domácnosti, jejíž koncové prvky aplikace emuluje. Tento objekt představuje a zaobaluje spodní vrstvu systému, neboli zařízení, které se vyskytují přímo v domácnosti. Byl vytvořen z důvodu správy připojení k adaptérům, které se v domácnosti nachází. Existence tohoto objektu by také snadno umožňovala implementaci uložení nastavení či možnost spuštění dvou instancí aplikace.

Důležité metody a třídní proměnné třídy home

- **std::list⟨adapter*⟩ adapterList**
  Seznam adaptérů domácnosti v seznamu ze standardní knihovny jazyka C++.
- **void addAdapter(adapter* newAdapter)**
  Tato metoda přidá adaptér, který je předán v argumentu, do seznamu adaptérů patřící do domácnosti.
- **void deleteAdapter(adapter* adapterToDelete)**
  Tato metoda odstraní adaptér a s ním všechny koncové prvky, které jsou na něj napojené.

4.2 Třída adaptéru

Třída `adapter` zapouzdřuje údaje, které jsou třeba k připojení se k adaptéru, nejde o jeho emulaci. Jde o IPv4 adresu a doménové jméno. Připojení pak probíhá pomocí toho údaje, které je uživatelem zvoleno. Obsahuje také seznam abstraktních koncových prvků, které jsou na daný adaptér napojeny. Třída také spravuje síťovou komunikaci s daným adaptérem, což je popsáno v sekci 5.2.

Důležité metody a třídní proměnné třídy adapter

- **QHostAddress adapterAddress**
  Třídní proměnná, ve které je uložena IPv4 adresa adaptéru.
- **QString domainName**
  Třídní proměnná, ve které je uloženo doménové jméno adaptéru.
- **std::list⟨Element*⟩ sensorList**
  Seznam koncových prvků napojených na adaptér, uložené v seznamu ze standardní knihovny jazyka C++
- **void deleteSensor(Element* sensorToDelete)**
  Tato metoda vypojí koncový prvek ze sítě a odstraní ho ze seznamu.
- **void addSensor(Element* newSensor)**
  Metoda, která přidá a inicializuje do adaptéru nový koncový prvek, který je v argumentu.

4.3 Abstraktní třída koncového prvku

Třída `Element` představuje abstraktní koncový prvek daného typu s daným seznamem veličin, které měří či přepíná. Definuje rozhraní pro třídy, které od ní dědí, a také jejich společné metody. Jelikož jde o abstraktní třídu, neexistuje tedy žádná její instance - díky jejím vlastnostem jsou ale definovány ve vyšších vrstvách funkce s touto třídou v argumentu, do kterých vstupují všechny třídy, které od ní dědí. Některé třídní metody jsou definované jako prázdné, ale ve třídách, které od třídy `Element` dědí, jsou redefinovány a poté je využito polymorfní veřejné dědičnosti a u objektů je použita buď metoda prázdná, nebo je vykonán příslušný kód. [9]

Důležité metody a třídní proměnné třídy Element

- **std::vector⟨TType *⟩ valuesList**
  Seznam veličin a spínačů, které koncový prvek spravuje.

Třída senzoru

Třída s názvem `Sensor` dědí od třídy `Element` a představuje koncový prvek, který měří a přenáší $0 - n$ senzorů. Obsahuje také vnitřní časovač, ve kterém je uložen čas dalšího buzení - doba, za kterou se prvek probudí a odešle hodnoty na adaptér.

Důležité metody a třídní proměnné třídy Sensor

- **int interval**
  Interval, ve kterém se zařízení připojuje k adaptéru.
- **QTimer* activate**
  Časovač, který po uplynutí intervalu vyšle signál k připojení.
- **void activateConnection()**
  Redefinovaná metoda třídy `Element`, která přijímá signál od časovače na připojení adaptéru.

- **void receiveActivationTime(int)**
  Redefinovaná metoda rodičovské třídy, která přepíše uložený interval buzení nově přijatým.

Třída aktoru

Třída `Actor` dědící od abstraktního koncového prvku `Element` je koncový prvek emulující přepínání $1 - n$ aktorů, které jsou možné přepínat pomocí serveru. Přijímá tedy hodnoty aktorů, které v seznamu veličin přepisuje - musí být tedy neustále k dispozici.

Důležité metody a třídní proměnné třídy Aktor

- **void receiveTypeList(QByteArray)**
  Redefinovaná metoda třídy `Element` přijímající nové hodnoty spínačů od adaptéru.

Třída senzoru a aktoru

`SensorActor` je třída koncového prvku, který měří $0 - n$ senzorů a má $1 - n$ aktorů. Kvůli pravidelnému přenášení dat na adaptér je mezi jeho třídními proměnnými také časovač. Na rozdíl od třídy `Sensor` je ale k síti připojen neustále, což je ale řešeno mimo tuto třídu.

Důležité metody a třídní proměnné třídy SensorAktor

- **void receiveTypeList(QByteArray)**
  Tato metoda, redefinovaná ze třídy `Element`, přijímá ze síťové části data a přepisuje hodnoty spínačů v seznamu všech hodnot zařízení.

- **void activateConnection()**
  Metoda, která upozorní objekt, že je třeba odeslat data na server.

- **void receiveActivationTime(int)**
  Tato metoda přijímá a přepisuje interval odesílání dat na server.

4.4 Abstraktní třída přenášených typů

Třída `TType` je abstraktní třídou, která abstrahuje přenášené veličiny a spínače. Definuje rozhraní pro oba z těchto typů, čímž umožňuje jejich abstrakci pro objekty, jejichž jsou součástí. Všechny objekty, které od této třídy dědí slouží k uchování hodnoty daného zařízení – aktuální naměřené hodnoty nebo v druhém případě hodnoty aktoru přijatou od serveru.

Tato třída obsahuje rozhraní jak pro jednotlivé senzory, tak pro jednotlivé aktory. Jsou to metody pro generování náhodné hodnoty z rozsahu, jak je popsáno v sekci 3.3, a metoda pro uložení nové hodnoty ze serveru pro aktory. Pro potřeby sestavení zprávy posílané po síti také vrací svůj typ podle protokolu, tak je popsáno v sekci 5.1.

Třídy, které od ní dědí, můžeme vidět v tabulce 4.1. Kromě názvu třídy je zde uveden i název zobrazovaný v grafickém rozhraní, jednotky pro danou veličinu – mohla by nastat situace, kdy stejnou veličinu měříme v jiných jednotkách. Také rozsah hodnot a jde-li o senzor či aktor.

Díky abstraktní třídě s rozhraním pro různé typy by nebylo přidávání typů problém. Z hlediska implementace by to znamenalo pouze přidání třídy a její inicializace při přidávání veličin. Další změny např. v kódu síťové komunikace či metod adaptéru by nebyly třeba.

Název třídy	**Název**	**Jednotky**	**Rozsah hodnot**	**Typ**
Tempterature	Teplota	°C	minimum - maximum	Senzor
Humidity	Vlhkost	%r	minimum - maximum	Senzor
Presure	Tlak	hPa	minimum - maximum	Senzor
SwitchSensor	Sepnutí	-	true / false	Senzor
SwitchActor	Sepnutí	-	true / false	Aktor
LightIntensity	Intenzita světla	lx	minimum - maximum	Senzor
SoundIntensity	Intenzita hluku	dB	minimum - maximum	Senzor
Emision	Emise (CO_2)	ppm	minimum - maximum	Senzor

Tabulka 4.1: Třídy dědící od abstraktní třídy přenášeného typu.

Důležité metody a třídní proměnné třídy TType

- **float minimum**
  Třídní proměnná, ve které je uloženo minimum pro generování hodnot.
- **float maximum**
  Třídní proměnná určující maximum rozsahu generování hodnot.
- **bool state**
  Proměnná, ve které je uložen momentální stav spínače.
- **float getRanVal()**
  Metoda pro generování náhodných hodnot z rozsahu pro veličiny.
- **quint8 getRanBool()**
  Metoda pro generování náhodné hodnoty pro spínač - senzor.
- **void update(bool newState)**
  Metoda sloužící k přepsání hodnoty spínače - aktoru.

Kapitola 5

Síťová komunikace

Tato kapitola se zabývá síťovou částí aplikace – je zde představen komunikační protokol a poté samotné řešení a implementace komunikace v aplikaci. Je zvoleno řešení pomocí Qt, neboť nabízí vhodné knihovny pro síťové objekty a také datové typy s velikostí nezávislou na platformě, což je důležité pro dodržení protokolu.

5.1 Komunikační protokol emulátoru

Dále je popsán komunikační protokol [8] používaný ke komunikaci mezi koncovými prvky a adaptérem. Slouží k inicializaci párování zařízení a výměně dat mezi nimi.

Formát zprávy odesílané na adaptér

Zpráva odesílaná na adaptér z koncového prvku se skládá z posloupnosti bitů různého významu, jak je uvedeno v tabulce 5.1. Dále jsou definovány přenášené typy pro jednotlivé jednotky, které jsou jednoznačně identifikované – ty odpovídají třídám z tabulky 4.1.

Bity	Význam	Použitý datový typ	Velikost v Bytech
0 – 7	Inicializační hodnota	unsigned char	1
8 – 23	Verze protokolu	unsigned short	2
24 – 39	Stav baterie	unsigned short	2
40 – 55	Kvalita signálu	unsigned short	2
56 – 63	Počet přenášených veličin	unsigned char	1
64 – 71	Rezervované	pad byte	1
72 – XX	Seznam dvojic typ:hodnota		

Tabulka 5.1: Zpráva zasílaná z koncových prvků na adaptér.

Inicializační hodnota nabývá hodnoty 0xFF, jde-li o první připojení daného koncového prvku do sítě. Při opětovném posílání hodnot je již inicializační hodnota rovna 0x00. Velikost této hodnoty ale nabízí o budoucna přidání dalších stavů koncových prvků. Druhou hodnotou zprávy je verze protokolu. Tato aplikace používá protokol v první verzi, je odesílána tedy hodnota 1. Stav baterie a kvalita signálu nabývají hodnoty 0 – 100, čímž je myšlena hodnota této veličiny v procentech. Počet přenášených veličin určuje délku zprávy – respektive délku seznamu dvojic typ:hodnota a nejčastěji nabývá hodnoty 1.

Seznam na konci zprávy, délky jaké je uvedeno ve zprávě předtím, se skládá z posloupnosti typ a hodnota, kdy typ je bytová hodnota, odpovídající veličinám podle tříd ze sekce 4.4. V níže uvedené tabulce 5.2 vidíme hodnotu posílanou protokolem pro každou veličinu. Po této informaci o přenášeném typu následuje momentální hodnota. U některých veličin má hodnotu čtyřbytovou – jde o typ float, u jiných jeden byte – typ bool, jak je také uvedeno v tabulce. V současné verzi protokolu byl datový typ *float* nahrazen datovým typem *long*. Tato změna zatím není implementována, protože serverová aplikace, která je testována, zatím používá starší verzí protokolu.

Název	Jednotka	Označení typu	Datový typ hodnoty	Velikost v Bytech
Teplota	°C	0x00	float	4
Vlhkost	%r	0x01	float	4
Tlak	hPa	0x02	float	4
Spínač (senzor)	-	0x03	bool	1
Spínač (aktor)	-	0x04	bool	1
Intenzita světla	lx	0x05	float	4
Intenzita hluku	dB	0x06	float	4
Emise (CO_2)	ppm	0x07	float	4

Tabulka 5.2: Rozdělení typ:hodnota.

Zprávy zasílané na senzory

Server je schopný rozeznat, o jaký typ prvku se jedná podle délky seznamu dvojic a stavové informace, a podle toho zasílá zařízení odpověď, která je jen přeposlána adaptérem na zařízení. Zpráva zasílaná adaptérem na senzor je v tabulce 5.3, zpráva zasílaná na aktory je v tabulce 5.1 a zpráva zasílaná na sensor/aktory pak v tabulce 5.1.

Bity	Význam	Velikost v Bytech
0 – 15	Verze komunikačního protokolu	2
16 – 79	ID senzor	8
80 – 95	Čas dalšího buzení	2

Tabulka 5.3: Formát zprávy posílané na senzor.

Bity	Význam	Velikost v Bytech
0 – 15	Verze komunikačního protokolu	2
16 – 79	ID senzor	8
80 – 95	Čas dalšího buzení	2
96 – XX	Seznam dvojic typ:hodnota	

Tabulka 5.4: Formát zprávy posílané na aktor.

Bity	Význam	Velikost v Bytech
0 – 15	Verze komunikačního protokolu	2
16 – 79	ID senzor	8
80 – 95	Čas dalšího buzení	2
96 – XX	Seznam dvojic typ:hodnota	

Tabulka 5.5: Formát zprávy posílané na senzor/aktor.

Seznam dvojic na konci zprávy odpovídá popisu z tabulky 5.2 z minulé sekce.

Komunikace

Tato sekce popisuje výměnu zpráv mezi komunikujícími objekty, jejich pořadí a také v jakém případě jde o komunikaci synchronní nebo asynchronní. Při procesu párování jde o jednoduchou výměnu zpráv, kdy odpověď je pro každý typ prvku jiná. Tento mechanismus probíhá při každém novém připojení prvku do sítě, nebo také v případě, kdy se koncovému zařízení vybila baterie nebo ztratilo signál. Tyto situace ale aplikace emulátoru nesimuluje.

Dále se výměna zpráv u jednotlivých koncových prvků liší. U koncového prvku typu senzor probíhá komunikace synchronně, kdy při každém buzení senzor odešle zprávu a po přijetí odpovědi se opět uspí.

Aktor po inicializaci další zprávy neodesílá – pouze přijímá zprávy od adaptéru s novými hodnotami. Čas buzení ve zprávě je roven 0, což značí, že se prvek neuspává. Zprávy z adaptéru totiž přicházejí asynchronně.

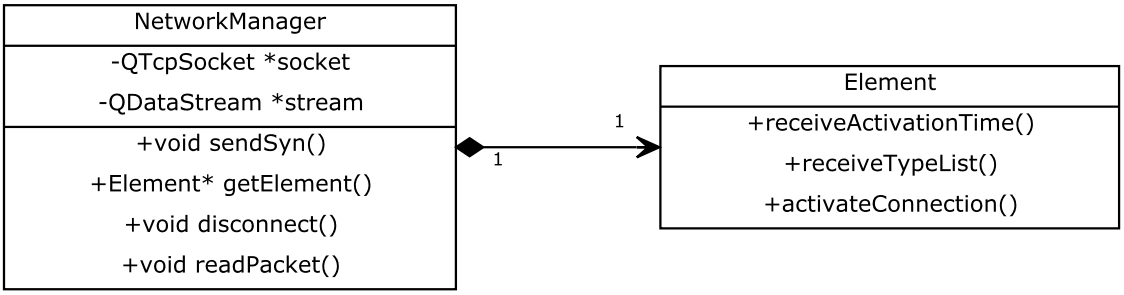
Typ koncového zařízení sensor/aktor se neuspává, protože očekává asynchronní zprávy od adaptéru s novým nastavením pro aktory, ale zároveň v daném časovém intervalu odesílá hodnoty veličin, které měří. Je-li čas buzení roven 0 chápe zařízení zprávu jako nové nastavení pro aktory, v opačném případě jde o odpověď pro sensory a aktualizuje se interval odesílání dat.

5.2 Implementace komunikace jednotlivých koncových zařízení

Tato sekce řeší návrh síťové části aplikace a implementaci síťového protokolu ze sekce 5.1.

Síťová komunikace je jednak řešena v objektech třídy `NetworkManager`, kde je řešeno připojování k síti přes soket, její odpojení, kódování zprávy do zprávy protokolu a také její dekódování. Tato třída zároveň dědí od třídy z knihovny Qt, `QThread`, která reprezentuje vlákno. Tato třída je provázána s jedním koncovým prvkem podle obrázku 5.1, od kterého získává o daném prvku data. Každé zařízení tedy běží ve svém vláknu.

Mezi hlavními funkcemi třídy `NetworkManager` je kódování zprávy do bitového protokolu, takže je třeba zachovat správné velikosti jednotlivých proměnných posílaných po síti. Podle standardu *RFC* jsou po síti bity přenášeny v kódování Big-Endian [10], což bylo třeba zajistit. To je v programu řešeno pomocí třídy `QDataStream` z knihovny Qt, která umožňuje datový proud mezi souborem, v tomto případě soketem, který zde představuje výstup, a proměnnou v kódu, která je vstupem. Pomocí operátoru `<<` jsou data serializována a posílána přes soket po síti právě v požadovaném kódování, které je pro tuto třídu výchozí [4]. Jelikož se velikosti jednotlivých datových typu uvedených v protokolů mohou lišit na jiných platformách, byly zvoleny datové typy s přesně určenou velikostí a to pro `unsigned`



Obrázek 5.1: Třída síťových objektů `NetworkManager` a její vztah s třídou `Element`.

`char` typ `quint8`, který reprezentuje osmibitový int bez znaménka, a pro `unsigned short` typ `quint16` reprezentující dvoubytový int bez znaménka. Oba použité typy jsou z knihovny Qt. Jelikož je ale po síti posílán i typ `float`, bylo třeba ve třídě `QDataStream` nastavit i její přesnost na 32-bitů podle protokolu. Další částí síťové třídy je pak dekódování příchozích dat, které je řešeno obdobně jako kódování, opět pomocí třídy `QDataStream`.

Síťová komunikace je pak ovládána funkcemi ve třídě `adapter`, které žádají o připojení a dostávají od vláken senzorů signál o odpojení a poté od časovače signál, že uplynula doba, kdy senzor spí. Poté se volá metoda na odeslání dat.

Důležité metody a třídní proměnné třídy NetworkManager

- **QTcpSocket *socket**
  Třídní proměnná soketu, přes který probíhá komunikace.
- **QDataStream *stream**
  Třídní proměnná datového toku, přes který je otevřen soket.
- **void readPacket(QByteArray)**
  Metoda, ve které se dekóduje zpráva přijatá přes soket.
- **void writePacket()**
  Metoda, ve které se zpráva kóduje a posílá přes soket.
- **void sendSyn(QString domain, QHostAddress address, bool useIP,quint8 initValue = 0)**
  Metoda, ve které zařízení zkontroluje, zda je stále připojeno a poté vyšle signál k poslání dat.

Důležité metody a třídní proměnné třídy adapter

- **void sensorWakeUp(Element*)**
  Metoda, která najde odpovídající vlákno k danému koncovému prvku a vyžádá opětovné připojení.

Kapitola 6

Grafické rozhraní aplikace

Aplikace, kterou tato práce popisuje, má grafické rozhraní, které umožňuje jednodušší a intuitivnější testování než nabízely konzolové skripty. Jelikož je aplikace implementována v jazyce C++, byl zvolen framework Qt ve verzi 4.8. Tento framework umožňuje vytváření multiplatformních aplikací a existuje k němu rozsáhlá dokumentace. Vývoj aplikace probíhal na Linuxu v prostředí aplikace Qt Creator, který je popsán v sekci 6.1.

Tato kapitola se zabývá popisem prostředí Qt Creator, ve kterém byla aplikace vyvíjena, samotného návrhu grafického rozhraní, jeho implementací a také uvádí příklady užití.

6.1 Nástroj Qt Creator

Qt Creator [5] je vývojové prostředí pro vývoj Qt aplikací. Je součástí Qt Projektu. Obsahuje editor kódu, debugger a návrhové prostředí pro grafická rozhraní, Qt Designer. Je dostupný pod licencemi GPL a LGPL nebo komerční licencí [6].

Při vytváření grafické třídy generuje Qt Creator kromě hlavičkových souborů a souborů obsahujících implementaci ještě soubory s příponou `.ui`, které definují obsah grafického formuláře pomocí XML. Tyto soubory se automaticky generují při práci v Qt Designeru, kde se vzhled formulářů vytváří jednoduše pomocí grafického rozhraní. Další výhodou je, že se o tyto grafické prvky definované v XML souboru nemusíme starat, protože se samy inicializují a v destrutoru se pouze smaže objekt reprezentující celý formulář. Jsou dostupné v celém objektu dané třídy právě pomocí objektu `ui`.

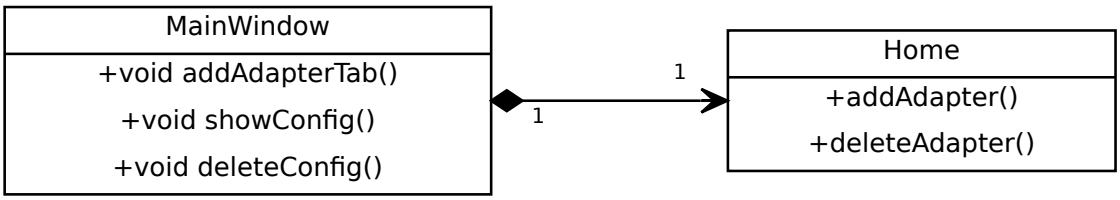
6.2 Návrh a implementace grafického rozhraní

Hlavním cílem návrhu grafického rozhraní byla jednoduchost a tedy přehlednost aplikace. Také bylo navrhnuto tak, aby byly jeho objekty zcela odděleny od hlavních tříd, ve kterých probíhá veškerá logika programu. V objektech grafického rozhraní jsou tedy výhradně metody a proměnné, které se starají o zobrazování logiky probíhající v hlavních třídách aplikace popsaných v kapitole 4.

Jsou to třídy dědící od grafických objektů z knihovny QT, upravené pro potřeby této aplikace. Hlavní a základní třídou je třída `MainWindow`, která je základní třídou generovanou při založení nového projektu v Qt Creatoru. Je to třída představující hlavní okno aplikace, ve které se v záložkách objevují jednotlivé adaptéry a v nich, ve třídě `tabInside`, seznam koncových zařízení připojených na adaptér. Koncové prvky pak zobrazuje třída `sensorWidget`

a veličiny a přepínače v něm třída `sensorWidgetExtension`, která lze ve výpisu koncových zařízení skrýt. Dalšími grafickými třídami jsou poté `adapterConfig` a `sensorConfig` představující grafické rozhraní pro konfiguraci adaptéru a koncového prvku.

Komunikace tlačítek s logikou aplikace je řešena pomocí signálů a slotů – kdy grafická třída při jakékoliv akci vyvolané uživatelem posílá signál, který je propojen se slotem – což je třídní metoda, který se volá v hlavních objektech aplikace. Naopak při změně v programu vysílá signál objekt některé z hlavních tříd a je odchycen slotem v objektu grafického prostředí.

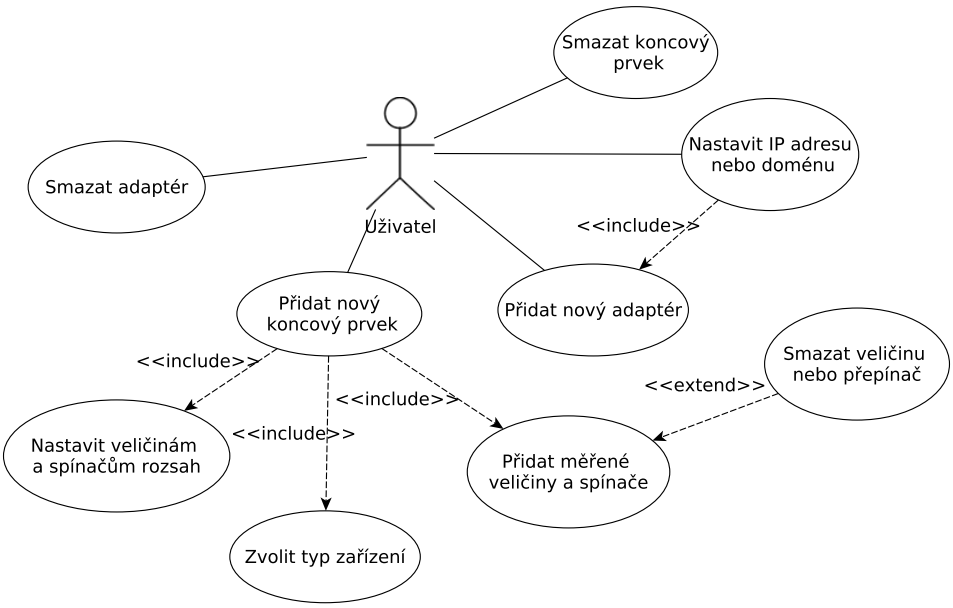


Obrázek 6.1: Vztah třídy `MainWindow` a `home`.

Grafické objekty mají strukturu podobnou struktuře hlavních objektů z přílohy A, kdy hlavní grafické třídy `MainWindow`, `tabInside`, `sensorWidget` obsahují v třídních proměnných ukazatel na objekt hlavní třídy. Pro třídu `MainWindow` a objekt třídy `home` je schéma na obrázku 6.1, pro ostatní třídy je propojení analogické. Pro třídu `tabInside` objekt třídy `adapter` a pro třídu `sensorWidget` objekt třídy `Element`.

6.3 Popis ovládání

Tato sekce rozebírá použití a ovládání aplikace. Zabývá se jednotlivými prvky grafického prostředí a popisuje je. Jsou zde také zmíněny a více rozepsány jednotlivé případy užití podle diagramu na obrázku 6.2.



Obrázek 6.2: Use-Case diagram aplikace.

Akce: Přidat nový adaptér

Přidání nového adaptéru je započato kliknutím na tlačítko **Pridej Adapter**, které je vidět nahoře v základním okně aplikace na obrázku 6.3. Po kliknutí se objeví konfigurační okno adaptéru z obrázku 6.4, kde můžeme adaptér pojmenovat pro účely rozlišování a hlavně vybrat, jestli se bude připojovat pomocí doménového jména či IPv4 adresy. Toto odpovídá akci *Nastavit doménové jméno nebo IP adresu adaptéru* popsané v podsekci 6.3.

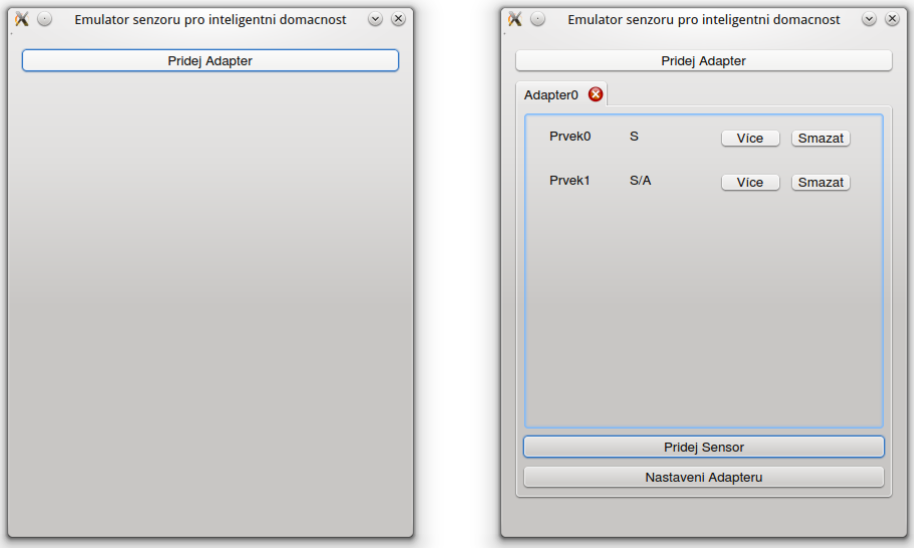
Akce: Smazat adaptér

Tato akce se provádí stisknutím tlačítka na odstranění adaptéru umístěného na jeho záložce, které můžeme vidět na obrázku 6.3 vpravo.

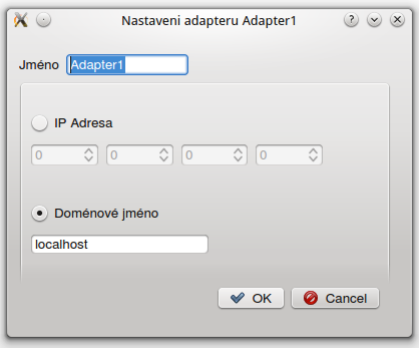
Akce: Přidat nový koncový prvek

Tato akce začíná kliknutím na tlačítko **Pridej Sensor**, které je v pravé části obrázku 6.3 – tedy v okně, kde je již přidaný adaptér. Nový koncový prvek musíme totiž nějakému adaptéru přiřadit, čímž dosáhneme jeho přidáním na záložce adaptéru, ke kterému patří. Po kliknutí se objeví konfigurační okno koncového prvku – viz obrázek 6.5, ve kterém vykonáme akce *Zvolit typ zařízení*, *Přidat měřené veličiny a spínače* a *Nastavit veličinám a spínačům rozsah* podle diagramu 6.2. Typ zařízení zvolíme v záložce **Obecne**, kde lze nastavit koncovému prvku i specifické jméno a v případě prvku typu senzor nebo senzor/aktor i interval odesílání zpráv. V záložce **Veliciny** lze pak podle zvoleného typu zařízení přidat jednotlivé veličiny nebo spínače, které zařízení měří či přepíná. U každé z nich lze pak zvolit rozsah, ze kterého se budou generovat hodnoty odesílané zprávami. U spínačů pak výchozí pozice přepínače.

Chceme-li při vytváření některou z veličin smazat – což odpovídá akci *Smazat veličinu nebo přepínač*, v rozbalovacím menu zvolíme možnost **Smazat**.



Obrázek 6.3: Vzhled okna aplikace před a po přidání adaptéru.



Obrázek 6.4: Konfigurační okno adaptéru.

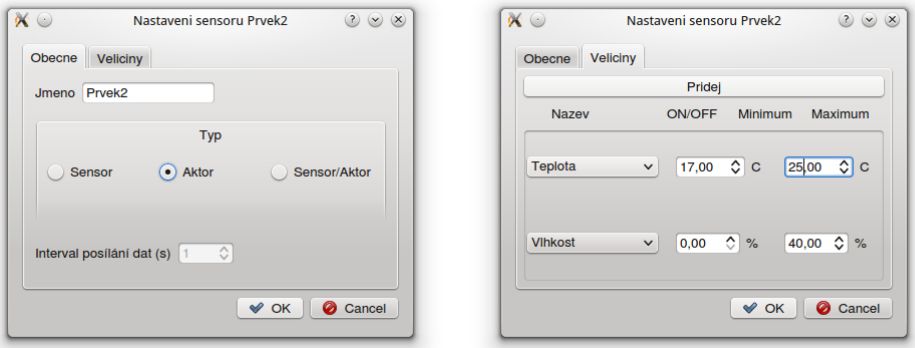
Koncový prvek je poté přidán do seznamu k adaptéru, ke kterému se připojuje. Je označen písmenem podle typu zařízení a je možné zobrazit, které veličiny přenáší.

Akce: Smazat koncový prvek

Akce se provádí stisknutím tlačítka **Smazat** umístěného vedle každého koncového prvku. Stisknutí provede smazaní prvku, vedle kterého se tlačítko nachází – tyto tlačítka můžeme vidět v pravém okně na obrázku 6.3.

Akce: Nastavit doménové jméno nebo IP adresu adaptéru

Tato akce se samostatně spustí tlačítkem **Nastaveni Adapteru** v záložce daného adaptéru v hlavním okně aplikace – viz obrázek 6.3. Poté se objeví konfigurační okno adaptéru z obrázku 6.4, kde můžeme přenastavit jeho jméno nebo IP adresu či doménové jméno.



Obrázek 6.5: Konfigurační okno koncového prvku.

Kapitola 7

Testování

Tato kapitola se zabývá testováním aplikace v různých prostředích a na různých platformách. Tyto prostředí zde budou nejdříve představena se svými parametry, poté budou představeny nastavení aplikace, které byly pro testování použity. A nakonec budou zhodnoceny a porovnány výsledky.

K testování byl použit skript emulující adaptér `ademu.py` v jazyce Python od Bc. Martina Douděry – viz podsekce 7.1. Jako emulátor serveru pro testování byl pak použit skript `server.py` také v jazyce Python od Bc. Petera Boby. Tyto skripty jsou dostupné v repozitáři výzkumné skupiny.

Na závěr byla testována i serverová aplikace od Matúše Blaha – ve verzi dostupné na serveru výzkumné skupiny `ant-2.fit.vutbr.cz`, což je popsáno v sekci 7.2.

7.1 Parametry testovacích prostředí

V této sekci budou popsány jednotlivá prostředí, ve kterých testování probíhalo a také skripty a programy, pomocí kterých se aplikace testovala.

Platformy použité k testování emulátoru

Aplikace byla testována na platformách s dvěma různými operačními systémy, což otestovalo její multiplatformnost. První testování probíhalo na stroji s 64bitovou verzí operačního systému *Windows 7*. Druhé testování pak proběhlo na stroji se systémem *Linux Mint 16* – 32bit, s grafickým prostředím KDE.

Platformy použité pro běh serveru

Skript emulující server v jazyce Python `server.py` zmíněný v úvodu této kapitoly byl spuštěn na stejném stroji jako testovaná aplikace – *Linux Mint 16* – 32bit, šlo tedy o testování v lokálním prostředí. Aplikace serveru dostupná na `ant-2.fit.vutbr.cz` byla spouštěna tamtéž. Jde o stroj s 64bitovým linuxovým systémem *CentOS*.

Skript ademu.py

Tento skript emulující adaptér byl použit jako mezivrstva mezi koncovými prvky a serverem. Jde o jednoduchý program, který se spouští s dvěma parametry – IPv4/IPv6 adresou nebo doménovým jménem serveru, na kterém naslouchá serverová část systému. Druhý parametr

je čtyřmístné sériové číslo adaptéru. Program souží k přeposílání hodnot z koncových prvků do vyšší vrstvy.

Tento skript byl vždy spouštěn na stroji se 32bitovým systémem *Linux Mint 16*. V případě testování na systému *Windows 7* se k němu emulátor připojil přes lokální síť.

Skript server.py

Skript `server.py` je emulátor serveru. Prostřednictvím adaptéru přijímá informace od koncových zařízení a emuluje změny vykonané uživatelem – ovládání spínačů a změny intervalů zasílání pro senzory. Spouští se bez parametrů. Tato aplikace sloužila primárně k testování funkčnosti emulátoru.

Serverová aplikace

Tato aplikace společně s databází představuje jednu z vrstev systému, zprostředkovává komunikaci mezi koncovými prvky, respektive adaptérem, a uživatelem. Spouští se společně s databází pomocí skriptu v jazyce Bash `run_server.sh`.

7.2 Průběh testování

Tato sekce popisuje testování, které sloužilo k vývoji samotné aplikace, ale také se zabývá testy nad aktuální verzí serverové části systému, která se nachází na dedikovaném serveru `ant-2.fit.vutbr.cz`.

Před každou emulací, která se prováděla, se restartovaly všechny prostředky, které v testu figurovaly, aby se zabrání ovlivnění výsledků jinými faktory. Jelikož jde o síťovou komunikaci, tak se při testování nesmělo zapomenout na to, že této výměně dat mohou bránit bezpečnostní opatření jako například *firewall*. Takže se muselo zajistit, aby mohly obě strany data bez problému přijímat i odesílat.

Testování funkčnosti aplikace

Testování samotné aplikace probíhalo po celou dobu jejího vývoje. Mimo testování v oblasti konektivity a samotného odeslání a čtení zprávy, bylo nutné testovat, jsou-li zprávy posílané po síti platformě nezávislé, ve správném kódování a jestli mají jednotlivé její části – definované podle protokolu v sekci 5.1, správnou velikost.

Velikost v Bytech	Význam odeslané hodnoty	Odeslaná hodnota	Odpovídající část zprávy
1	Inicializační hodnota	0xFF	ff
2	Verze protokolu	1	0001
2	Stav baterie	100	0064
2	Kvalita signálu	100	0064
1	Počet přenášených veličin	1	01
1	Rezervované	0	00
1	Typ - Teplota	0x00	00
4	Hodnota	$6,0$	41500000

Tabulka 7.1: Dekódování zprávy z výstupu.

Jako příklad testování je uvedena tabulka 7.1, ve které je vidět dekódování zprávy. Jde o výstup, získaný ze skriptu `ademu.py` funkcí `print` a použitím metody `encode('hex')` nad binárními daty. Tabulka ukazuje, jak lze danou zprávu číst a vyhodnotit a zda je dodržen protokol. V nejpravějším sloupci lze vidět přímo části výstupu, zatímco v nejlevějším předpokládanou velikost v bytech podle protokolu. Podle toho lze určit sloupec *Odeslaná hodnota* a poté rozhodnout, jestli jeho hodnota spadá do rozsahu hodnot odpovídající části zprávy podle sloupce *Význam odeslané hodnoty*.

Jak již bylo dříve zmíněno, v této části testování byl využit emulátor serveru a emulátor adaptéru.

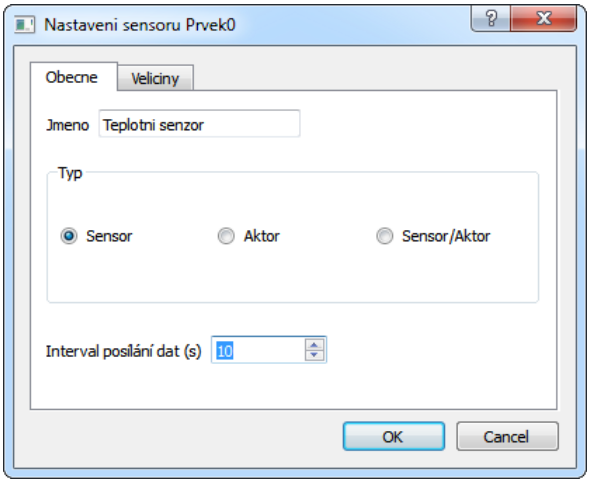
Testování serverové aplikace

Tato podsekce se zabývá testováním serverové části systému. Aplikace byla spuštěna na dedikovaném serveru `ant-2.fit.vutbr.cz` a emulátor byl spuštěn postupně na stroji s 64bitovým systémem *Windows 7* a poté na stroji s 32bitovou distribucí Linuxu. Komunikace probíhala přes mezivrstvu adaptéru – reprezentovaném emulátorem adaptéru, spuštěném na stroji s Linuxem.

Obrázek 7.1: Konfigurace emulátoru.

Zvolená metoda testování spočívala v postupném přidávání senzorů s jednou přenášenou veličinou, viz obrázek 7.1. Z tohoto obrázku lze také vyčíst, že k adaptéru jsou připojeny dva senzory, z nichž jeden měří teplotu a druhý tlak. V aplikaci je viditelná i momentální vygenerovaná hodnota. Interval odesílání zpráv byl 10 s, podle okna nastavení koncového prvku na obrázku 7.2.

Jak lze vidět z výstupů serverové aplikace v příloze B, které odpovídají tomuto nastavení emulátoru – kdy první z výstupů je pro senzor měřící tlak a druhý pro senzor měřící



Obrázek 7.2: Konfigurace koncového prvku.

teplotu, je server schopen rozeznat jednotlivé proměnné a přenášenou hodnotu senzoru. Jediná neobvyklá věc v tomto výpisu jsou řádky `Length` a `Recived datasize`, které by se měly shodovat. Test byl proto opakován na druhé platformě, ale výstupy ze serverové aplikace byly obdobné. Proto se přistoupilo k druhému způsobu testování – emulace prvků s více než jednou veličinou, což způsobilo u obou prvků stejnou chybu serveru.

Po diskuzi s autorem programu se došlo k závěru, že serverová aplikace neimplementuje v protokolu rezervované místo o velikosti jednoho byte – tzv. *pad byte*[1].

[1] podle protokolu definovaného v sekci 5.1.

Kapitola 8

Závěr

Tato bakalářská práce se zabývá systémy inteligentní domácnosti, které jsou v současné době stále rozšířenější. Popisuje možné výhody i nevýhodný této ideje. Srovnává již existující a dostupné systémy s tím, který je vyvíjený výzkumnou skupinou na FIT VUT.

Tématem a aplikací, kterou popisuje, je nástroj na testování serverové části systému. To probíhá emulováním koncových zařízení tvořících jeho nejnižší vrstvu. Proto bylo třeba tyto prvky nejprve vymezit a určit do jaké míry budou abstrahovány. Dalším krokem byl objektový návrh, který definuje hlavní třídy tvořící logickou strukturu programu.

Jelikož jde o aplikaci, která se účastní síťové komunikace, bylo velmi důležité dodržet protokol – včetně velikostí jednotlivých přenášených datových typů na různých platformách. Do budoucna by se dalo uvažovat o změně protokolu kvůli změně datového typu přenášeného po síti pro veličiny – jakmile tuto změnu implementuje i serverová aplikace.

Práce dále popisuje návrh a implementaci grafického uživatelského rozhraní a zmiňuje i jednotlivé případy užití aplikace. Menší počet prvků v aplikaci napomáhá k orientaci v prostředí a nabízí jednoduché testování.

Nakonec je popsáno, jak byla aplikace testována pomocí emulátoru serveru na různých platformách. Druhou částí bylo testování serverové aplikace, kvůli jejímuž vývoji aplikace vznikla. Má za úkol emulovat hardwarové prvky, které jsou vyvíjeny výzkumnou skupinou, tudíž aplikace umožňuje vlastní nastavení veličin, které aplikace měří a spínačů, které kontroluje. Z důvodu, že byla aplikace testována při vývoji pouze na emulátoru serveru, jsou ovladatelné prvky – aktory, implementovány jen z části. Jak se při testování serverové aplikace ovšem ukázalo, je emulátor schopen objevit nesrovnalosti v komunikačním protokolu.

Hlavní přínos této práce vidím nejen ve výsledné aplikaci, do které by bylo třeba pro reálné nasazení přidat více funkcí, ale také v možnosti spolupráce s výzkumnou skupinou a hlubšímu porozumění použitých technologií.

Literatura

[1] About. *Xbmc.org* [online]. [cit. 2014-05-20]. Dostupné z: http://xbmc.org/about/.

[2] CANNYNET S.R.O. *CannyNet* [online]. 2011-2014 [cit. 2014-02-04]. Dostupné z: http://www.cannynet.com/.

[3] Domotica. *Smart-homes.nl* [online]. [cit. 2014-05-15]. Dostupné z: http://www.smart-homes.nl/Domotica.aspx.

[4] QDataStream Class Reference. *Qt Project* [online]. [cit. 2014-05-18]. Dostupné z:http://qt-project.org/doc/qt-4.8/qdatastream.html#ByteOrder-enum.

[5] Qt Creator. *Qt Project* [online]. 2014-01-22 [cit. 2014-05-18]. Dostupné z: http://qt-project.org/wiki/category:tools::qtcreator.

[6] Qt Licensing. *Qt Project* [online]. [cit. 2014-05-18]. Dostupné z: http://qt-project.org/products/licensing.

[7] KORČEK, P.: Smarthome architecture. *Internet of Things* [online]. 2013 [cit. 2014-02-04]. Dostupné z: https://merlin.fit.vutbr.cz/wiki-iot/index.php/Smarthome_architecture.

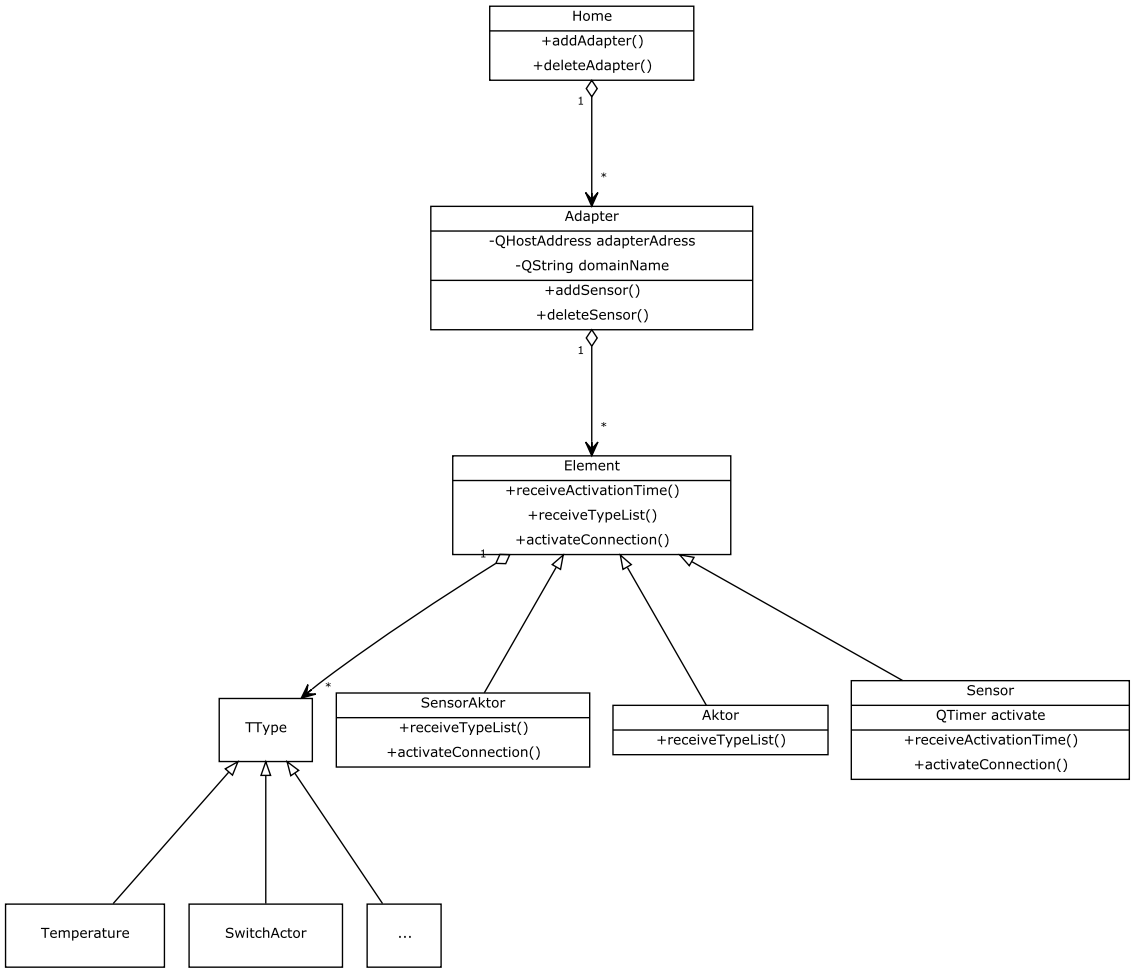
[8] KORČEK, P.: Smarthome sensors: Komunikačný protokol. *Internet of Things* [online]. 2013 [cit. 2014-02-04]. Dostupné z: https://merlin.fit.vutbr.cz/wiki-iot/index.php/Smarthome_sensors.

[9] PRATA, S.: *Mistrovství v C++*. Brno: Computer Press, a.s., 2007, ISBN 978-80-251-1749-1.

[10] Reynolds, J.; Postel, J.: RFC 1700. *Assigned Numbers* [online]. 1994 [cit. 2014-05-19]. Dostupné z: http://tools.ietf.org/html/rfc1700.

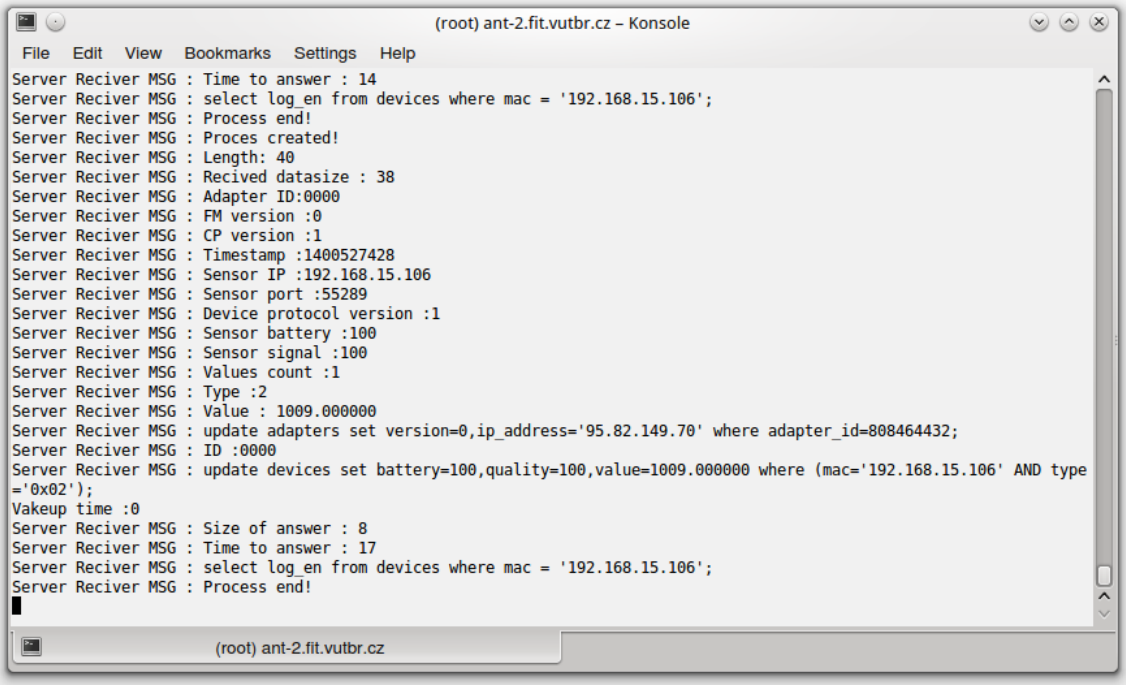
Příloha A

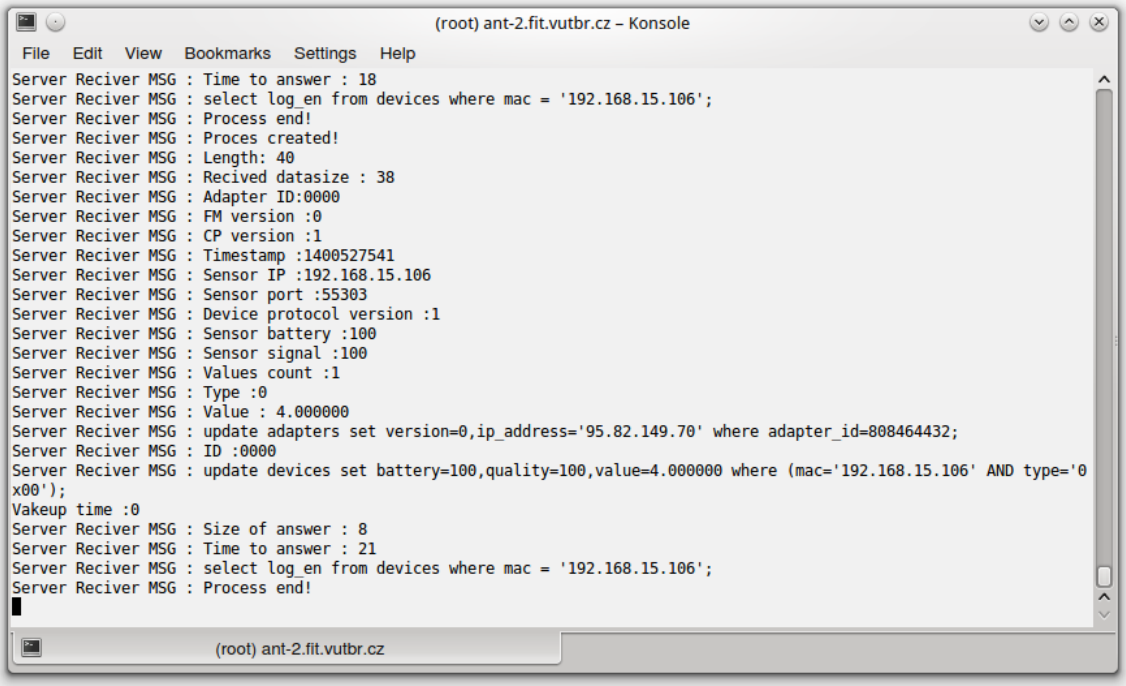
Diagram tříd



Příloha B

Výstup serverové aplikace




(root) ant-2.fit.vutbr.cz – Konsole
File Edit View Bookmarks Settings Help
Server Reciver MSG : Time to answer : 18
Server Reciver MSG : select log_en from devices where mac = '192.168.15.106';
Server Reciver MSG : Process end!
Server Reciver MSG : Proces created!
Server Reciver MSG : Length: 40
Server Reciver MSG : Recived datasize : 38
Server Reciver MSG : Adapter ID:0000
Server Reciver MSG : FM version :0
Server Reciver MSG : CP version :1
Server Reciver MSG : Timestamp :1400527541
Server Reciver MSG : Sensor IP :192.168.15.106
Server Reciver MSG : Sensor port :55303
Server Reciver MSG : Device protocol version :1
Server Reciver MSG : Sensor battery :100
Server Reciver MSG : Sensor signal :100
Server Reciver MSG : Values count :1
Server Reciver MSG : Type :0
Server Reciver MSG : Value : 4.000000
Server Reciver MSG : update adapters set version=0,ip_address='95.82.149.70' where adapter_id=808464432;
Server Reciver MSG : ID :0000
Server Reciver MSG : update devices set battery=100,quality=100,value=4.000000 where (mac='192.168.15.106' AND type='0
x00');
Vakeup time :0
Server Reciver MSG : Size of answer : 8
Server Reciver MSG : Time to answer : 21
Server Reciver MSG : select log_en from devices where mac = '192.168.15.106';
Server Reciver MSG : Process end!
(root) ant-2.fit.vutbr.cz