VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Jakub Michálek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SOFTWARE PRO OVLÁDÁNÍ A SPRÁVU INTELIGENTNÍCH PRVKŮ DOMÁCNOSTI
SOFTWARE FOR CONTROL AND MANAGEMENT OF HOME AUTOMATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jakub Michálek**
AUTHOR

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

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Jakub Michálek ***ID:*** 186140

Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Software pro ovládání a správu inteligentních prvků domácnosti

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit softwarové řešení pro ovládání a správu inteligentních zásuvek v domácnosti.
Daný program bude provozován na centrální stanici, která koncentruje naměřené hodnoty ze skupiny inteligentních zásuvek.
Vytvořený software bude umožňovat ukládání a přehlednou vizualizaci přijímaných dat s možností ovládání připojených zařízení podle stanoveného protokolu a použitých rozhraní.
Součástí práce je tvorba testovacího programu, který generuje testovací data dle použitého komunikačního protokolu a na určeném rozhraní, pro ověření funkčnosti vytvořeného řešení.
V práci bude diskutována problematika jednoho centrálního řídícího prvku, možnosti omezení následků výpadku centrálního řídícího prvku a případná implementace metod zajištění kybernetické bezpečnosti.

DOPORUČENÁ LITERATURA:

[1] HANES, David, Gonzalo SALGUEIRO, Patrick GROSSETETE, Robert BARTON a Jerome HENRY. IoT fundamentals: networking technologies, protocols, and use cases for the Internet of things. Indianapolis, Indiana, USA: Cisco Press, [2017]. Cisco Press fundamentals series. ISBN 1587144565.

[2] Smart grids and their communication systems. New York, NY: Springer Berlin Heidelberg, 2018. ISBN 9789811317675.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 18.8.2020

Vedoucí práce: Ing. et Ing. Petr Musil

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato diplomová práce popisuje vytvoření aplikace pro ovládání a správu inteligentních zásuvek v domácnosti. V první části je popsán software, který byl využit při tvorbě této aplikace. Konkrétně aplikační platforma Qt a relační databázový systém SQLite. V druhé části je popsána struktura a funkce této aplikace. Další část popisuje aplikaci sloužící k testování funkčnosti vytvořené aplikace a navržený komunikační protokol. Poslední část práce popisuje testování a zhodnocení vytvořené aplikace.

KLÍČOVÁ SLOVA

Qt, SQLite, Databáze, Aplikace, GUI, Inteligentní zásuvka

ABSTRACT

This master thesis describes the development of an application for control and management of smart sockets at home. The first part describes the software that was used in creating this application. Specifically, the Qt application platform and the SQLite relational database system. The second part describes the structure and function of this application. The next part describes the application used to test the functionality of the created application and the designed communication protocol. The last part of the work describes the testing and evaluation of the created application.

KEYWORDS

Qt, SQLite, Database, Application, GUI, Intelligent socket

MICHÁLEK, Jakub. *Software pro ovládání a správu inteligentních prvků domácnosti*. Brno, 2020, 82 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. et Ing. Petr Musil

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Software pro ovládání a správu inteligentních prvků domácnosti“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. et Ing. Petru Musilovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato diplomová práce se zabývá návrhem, vytvořením a popisem klíčových částí aplikace, která umožňuje ovládání a správu inteligentních zásuvek v domácnosti. Správa zásuvek zahrnuje jejich registraci, editaci a dělení do skupin. Data přijatá ze zásuvek jsou ukládaná do databáze. Aplikace obsahuje funkce pro zobrazení těchto dat, jak graficky, tak i textově. Ke komunikaci s touto aplikací je definován vlastní komunikační protokol. Pro ověření funkčnosti tohoto protokolu a aplikace je navržena a naprogramována testovací aplikace.

Důvodem pro výběr tohoto tématu byla potřeba vytvořit nezávislý multiplatformní software pro správu a ovládání inteligentních zásuvek s případnou možností rozvoje na další prvky inteligentní domácnosti. Práce je rozdělena do dvou hlavních částí, teoretické a praktické.

Teoretická část této práce přiblíží čtenáři technologie, které byly využity pro tvorbu výsledné aplikace. První kapitola popisuje aplikační platformu Qt, její vlastnosti, moduly, nástroje a prostředí, ve kterém se aplikace pro tuto platformu vyvíjí. Druhá kapitola seznámí čtenáře s relační databází SQLite, jejími vlastnostmi, výhodami, nevýhodami a využitím.

V praktické části práce je nejprve popsána struktura vytvořené aplikace a databáze, kterou tato aplikace využívá. Dále jsou popsány třídy této aplikace a s nimi související metody, signály, sloty a uživatelská rozhraní. Poté je stejným způsobem popsána testovací aplikace. Následně je popsán navržený komunikační protokol. V poslední části je popsán princip testování vytvořené aplikace a její zhodnocení.

Část I

Teoretická část

1 Aplikační platforma Qt

1.1 Stručná historie

V roce 1990 Haavard Nord a Eirik Chambe-Eng přišli s první myšlenkou vytvoření multiplatformního grafického uživatelského prostředí. K prvnímu věřejnému vydání došlo v roce 1995, obsahovalo zdrojové kódy a bylo pod licencí Qt Free Edition License společnosti Troll Tech.

Při vzniku bylo cíleno na snadnou tvorbu jednotného multiplatformního grafického rozhraní. V dnešní době knihovna Qt však představuje vysoce komplexní softwarovou sadu nejrůznějších modulů k vytváření softwarového obsahu.

V roce 2008 firma Nokia zakoupila od společnosti Trolltech Qt toolkit. Na začátku roku 2011 Nokia prodala práva na provoz podpůrných služeb a prodej licencí QT pro komerční účely firmě Digia. Od roku 2014 do současnosti vlastní práva společnost The Qt Company, dceřiná společnost firmy Digia. [1]

1.2 Obecné informace

Qt je jednou z řady knihoven pro tvorbu grafického uživatelského rozhraní (GUI). Mimo tvorby GUI obsahuje Qt i řadu dalších nástrojů: podpora zpracování XML, práce s databází a multimédii, síťové programování, testování, integruje vykreslovací jádro WebKit atp. Právě tato univerzálnost je jednou z největších předností Qt. Aplikace založené na Qt je dnes možné najít od mobilních telefonů přes vestavěné systémy až po klasické desktopové aplikace (prostředí KDE, Google Earth). [2]

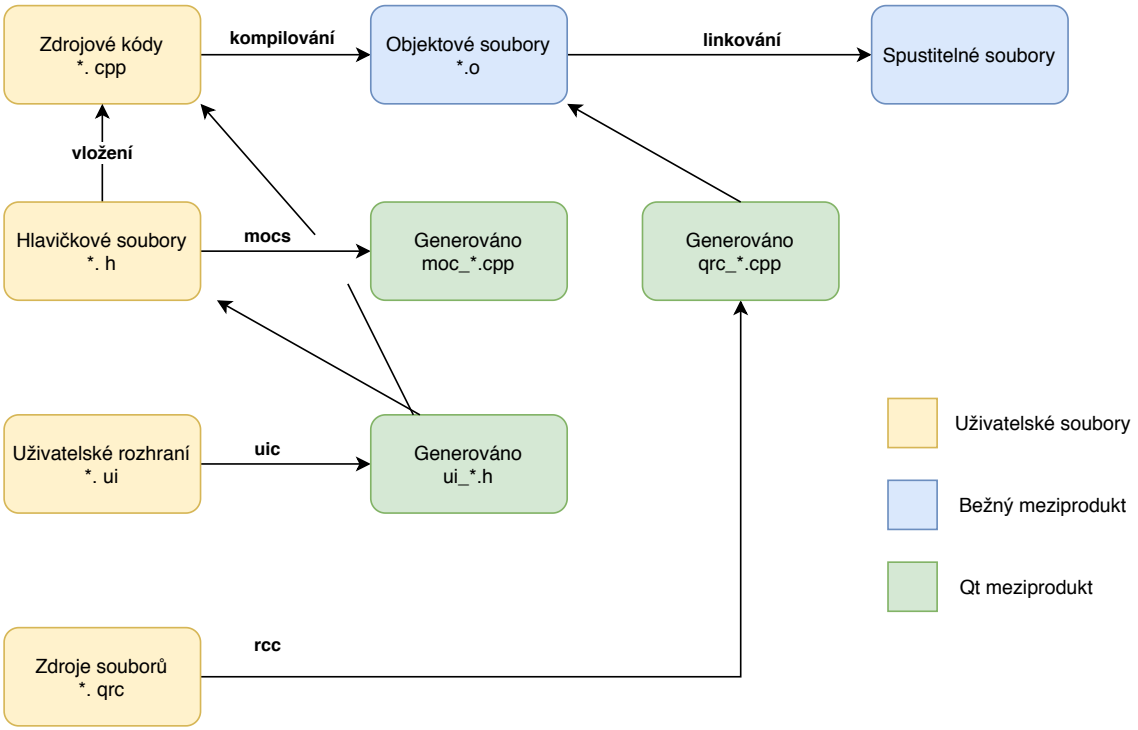
1.2.1 Podporované programovací jazyky

Hlavní podporované jazyky jsou C++ a QML. QML je specifický jazyk pro návrh a tvorbu uživatelských rozhraní. Flexibilita a modularita ovšem zajišťuje i použití jiných programovacích jazyků díky jazykovým vazbám (language binfings). Mezi tyto jazyky patří například Python, Rust, Go, Node.js, Java, C# a další. [2]

1.2.2 Implementace

Qt aplikace lze implementovat dvěma způsoby:

1. Pomocí nástroje Qt Creator. Ten umožňuje zejména velmi pohodlné vizuální programování pomocí sestavování aplikace z připravených grafických prvků. Část kódu je v tomto případě generována automaticky a programátor zejména dopisuje podrobné nastavení vlastností objektů, případně reakce na události. Překlad aplikace proběhne pouhým stisknutím příslušného tlačítka.
2. Pomocí libovolného vývojového nástroje a překladače C++. V tomto případě musí být veškerý kód psán klasickým způsobem. Tento přístup je využíván např. při implementaci serverových aplikací. Úpravy v kódu lze dělat v libovolném editoru a překlad je následně proveden z příkazové řádky. V tomto případě má překlad několik fází:
   (a) Příkazem `qmake -project` je vygenerován projektový soubor pro Qt. Zde je nastavení linkování knihoven, seznam překládaných hlavičkových a implementačních souborů a dodatečná nastavení překladu.
   (b) Příkazem `qmake` je následně vygenerován projektový soubor pro danou platformu (např. Makefile, XCode projekt atp.).
   (c) Poslední fáze překladu proběhne v příslušném vývojovém prostředí, případně příkazem `make` v příkazové řádce. [3]



Obr. 1.1: Překlad programu využívající knihovnu Qt [4]

1.3 Licencování

Platforma je dostupná pod rozmanitou škálou licencí. The Qt Company prodává komerční licence, které se v současné době dělí na dvě odnože. Pro aplikační vývoj a vývoj aplikací fungujících na vestavěných systémech tzv. Embedded Systems.
K dispozici je také bezplatná varianta pod licencemi GPL (verze 2 a 3) a LGPL verze 3.

Při použití komerční verze lze aplikační platformu linkovat staticky, oproti opensource verzi, kde musí být knihovna linkována k projektu dynamicky. Některé moduly mohou mít odlišnou dostupnost pod opensource licencí. [5]

1.4 Vlastnosti Qt

Vlastnosti Qt, které jsou zřejmé už při začátku používání této knihovny jsou multiplatformost a vícejazyčnost. Při tvorbě programu jsou potom důležitými vlastnostmi využití signálů a slotů, řízení událostmi, asynchronní zpracování a vlákna.

1.4.1 Multiplatformost

Jak již bylo zmíněno, Qt je multiplatformí projekt a mezi podporované destopové platformy patří Windows, Linux, Mac OS. Mezi mobilní platformy patří, ať už známější (Android, iOS, Windows Phone) nebo méně rozšířené (BlackBerry, Ubuntu Touch, Sailfish OS). V poslední době se Qt snaží zapojit do vývoje aplikací pro vestavěné systémy. [6]

1.4.2 Podpora národních jazyků a zvyklostí

Aplikace Qt lze snadno překládat do jiných jazyků. Na programátora jsou přitom kladené jen velmi snadno splnitelné požadavky - každý řetězec určený k překladu je potřeba uzavřít do volání funkce `tr()`. Řetězce jsou součástí přímo zdrojového kódu (na rozdíl třeba od Androidu) a dají se exportovat do xml souboru jednoduchou utilitou. Pro překladatele pak existuje v Qt aplikace pro tvorbu překladů. Ta se dá samostatně nainstalovat na počítač překladatele. Překladateli pak stačí poslat xml soubor s texty pro překlad pro doplnění nových výrazů. Překladatelé si kvůli přehledu často otevírají několik jazykových mutací najednou (angličtinu a češtinu), i když vytvářejí třeba překlad do ruštiny. [6]

1.4.3 Signály a sloty

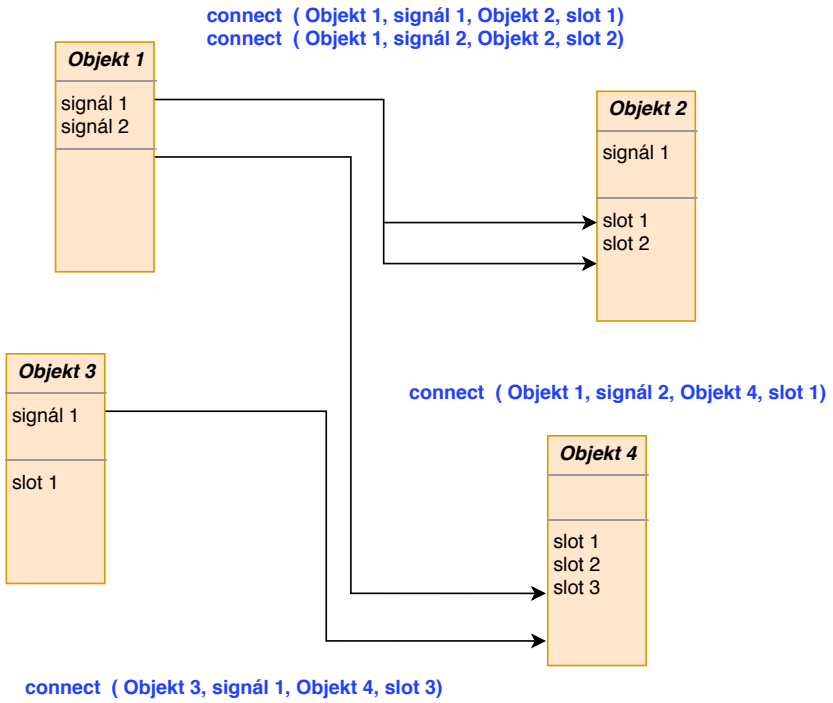
Signály a sloty jsou jeden z nejužitečnějších mechanismů, kterými knihovna Qt disponuje. Rozšiřují možností jazyka C++ a jsou pro Qt typické.

Každý objekt, který v Qt existuje, může emitovat nějaký signál. Například časovač, třída `QTimer`, po vypršení nastavené doby vyvolá signál `timeout()`. Časovač se nestará, kam se signál posílá, jestli se předává do stejného vlákna či do jiného vlákna, kolika jiným objektům se signál předává, nebo jestli se vůbec signál zpracovává.

Každý objekt, který v Qt existuje, může mít nějaké sloty. Slot je běžná metoda třídy, která může být napojená na libovolný signál. Propojení signálů a slotů je dynamické, děje se až za běhu programu. Objekt se nestará, jestli má na své sloty napojené nějaké signály. Propojení může realizovat i jiný objekt.

Důležité také je, že signály fungují bezpečně i mezi vlákny. Qt v takovém případě předává data přes frontu událostí a díky tomu, se nemusejí používat jiné prostředky pro mezivláknovou komunikaci (zamykání, synchronizace).

Při propojení signálů a slotů se provádí i typová kontrola. Protože se propojení realizuje až za běhu programu, překladač není obvykle schopný rozeznat problémy v době překladu, ale aplikace bez potíží upozorní na nesoulad mezi propojovanými signály a sloty při vytváření propojení za běhu aplikace. [6]



Obr. 1.2: Ukázka propojení objektů pomocí signálů a slotů [7]

1.4.4 Řízení událostmi

Se signály a sloty úzce souvisí událostní charakter celé knihovny. Typický program v Qt je celý řízený událostmi a velká část knihovních funkcí a objektů, funguje zcela asynchronně. Narozdíl od systémových utilit napsaných v jazyce C, kde jsou běžné postupy typu "Přečti 10 bajtů ze socketu". V Qt se takovým způsobem nepřečte nic. Pro Qt je typické, že socket dá vědět, že data přišly ze sítě signálem a teprve potom dochází k pokusu o přečtení očekávaných deseti bajtů. [6]

1.4.5 Správa paměti

U jazyku C a C++ je správa paměti složitá. U těchto jazyků se často používají operátory `new` a `delete`, nutnost na každý `new zavolat delete`, vede často k chybám a únikům paměti.

Knihovna Qt přistupuje ke správě paměti jinak. Každý objekt v Qt dědí ze třídy `QObject` a při jeho vytváření se kromě vyjímečných situací novému objektu vždy předává ukazatel na jeho rodiče. Poté při smazání některého objektu se automaticky zlikvidují i všichni jeho potomci.

Qt nepoužívá garbage collector. Pokud je tedy nutné používat sofistikovanější metody pro správu paměti, je možno použít třídy `QSharedPointer` (čítač referencí) nebo `QObjectCleanupHandler` (garbage collector). [6]

1.4.6 Modularita

Knihovna Qt je značně modulární. Celá knihovna se všemi moduly může být velmi rozsáhlá. Díky modularitě, je však možné celou instalaci Qt ořezat a na výsledném zařízení může knihovna zabírat pouze pár MB skutečně potřebných částí. Díky tomuto je možno knihovnu používat také pro embedded aplikace na výkonově velmi omezených platformách. [6]

1.5 Moduly Qt

Mezi nejdůležitější moduly knihovny Qt patří :

- QtCore - tento modul popisuje třídu `QObject`, dále vlákna, externí procesy, kontejnery, časovače a MVC.
- QtGui - zahrnuje to, co se zobrazuje uživateli na obrazovce.
- QtMultimedia - zahrnuje podporu pro video a audio.
- QtNetwork - modul zajišťuje možnost komunikace programu pomocí protokolu IP.
- QtSql - zajišťuje připojení k SQL databázím.

- QtQml, QtQuick - odděluje části grafického uživatelského rozhraní od výkonné části aplikace.

1.5.1 QtCore

Jak již bylo zmíněno, zahrnuje třídu `QObject`. Prakticky všechno v Qt používá jako základní třídu `QObject`. Tato třída zajišťuje spojení signálů a slotů a také správu paměti.

Také obsahuje třídu `QThread`, která je nezbytná pro vytváření vícevláknových aplikací. Signály posílané mezi různými vlákny se automaticky předávají přes frontu. Z toho plyne, že pro většinu aplikací není potřeba používat mutexy, či jiné způsoby ochrany paměti nebo synchronizaci.

1.5.2 QVariant

Obsahuje celý soubor různých datových typů a kontejnerů. Mezi nejužitečnější a často používané patří `QString, QByteArray a QDateTime`.

1.5.3 Kontejnery

Mezi základní kontejnery v Qt patří třídy `QList, QLinkedList, QVector, QStack, QQueue, QSet, QMap a QHash`.

Kontejnery jsou velmi důležitou částí. Umožňují používat různé způsoby organizace dat podle potřeby a dále rozšiřují C++ o jazykové konstrukce, které jinde nejsou obvyklé. Také přinášejí do C++ asociativní pole, přičemž si je možné vybrat algoritmus (hash, btree), kterým je pole organizované.

1.5.4 MVC

Model/view/kontroler návrhové schéma. Objekty třídy `QAbstractItemModel`. Každý takový objekt pak lze zobrazit v nějakém view z GUI části. View samozřejmě může být různé, například aplikace obsahující tabulku s daty a několik různých grafů prezentující různé pohledy na tatáž data. Data (model) se udržuje v paměti jen jednou, zobrazovat se mohou data na více místech. Existují i modely propojené přímo s databázovou tabulkou. Zobrazit a spravovat data v tabulce tak lze přímo v GUI s minimální námahou programátora. Modely mohou pracovat v líném režimu - data se sbírají do modelu a ve view se zobrazují až v momentě, kdy je to potřeba. [6]

1.6 Qml a QtQuick

Původní widgety v knihovně Qt jsou jednoznačně orientované na vývoj desktopových aplikací. Knihovna Qt má však ambice běžet na mnoha různých zařízeních (PC, mobily, tablety, čtečky) s různými operačními systémy (Linux, Windows, Android, iOS). Protože zobrazovací a vstupní možnosti mnoha jiných různých zařízení (mobily, tablety) se od původního desktopu výrazně liší, ukázalo se vhodné oddělit výkonnou část a část GUI. Pro výkonnou část lze dále použít knihovnu Qt a C++, pro část GUI se prosazuje jazyk QML s toolkitem QtQuick.

Jazyk QML je postavený nad JavaScriptem, ale je snadno propojitelný s výkonnou částí aplikace v C++ pomocí signálů a slotů.

Technologie stojící za různými verzemi QtQuick se liší: QtQuick 1 je založená na `QPainter` či `QGraphicsView`, QtQuick2 pak na OpenGL. Liší se i interpreter jazyka JavaScript schovaný za různými verzemi Qt. [6]

1.7 Nástroje

Platforma Qt také nabízí vlastní multiplatformní vývojové nástroje pro zvýšení produktivity a urychlení programování s touto technologií. Součástí je také nástroj pro zobrazení dokumentace, nápovědy a ukázky využití konkrétní technologie v reálné aplikaci.

1.7.1 Qt Creator

Qt Creator je multiplatformní vývojové prostředí (IDE), které umožňuje vývojářům vytvářet desktopové a mobilní aplikace. Je k dispozici jako samostatný balíček nebo v kombinaci s knihovnami Qt a vývojovými nástroji jako kompletní SDK. Qt Creator do sebe integruje Qt Designer. Qt Creator obsahuje:

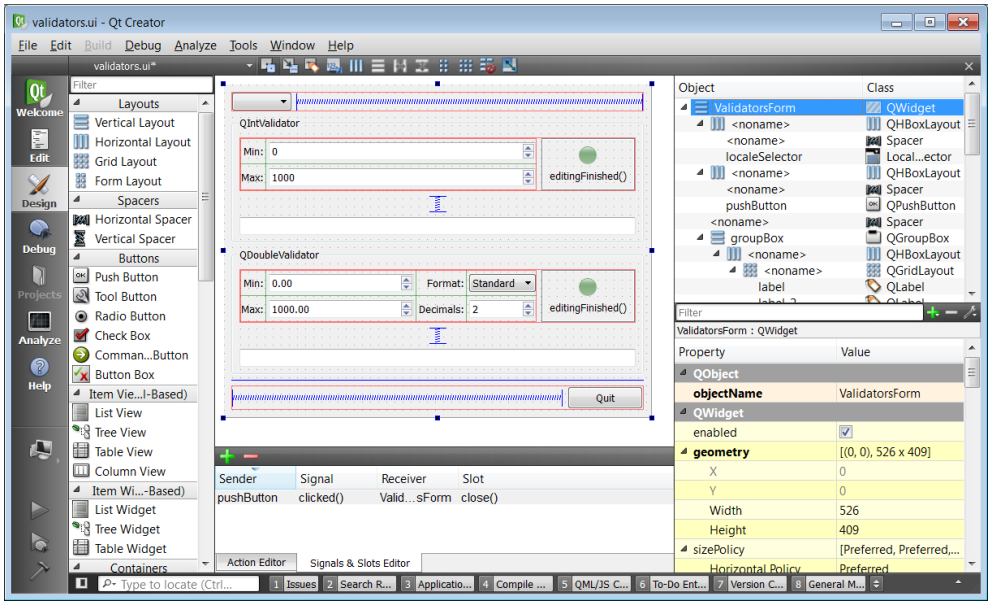
- editor kódu pro C++ a JavaScript,
- prostředí pro návrh UI,
- GDB a CDB debuggery,
- podporu pro správu verzí,
- simulátor pro uživatelská rozhraní mobilních aplikací a další.

Kromě toho má i zabudovaný Qt Assistant, takže kompletní dokumentace je vždy po ruce. Tato součást je propojená s editorem tak, že stačí kurzorem myši najet na Qt objekt nebo metodu, zmáčknout F1 a relevantní dokumentace se objeví v panelu hned vedle kódu. Dále je k dispozici debugger (používá se GDB), podpora různých SCM (Git, Subversion (SVN) a Perforce) a mnoho dalších funkcí.

Aplikace je dostupná buď ve verzi free s LPGL licencí a nebo jako komerční aplikace, kterou je potřeba zakoupit. LPGL licence nemá žádná omezení, je ale potřeba sdílet vytvářené zdrojové kódy. Velkou předností Qt Creator je precizně zpracovaná dokumentace včetně ukázek aplikací i se zdrojovými kódy. Mezi podporující operační systémy patří Windows, Linux i macOS. [8]

1.7.2 Qt Designer

Qt Designer je nástroj pro návrh grafického rozhraní. Umožňuje vytvářet grafické rozhraní prostým přetahováním widgetů do okna. Widgetům poté můžete nastavit vlastnosti a snadno je zakomponovat do různých rozložení (layoutů) a následně dle potřeby propojit požadované signály se sloty. A to vše snadno a graficky. [9]



Obr. 1.3: Ukázka vzhledu vývojového prostředí Qt Creator [8]

2 Databáze SQLite

2.1 Obecné informace

SQLite je relativně malá knihovna, která implementuje kompletní relační databázový systém. Nejedná se však o klasickou klient – server databázi jako jsou například databáze MySQL a PostgreSQL. Naopak je zde databáze uložena v souboru a předpokládá se tedy, že k datům přistupuje software, který běží na stejném počítači či serveru, ve kterém jsou uložena data.

Základní myšlenkou projektu SQLite je jednoduchost. Jak již bylo uvedeno, samotná databáze s daty je uložena v klasickém souboru na disku. Dále je třeba zmínit to, že SQLite prakticky vůbec nepotřebuje žádnou konfiguraci. Databázový software je absolutně spokojen, pokud je mu sděleno, s jakým databázovým souborem se hodlá pracovat. Samozřejmě je nutné, aby uživatel, pod kterým poběží software přistupující k datům, měl k databázovému souboru náležitá práva. Neznamená to však, že SQLite nelze vůbec konfigurovat. Pokud je potřeba, může se chování vytvořené instance SQLite upravit pomocí příkazu PRAGMA.

Autorem tohoto projektu je programátor D. Richard Hipp, který se rozhodl své dílo šířit pod public domain licencí, což může být považováno za jednu z výhod tohoto databázového systému, jelikož v případě sporného použití tohoto projektu se uživatel nemusí vůbec zabývat licenčními podmínkami. SQLite je použit například v následujících aplikacích: Google Chrome, Safari, Opera, Android Browser, Mozilla Firefox a Thunderbird, Skype a Adobe Reader. SQLite je podporován následujícími organizacemi: Facebook, Expensify, Mozilla, Navagiation Data Standard, Bentley a Bloomberg. SQLite lze použít na všech GNU/Linuxových distribucích, Windows i OS X. Z toho jasně vyplývá, že možnosti použití SQLite jsou velmi rozmanité a multiplatformní. [10]

2.2 RDBMS

Označení databáze je vlastně nepřesné a v odborné literatuře se je možné spíše setkat s označením RDBMS (Relation DataBase Management System). V překladu "systém řízení báze dat", což je velmi dlouhé označení, a proto bude dále využito označení databázový stroj nebo RDBMS. Databázový stroj (zde tedy SQLite) není jen úložiště dat. Jedná se o velmi sofistikovaný a odladěný nástroj, který za uživatele řeší spoustu problémů a zároveň je extrémně jednoduchý k použití. Pro komunikaci s databází je využit jazyk SQL, který je reprezentován v podstatě lidsky srozumitelnými větami. Spolu s ukládáním dat je ale třeba dále řešit mnoho dalších věcí jako jsou zabezpečení nebo optimalizace výkonu.

RDBMS toho ale dělá ještě mnohem více. Řeší za uživatele problém současné editace stejné položky několika uživateli ve stejný okamžik, který by jinak mohl zapříčinit nekonzistenci databáze. RDBMS data v tomto případě zamkne a odemkne až po vykonání zápisu. Dále umožňuje spojovat několik dotazů do transakcí, kdy se série dotazů vykoná vždy celá nebo vůbec. Nemůže nastat situace, že by se vykonala jen část. Tyto vlastnosti databázového stroje jsou shrnovány zkratkou ACID, která je vysvětlena v další sekci. [11]

2.3 ACID

ACID je akronym slov Atomicity (nedělitelnost), Consistency (validita), Isolation (izolace) a Durability (trvanlivost). Jednotlivé složky mají následující význam:

- **Atomicity** - Operace v transakci se provedou jako jedna atomická (nedělitelná) operace. To znamená, že pokud nějaká část operace selže, vrátí se databáze do původního stavu a žádné části transakce nebudou provedeny. Reálný příklad je např. převod peněz na bankovním účtu. Pokud se nepodaří peníze odečíst z jednoho účtu, nebudou ani připsány na účet druhý. Jinak by byla databáze v nekonzistentním stavu. Pokud by se uživatelé starali o práci s daty sami, mohlo by se jim to velmi jednoduše stát.
- **Consistency** - Stav databáze po dokončení transakce je vždy konzistentní, tedy validní podle všech definovaných pravidel a omezení. Nikdy nenastane situace, že by se databáze nacházela v nekonzistentním stavu.
- **Isolation** - Operace jsou izolované a navzájem se neovlivňují. Pokud se sejde v jeden okamžik více dotazů na zápis do stejného řádku, jsou vykonávány postupně, jako ve frontě.
- **Durability** - Všechna zapsaná data jsou okamžitě zapsána na trvanlivá úložiště (na pevný disk), v případě výpadku elektrické energie nebo jiného přerušení provozu RDBMS vše zůstane tak, jak bylo těsně před výpadkem.

Databáze se tedy chová jako černá skříňka, se kterou daná aplikace komunikuje a do které ukládá veškerá data. Její použití je velmi jednoduché a je odladěna tak, že pokud by chtěl uživatel vytvořit podobný systém, bylo by to pro něj velmi náročné. Uživatel se nemusí starat o to, jak jsou data fyzicky uložena. S databází komunikuje pomocí jednoduchého dotazovacího jazyka SQL. O databázi občas hovoříme jako o 3. vrstvě aplikace (1. vrstva je uživatelské rozhraní, 2. vlastní logika aplikace, 3. je právě datová vrstva). [11]

2.4 Výhody SQLite

Výhodou SQLite je značná jednoduchost použití, malá velikost, databáze uložená v souboru, většinou nulová potřeba jakékoli konfigurace, nulová cena a public domain licence.

Další výhodou je také jistá míra paralelního přístupu, kterou SQLite na rozdíl od konkurenčních projektů nabízí. SQLite umožňuje, aby z jedné databáze četlo najednou více instancí. Tento fakt však neplatí pro zápis.

Další výhodou je i fakt, že vzhledem k velice malé velikosti SQLite je až obdivuhodné, že podporuje skoro celý standard SQL-92. SQLite podporuje dokonce i cizí klíče, jejichž vynucení je však potřeba aktivovat použitím příkazu PRAGMA.

Další výhoda se může jevit také v tom, že je databáze uložena v jednom jediném souboru, a proto je její zálohování a případně i distribuce opravdu snadným úkolem. SQLite se z tohoto důvodu hojně využívá zejména v desktopových aplikacích. Díky tomu, a její rychlosti se ale uživatel musí smířit i s jejími nedostatky. [10]

2.5 Nevýhody SQLite

Hlavní nevýhodou SQLite lze shledat nízký výkon, který se projeví hlavně při zápisu většího množství dat. SQLite chápe každý insert jako vlastní transakci a podle dokumentace transakci dokončí opravdu až tehdy, kdy jsou data bezpečně fyzicky uložena na disku. Na jednu stranu je dobré, že se uživatel může v podstatě spolehnout na bezpečné uložení dat, ale na stranu druhou je nízký výkon silně obtěžující a hlavně omezující faktor. Tento problém lze vyřešit tím, že se uzavře více insertů do vlastní transakce (begin transaction a commit). Tím se docílí velmi výrazného zrychlení v případě, že se na databázi pouští větší množství insertů.

Nevýhodou je také již zmíněné uzavření celé databáze v případě probíhajícího zápisu. To znamená, že v podstatě nelze realizovat paralelní zápis do databáze. Také nelze z databáze číst, pokud je do ní zrovna zapisováno. [10]

2.6 Využití SQLite

SQLite se hodí pro malá nasazení, kdy k datům přistupuje pouze jeden fyzický stroj a to nejlépe pouze z jedné instance či vlákna. Typickým příkladem tedy může být jednoduché osobní účetnictví, malý pokladní terminál, správce osobních financí či aplikace pro finanční analýzu.

SQLite se také hodí jako databáze pro malý až střední web. Zde můžete těžit z faktu, že databáze je uložena v souboru na disku, takže můžete případně ušetřit nějaké finance vybráním hostingu bez relační databáze. To samozřejmě předpokládá, že programovací jazyk použitý v rámci hostingu (např. PHP) bude mít nainstalovány potřebné extenze a moduly.

SQLite bývá používán jako cache dat nad některým větším databázovým systémem (Postgresql, Oracle, atd.). To znamená, že vlastní data jsou uložena ve větším databázovém systému a SQLite obsahuje jen jejich jakýsi obraz, který může být i nějakým způsobem agregovaný. Při změně vlastních dat je samozřejmě nutné data v SQLite, který slouží jako cache, obnovit. Těží se zde hlavně z rychlosti čtení, který SQLite nabízí, a také z faktu, že nasazením cache SQLite, který bude z pravidla běžet na jiném stroji než vlastní velká databáze, dojde ke snížení zátěže hlavního databázového serveru. [10]

2.7 Podporované datové typy

K tomu, aby uživatel úspěšně vytvořil tabulku v SQLite je potřeba znát datové typy, které SQLite nabízí. Podporované datové typy jsou:

- NULL – hodnota null,
- INTEGER – celočíselná hodnota,
- REAL – číslo s plovoucí desetinnou čárkou,
- TEXT – textový řetězec,
- BLOB – binární data.

Je nutno uvést, že v SQLite je v podstatě jedno, s jakým datovým typem je sloupec v tabulce vytvořen. Do jakéhokoli sloupce mohou být uložena jakákoli data, a to bez ohledu na to, jaký typ sloupce byl uveden při vytvoření tabulky. Existuje zde však jedna výjimka, kterou je celočíselný primární klíč (INTEGER PRIMARY KEY). V dokumentaci SQLite je uvedeno, že to není chyba a že typ dat definovaný pro sloupec při vytvoření tabulky má být vnímán jako jakési doporučení typu dat, která mají být v daném sloupci obsažena. S ohledem na typické nasazení SQLite a jeho důraz na jednoduchost lze brát tuto vlastnost jako pozitivní věc, která by však u vyspělé databáze byla spíše na škodu. SQLite sám s touto situací opravdu počítá. Pokud bude například ve sloupci s typem INTEGER i textová data a zavolá se na tomto sloupci funkce SUM, dojde ke korektnímu sečtení číselných hodnot a textová data zůstanou nedotknuta. [10]

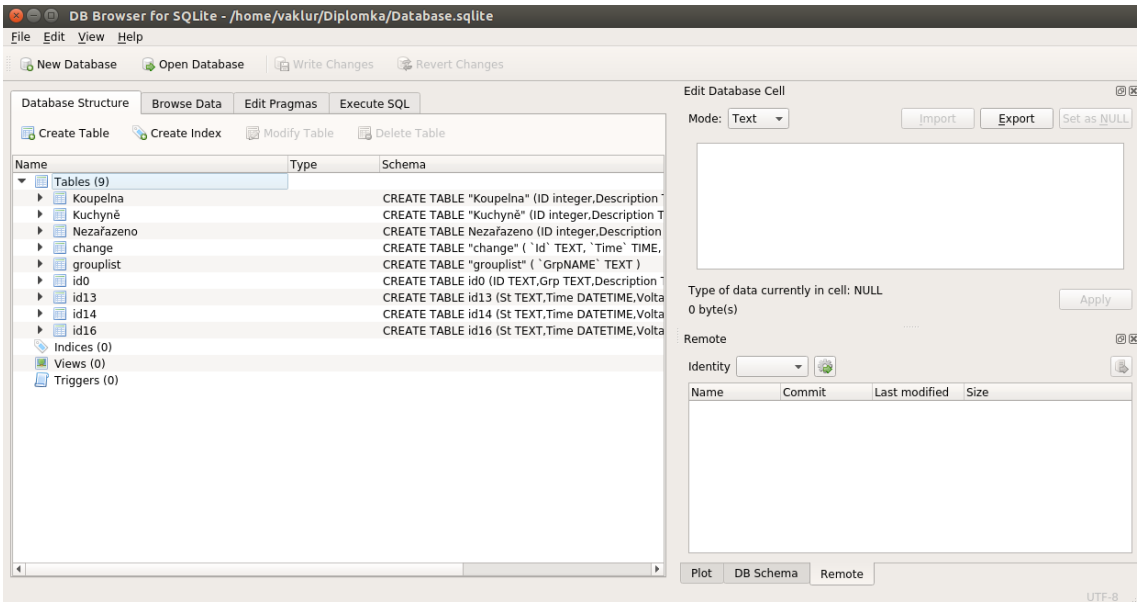
2.8 DB Browser

DB Browser for SQLite (DB4S) je vysoce kvalitní, vizuální nástroj s otevřeným zdrojovým kódem pro vytváření, návrh a úpravu databázových souborů kompatibilních s SQLite. Je schopen pracovat s databázemi ve formátu SQLite jejichž koncovky jsou obvykle *.db, *.sqlite, *.sqlite3 nebo *.db3.

DB4S je určen pro uživatele a vývojáře, kteří chtějí vytvářet, prohledávat a upravovat databáze. DB4S používá známé tabulkové rozhraní a složité příkazy SQL se nemusí učit. [12]

Nástroje programu:

- neomezené množství připojení,
- snadné přepínání mezi databázemi, ke kterým jste připojeni,
- export dat do CSV, Excelu nebo HTML souborů,
- tvorba SQL s širokou škálou databází,
- tvorba LOG souborů,
- záznam historie.



Obr. 2.1: Ukázka vzhledu programu DB Browser for SQLite

Část II

Praktická část

3 Aplikace IController

Následující část diplomové práce popisuje strukturu, vzhled a jednotlivé třídy, které aplikace obsahuje. Jednotlivé části aplikace jsou popsány pohledem programátora a také uživatele. Vzniklá aplikace nese jméno IController, zkratka pro Intelligent Controller. V dalších částech práce bude používán název inteligentní ovladač.

Pro vývoj aplikace byl zvolen programovací jazyk C++ s aplikační platformou Qt. Aplikační platforma Qt byla zvolena z důvodu tvorby grafického uživatelského rozhraní a usnadnění vývoje s programovacím jazykem C++. K ukládání dat byla použita relační databáze SQLite z důvodu rychlého přístupu k datům a jejich třídění. Aplikace byla vytvořena ve vývojovém prostředí Qt Creator. Ke správě databáze byl využit program DB Browser.

3.1 Popis aplikace

Aplikace umožňuje uživateli zejména zobrazování dat z inteligentních zásuvek v reálném čase, jak textově, tak i graficky. Dále je možné jednotlivé zásuvky přidávat, upravovat, odstraňovat a seskupovat do skupin na základě jejich umístění nebo jiných preferencí uživatele. Tyto skupiny se dají také přidávat a upravovat. Také lze přesouvat jednotlivé zásuvky z jedné skupiny do druhé.

Uživatel také může pozorovat chování zásuvek v určitém časovém období. A to jednak pomocí grafů, které uživateli zobrazují vývoj jednotlivých parametrů v čase a i formou statistik, které data průměrují. Statistiky také zobrazují celkovou spotřebu elektrické energie a počet aktivních záznamů v daném období.

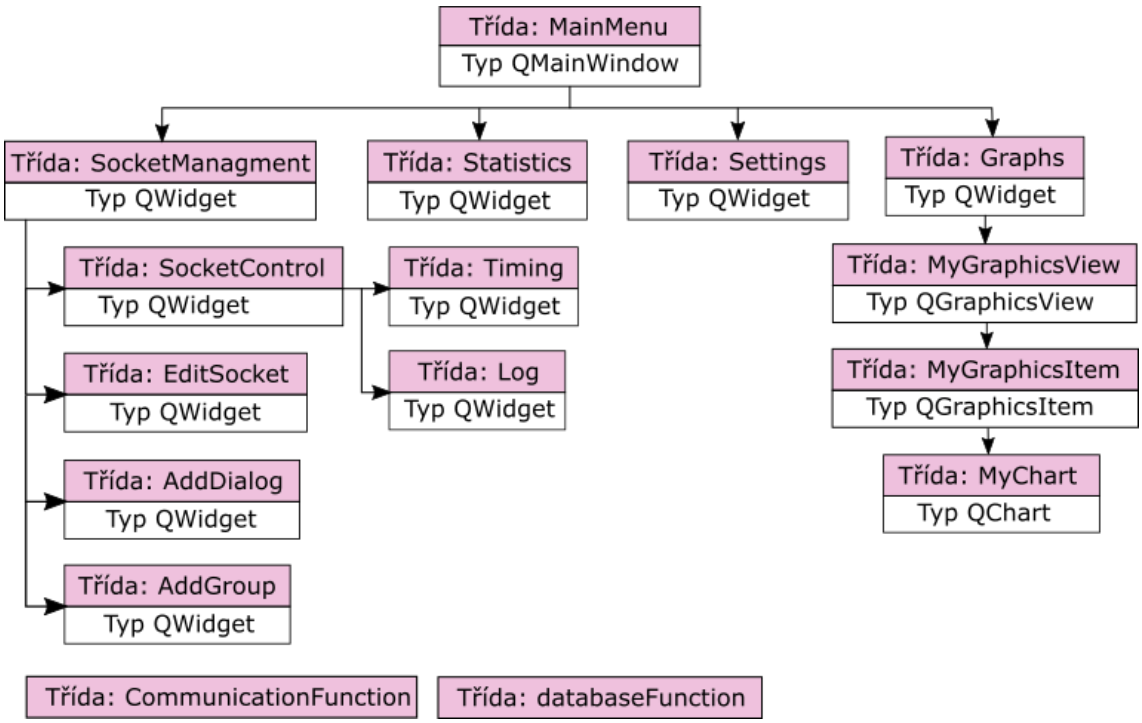
Další důležitou funkcí aplikace je možnost jednotlivé zásuvky vzdáleně ovládat. Buď přímo posláním požadavku na zapnutí/ vypnutí uživatelem nebo nastavením časovače, který na základě zvoleného času a data automaticky odešle požadavek ke změně stavu zásuvky.

Z důvodu, že zdrojový kód obsahuje několik stovek řádků kódu, nejsou v práci uvedeny zdrojové kódy metod, ale spíše vysvětlen jejich princip.

3.2 Struktura aplikace

Aplikace je z velké části tvořena z jednotlivých tříd, které dědí vlastnosti ze třídy `QWidget`. Každá z těchto tříd má sobě vlastní grafické uživatelské rozhraní. Dále třídou `databaseFunction`, která obsahuje metody pro ukládání dat do databáze a třídou `CommunicationFunction`, která obsahuje metody pro komunikaci s kolektorem dat.

Třídy typu `QWidget` na sebe navazují a zdánlivě tvoří stromovou strukturu, kdy při spuštění aplikace je jako první zobrazena třída `MainMenu`, která zabezpečuje veškerý chod aplikace a na další třídy se dostáváme z ní. Nejdůležitější třídy budou podrobně popsány v následujících částech práce. Struktura aplikace je znázorněna na obrázku 3.1.



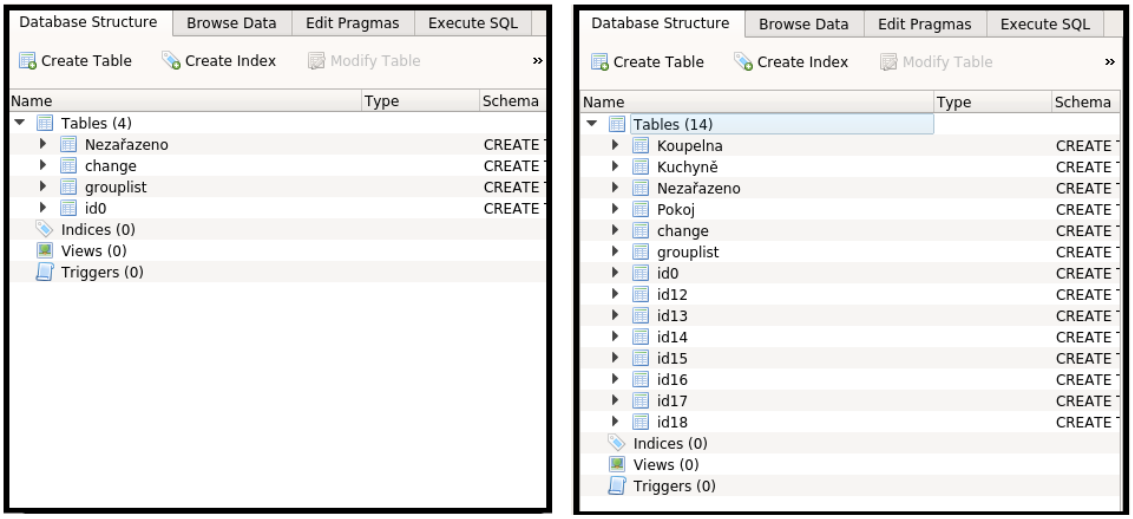
Obr. 3.1: Struktura aplikace IController

3.3 Databáze

V aplikaci je k ukládání dat používaná SQLite databáze s názvem IControllerDatabase. Cesta k této databázi se volí přímo v nastavení a bez připojené databáze poskytuje aplikace velmi omezené funkce.

Tabulky, které databáze obsahuje je možné rozdělit do dvou skupin. První skupinou jsou tabulky, které udržují informaci o přidaných objektech. Objekty v tomto případě zahrnují zásuvky, skupiny a časovače. Druhou skupinou jsou tabulky, které ukládají data těchto objektů. Například se jedná o tabulku vytvořenou pro každou přidanou zásuvku a ve které jsou ukládány všechny přijaté zprávy s daty. V následujících podkapitolách jsou podrobně popsané jednotlivé tabulky, jejich struktura, význam a využití.

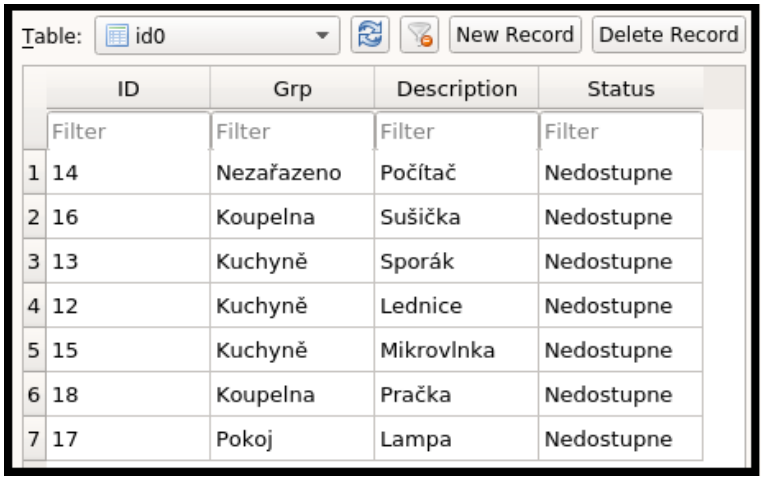
Při prvním spuštění aplikace jsou do databáze automaticky přidány tabulky `id0`, `grouplist`, `change` a `Nezařazeno`. Tyto tabulky se v aplikaci nedají smazat a jsou nutné pro správný chod všech funkcí aplikace. Při chodu aplikace jsou poté přidávány tabulky `idX a group` v závislosti na akcích, které provádí uživatel. Obsah databáze při prvním spuštění aplikace a při jejím chodu, je vidět na obrázku 3.2.



Obr. 3.2: Obsah databáze při prvotním spuštění aplikace (vlevo) a při jejím chodu (vpravo)

3.3.1 Tabulka id0

Tato tabulka je nejdůležitější tabulkou pro chod celé aplikace. Jsou v ní uloženy záznamy o všech přidaných zásuvkách. Konkrétně každý záznam obsahuje identifikátor zásuvky, její popis, skupinu a poslední známý stav, ve kterém se nacházela. Tyto záznamy jsou poté v samotné aplikaci využívány především k zobrazení registrovaných zásuvek, jejich editaci a mazání.

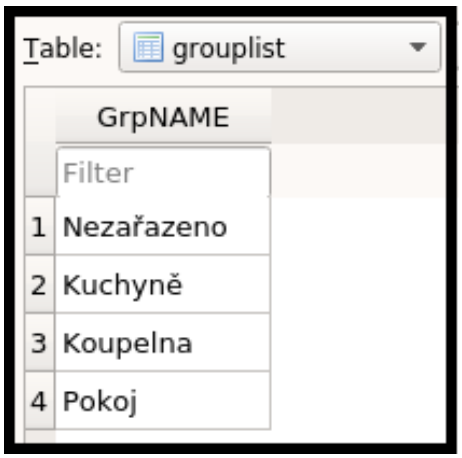


	ID	Grp	Description	Status
	Filter	Filter	Filter	Filter
1	14	Nezařazeno	Počítač	Nedostupne
2	16	Koupelna	Sušička	Nedostupne
3	13	Kuchyně	Sporák	Nedostupne
4	12	Kuchyně	Lednice	Nedostupne
5	15	Kuchyně	Mikrovlnka	Nedostupne
6	18	Koupelna	Pračka	Nedostupne
7	17	Pokoj	Lampa	Nedostupne

Obr. 3.3: Příklad obsahu tabulky `id0`

3.3.2 Tabulka grouplist

V této tabulce jsou uloženy záznamy o všech přidaných skupinách. Každý záznam obsahuje pouze název skupiny.



	GrpNAME
	Filter
1	Nezařazeno
2	Kuchyně
3	Koupelna
4	Pokoj

Obr. 3.4: Příklad obsahu tabulky `grouplist`

3.3.3 Tabulka group

Tabulka `group` je přidána do databáze vždy při přidání nové skupiny. Název group je nahrazen názvem vytvořené skupiny. Záznamy v této tabulce reprezentují zásuvky v dané skupině, konkrétně je záznam tvořen identifikátorem a popisem dané zásuvky.

Table: Kuchyně

	ID	Description
	Filter	Filter
1	13	Sporák
2	15	Mikrovlnka
3	12	Lednice

Obr. 3.5: Příklad obsahu tabulky `group`

3.3.4 Tabulka change

V této tabulce jsou uložené záznamy, které slouží pro automatické ovládání zásuvek. Nový záznam je přidán, pokud je v aplikaci zvolen nový časovač. Tento záznam obsahuje identifikátor zásuvky, čas, informaci o změně stavu, den změny stavu a pomocný sloupec, který reprezentuje den změny stavu vyjádřený číslem.

Table: change

	Id	Time	StChng	Days	DaysHlp
	Filter	Filter	Filter	Filter	Filter
1	14	00:00:00	Vypnout	Čt So	46
2	16	00:00:00	Zapnout	Pá	5
3	16	01:00:00	Vypnout	Pá	5
4	16	01:00:00	Vypnout	Po	1

Obr. 3.6: Příklad obsahu tabulky `change`

3.3.5 Tabulka idX

Tabulka `idX`, kde X reprezentuje identifikátor, je vytvořena vždy při přidání nové zásuvky do aplikace. Obsahuje záznamy, které reprezentují přijaté datové zprávy z dané zásuvky. Každý záznam obsahuje čas příjmu zprávy, stav zásuvky a na ní změřené veličiny. Záznamy z této tabulky slouží zejména k zobrazování aktuálních dat ze zásuvky uživateli. Dále jsou využívány ke tvorbě grafů a statistik.

Table: id14

	St	Time	Voltage	Current	Frequence	Power
	Filter	Filter	Filter	Filter	Filter	Filter
5533	on	2020-04-21...	244.0	5.0	50.0	1220.0
5534	on	2020-04-21...	259.0	3.0	50.0	777.0
5535	on	2020-04-21...	241.0	2.0	50.0	482.0
5536	on	2020-04-21...	241.0	3.0	50.0	723.0
5537	on	2020-04-21...	250.0	7.0	50.0	1750.0
5538	on	2020-04-21...	244.0	0.0	50.0	0.0
5539	on	2020-04-21...	259.0	2.0	50.0	518.0
5540	on	2020-04-21...	249.0	0.0	50.0	0.0
5541	on	2020-04-21...	253.0	12.0	50.0	3036.0
5542	on	2020-04-21...	244.0	3.0	50.0	732.0
5543	on	2020-04-21...	247.0	14.0	50.0	3458.0
5544	on	2020-04-21...	249.0	4.0	50.0	996.0
5545	on	2020-04-21...	252.0	10.0	50.0	2520.0
5546	on	2020-04-21...	242.0	10.0	50.0	2420.0
5547	on	2020-04-21...	249.0	5.0	50.0	1245.0
5548	on	2020-04-21...	244.0	11.0	50.0	2684.0
5549	on	2020-04-21...	240.0	14.0	50.0	3360.0
5550	on	2020-04-21...	247.0	8.0	50.0	1976.0
5551	on	2020-04-21...	244.0	10.0	50.0	2440.0
5552	on	2020-04-21...	248.0	8.0	50.0	1984.0
5553	on	2020-04-21...	256.0	8.0	50.0	2048.0
5554	on	2020-04-21...	248.0	10.0	50.0	2480.0
5555	on	2020-04-21...	240.0	14.0	50.0	3360.0
5556	on	2020-04-21...	253.0	3.0	50.0	759.0
5557	on	2020-04-21...	252.0	12.0	50.0	3024.0
5558	on	2020-04-21...	240.0	6.0	50.0	1440.0
5559	on	2020-04-21...	250.0	12.0	50.0	3000.0
5560	on	2020-04-21...	243.0	15.0	50.0	3645.0

Obr. 3.7: Příklad obsahu tabulky `idX`

3.4 Globální proměnné

Důležitou součástí pro chod aplikace jsou globální proměnné. Ty jsou definované ve zdrojovém souboru `globals.c` a v hlavičkovém souboru `globals.h`. V aplikaci jsou proměnné rozdělené do tří skupin, a to na proměnné sloužící k:

- Udržování stavu o dostupných zásuvkách a aktuální zobrazované zásuvce, do kterých patří:
  - `listID` - udržuje v sobě seznam všech identifikátorů zásuvek uložených v databázi,
  - `controlListID` - udržuje počet nepřijatých odpovědí na požadavky k zaslání dat pro každou zásuvku,
  - `actualID` - udržuje hodnotu identifikátoru aktuální zobrazované zásuvky,
  - `open_socket_control` - udržuje v sobě počet otevřených instancí třídy `SocketControl`.
- Přístupu do databáze a komunikaci po sériové lince, do kterých patří:
  - `datab1` - udržuje v sobě nastavení databáze a slouží pro ukládání dat do databáze a získávání dat z ní,
  - `serial` - udržuje v sobě nastavení sériového portu a slouží k přijímání a odesílání dat přes sériovou linku,
  - `df` - slouží k volání metod z třídy `databaseFunction`.
- Nastavení aplikace, a to konkrétní cesty k databázi, použitého sériového portu a časovače odesílání dat, do kterých patří:
  - `port` - udržuje v sobě hodnotu aktuálního nastaveného sériového portu,
  - `path` - udržuje v sobě cestu k aktuální používané databázi,
  - `sendTimeMSec` - udržuje v sobě hodnotu času v milisekundách, se kterým jsou odesílány požadavky do kolektoru dat.

Výpis 3.1: Zdrojový kód definující globální proměnné

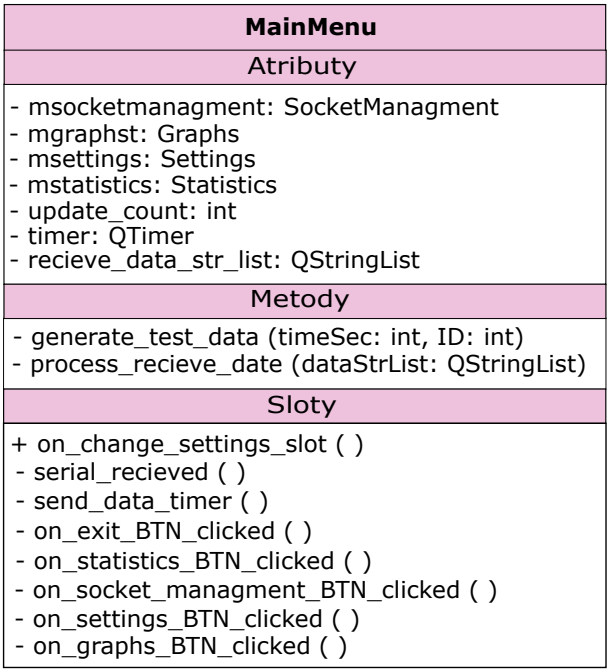
```
{
//******Global Variables*********//

//Manage IDs
QStringList listID = {};
QList <int> controlListID = {};
QString actualID ="0";
int open_socket_control=0;

// Database and serialPort
QSqlDatabase datab1;
QSerialPort *serial;
databaseFunction df;

// Settings
QString port="";
QString path="";
int sendTimeMSec =1000;
}
```

3.5 Popis jednotlivých tříd dědících ze třídy QWidget

3.5.1 Třída MainMenu

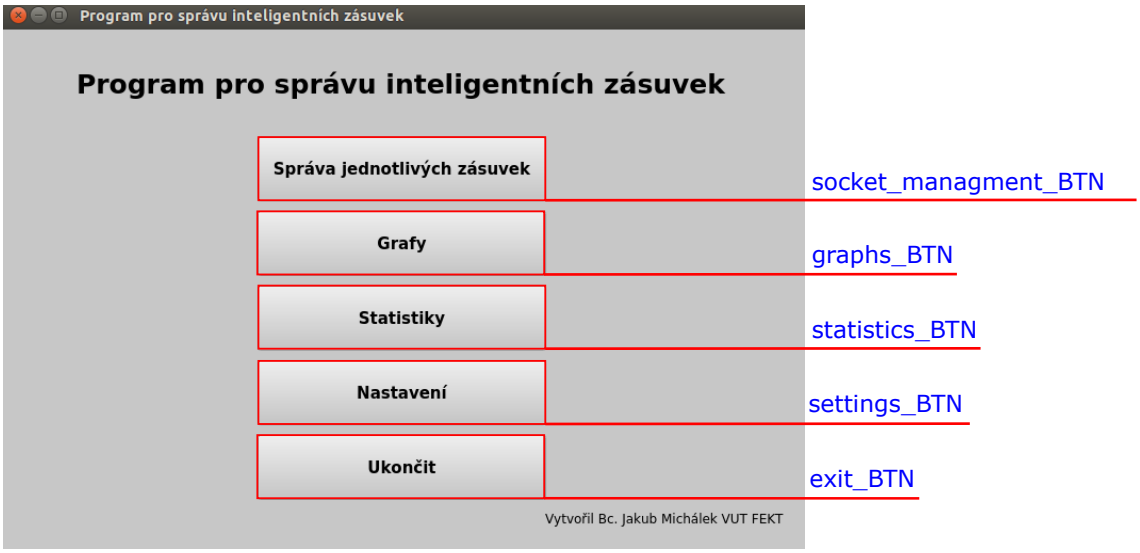
Třída `MainMenu`, která je potomkem třídy `QMainWindow`, reprezentuje hlavní třídu celé aplikace. Po spuštění aplikace se tato třída stará o načtení hlavní obrazovky. Slouží jako rozcestník do dalších částí aplikace a zajišťuje přijímaní dat z kolektoru dat a odesílání požadavků do kolektoru dat.

Při vytvoření objektu této třídy je načteno uložené nastavení pomocí třídy `Settings`. Dále jsou nastaveny výchozí hodnoty globálních proměnných a dochází k inicializování používané databáze a sériového portu. Pokud dojde k tomu, že sériový port není otevřen nebo se nepodařilo spojit s databází, je uživatel upozorněn dialogy na to, že funkčnost aplikace je značně omezena. Také je spuštěn časovač, který slouží k periodickému odesílání požadavků do kolektoru dat.



Obr. 3.8: Třída `MainMenu`

Grafické uživatelské rozhraní



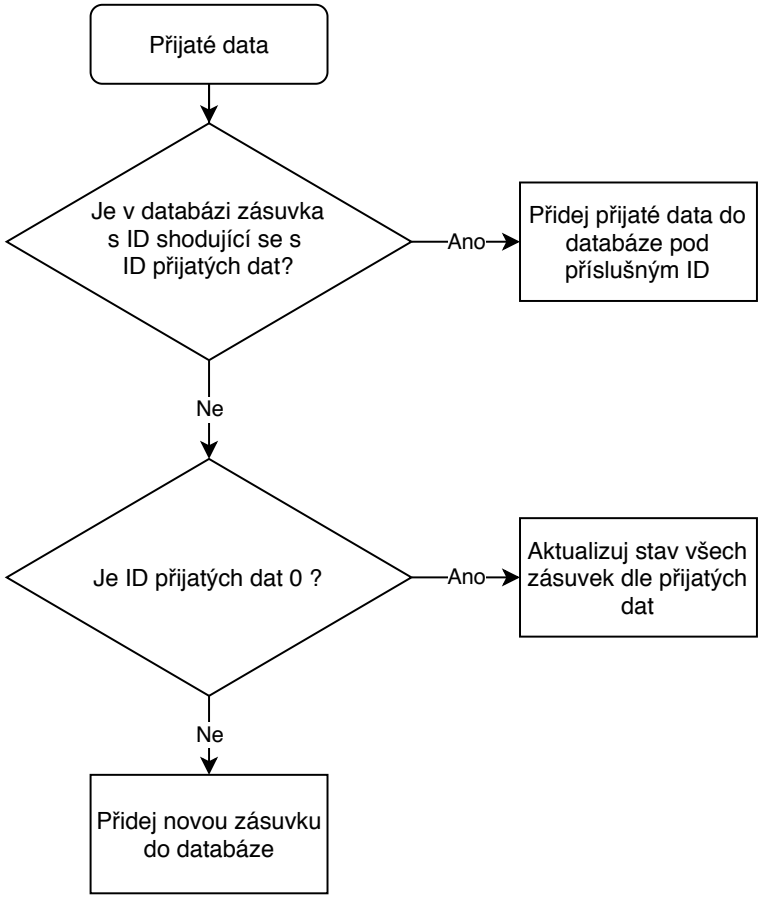
Obr. 3.9: Vzhled GUI třídy `MainMenu`

Uživatelské rozhraní je tvořeno pěti tlačítky. Kliknutím na čtyři z nich, jmenovitě `socket_managment_BTN`, `graphs_BTN`, `statistics_BTN` a `settings_BTN` dojde k vytvoření instance daných tříd a zobrazení příslušných widgetů. Kliknutí na tlačítko `exit_BTN` způsobí zobrazení dialogu zda chce uživatel skutečně aplikaci ukončit. Při potvrzení tohoto dialogu je nejprve ukončeno spojení s databází, zavřen sériový port a ukončen časovač pro odesílání dat. Poté dojde k ukončení celé aplikace.

Metody

- `generate_test_data` - metoda slouží ke generování umělých dat pro ověření funkčnosti aplikace. Vstupními parametry je identifikátor zásuvky, pro kterou mají být data generována a čas v sekundách, který určuje, jak velký časový vzorek dat bude od aktuálního času do minulosti vygenerován. Vygenerovaná data jsou uložena do databáze k příslušnému ID zásuvky.

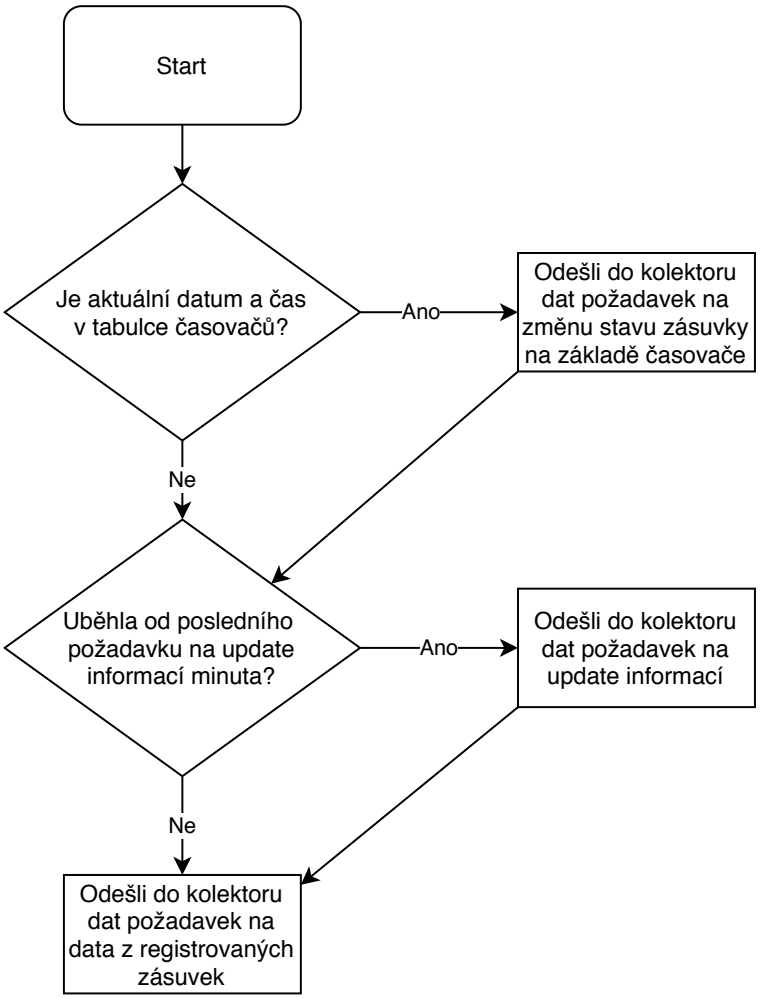
- `process_recieve_data` - metoda, která slouží ke zpracování přijaté zprávy ze sériového portu. Jejím vstupním parametrem je celá přijatá zpráva.
  Pokud je identifikátor ve zprávě shodný s identifikátorem některé ze zásuvek, které už jsou v databázi, jsou data obsažená ve zprávě přidány do databáze k příslušnému identifikátoru.
  Pokud je identifikátor ve zprávě rovný nule, značí to, že se jedná o zprávu, která obsahuje všechny zásuvky a jejich stavy, které jsou dostupné pro kolektor dat.
  Pokud identifikátor nesplňuje žádnou z předešlých podmínek, jedná se o data, která přišla ze zásuvky, která není v databázi. Zásuvka s tímto identifikátorem je přidána do databáze a do skupiny `Nezařazeno`.
  Na obrázku 3.10 je zobrazen vývojový diagram této metody.



Obr. 3.10: Vývojový diagram metody `process_recieve_data`

Sloty a signály

- `send_data_timer` - slot je volán při vypršení časovače pro odesílání dat. Nejprve dojde k ověření, zda se v tabulce pro automatické změny nachází záznam s aktuální datem a časem. Pokud ano, je poslán do kolektoru dat požadavek na změnu stavu zásuvky s identifikátorem odpovídající záznamu.
  Dále dochází k ověření, zda od posledního požadavku na aktualizaci informací z kolektoru dat uběhla jedna minuta. Pokud tomu tak je, odešle se nový požadavek.
  Při vypršení časovače se také pokaždé odesílá požadavek na nová data ze zásuvek registrovaných v aplikaci. Vývojový diagram chování tohoto slotu je vidět na obrázku 3.11.

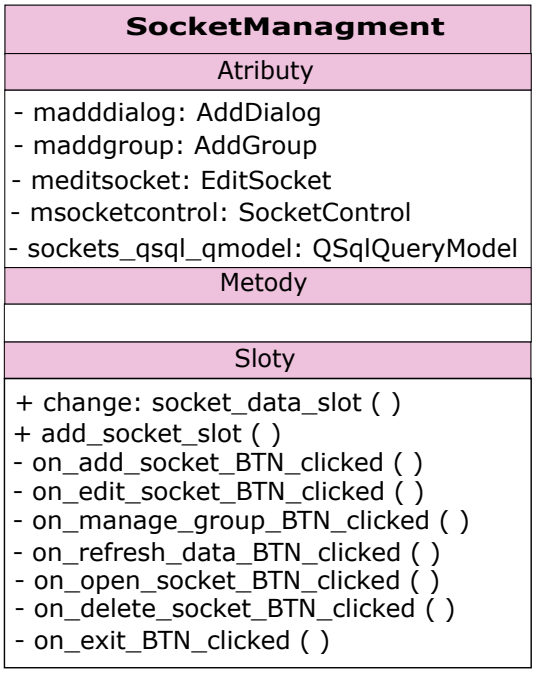


Obr. 3.11: Vývojový diagram slotu `send_data_timer`

- `on_change_settings_slot` - slot, který se volá pokud dojde ke změně nastavení ve widgetu `Settings`. Nové nastavení je nahráno do globálních proměnných. Na základě tohoto nastavení dojde k znovu inicializování používané databáze a sériového portu. Také dojde k přenastavení časovače pro odesílání zpráv.
- `serial_recieved` - volá se při přijetí dat na sériovém portu. Data jsou ukládána do proměnné a pokud přijatá data obsahují řetězec „\n“ je přijatá zpráva považována za celou. Tato zpráva je poté předána metodě `process_recieve_data` k dalšímu zpracování.

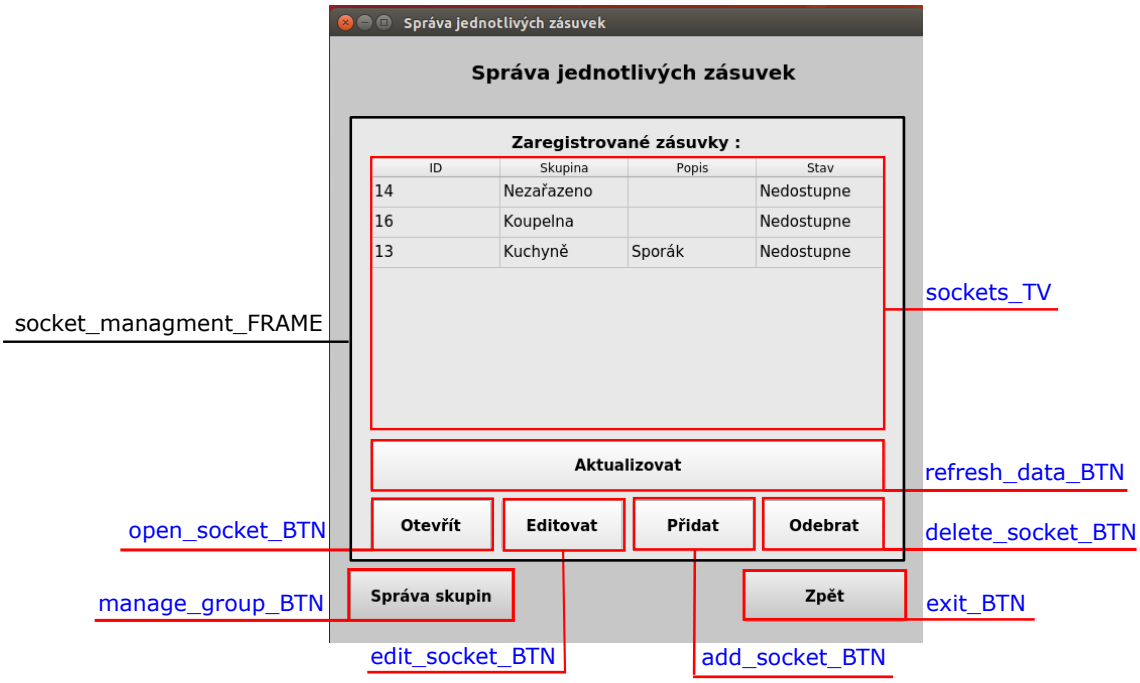
3.5.2 Třída SocketManagment

Třída `SocketManagment` slouží ke správě registrovaných zásuvek. Konkrétně se z ní je možné dostat k widgetům pro přidávání nových zásuvek, správu skupin, editaci a zobrazování dat z vybraných zásuvek. Dále obsahuje dialog sloužící k odstranění vybrané zásuvky.



Obr. 3.12: Třída `SocketManagment`

Grafické uživatelské rozhraní



Obr. 3.13: Vzhled GUI třídy `SocketManagment`

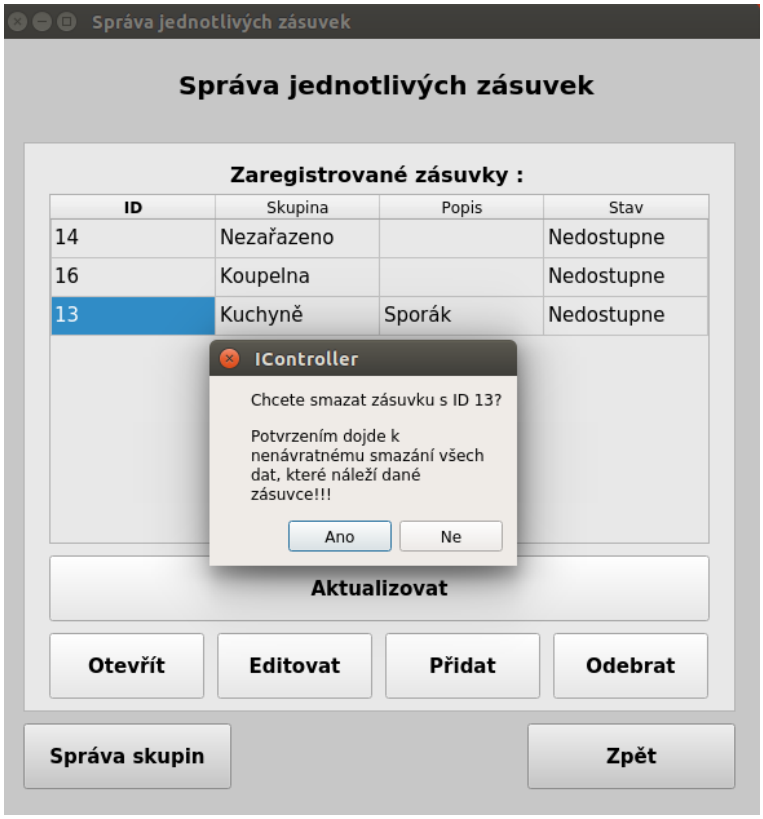
Uživatelské rozhraní je tvořeno rámem `socket_managmnet_FRAME`. Uvnitř něj se nachází tabulka `sockets_TV`, ve které jsou zobrazeny všechny registrované zásuvky, jejich skupiny, popis a stav. Dále tlačítkem `refresh_data_BTN`, pomocí kterého se dají manuálně aktualizovat data v tabulce. Pod tímto tlačítkem jsou tlačítka sloužící ke správě zásuvek:

- `open_socket_BTN` - otevírá widget pro otevření zvolené zásuvky,
- `edit_socket_BTN` - otevírá widget pro editaci zvolené zásuvky,
- `add_socket_BTN` - otevírá widget pro přidání nové zásuvky,
- `delete_socket_BTN` - otevírá dialog pro potvrzení smazání vybrané zásuvky.

Pod rámem se nachází tlačítko `manage_group_BTN`, pomocí kterého se otevírá widget pro správu skupin a tlačítko `exit_BTN`, které slouží k zavření widgetu třídy `SocketManagment`.

Sloty a signály

- `change_socket_data_slot` - slot, který se volá pokud ve třídě `EditSocket` dojde ke změně parametrů zásuvky. Zapříčiní aktualizaci tabulky zásuvek novými daty.
- `add_socket_slot` - slot, který se volá pokud je přidána nová zásuvka. Zapříčiní aktualizaci tabulky zásuvek novými daty.
- `on_delete_socket_BTN_clicked` - slot, který se volá při kliknutí na tlačítko `delete_socket_BTN`. Pokud není zvolena žádná zásuvka z tabulky je zobrazen chybový dialog. Pokud je zvolena existující zásuvka zobrazí se dialog, který slouží k tomu, aby uživatel potvrdil, zda si je jist, že chce danou zásuvku smazat. Pokud tento dialog potvrdí, dojde ke smazání všech záznamů o této zásuvce z databáze. Ukázka tohoto dialogu je vidět na obrázku 3.14.



Obr. 3.14: Vzhled dialogu pro smazání zásuvky

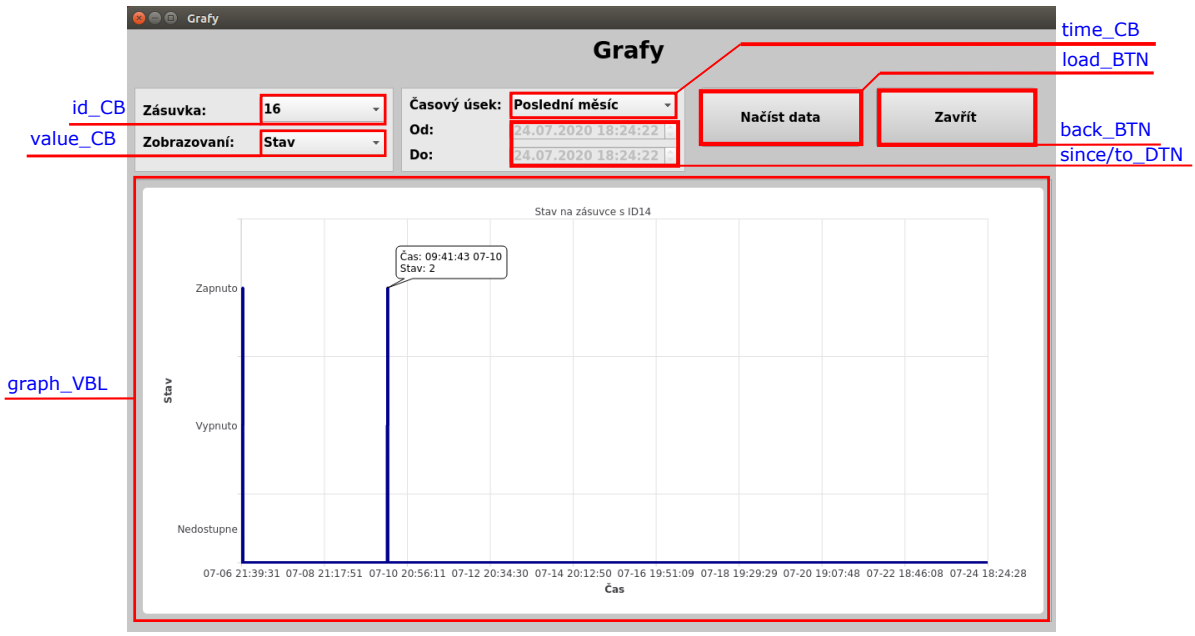
3.5.3 Třída Graphs

Třída `Graphs` slouží k zobrazení zvolených dat ze zásuvek v uživatelem zvoleném čase.

Graphs
Atributy
- view: MyGraphicsView
Metody
- data_processing (id: QString,dataType: QString,interval: QString): QLineSeries - sampling_settings (interval: QString): QPair <int,int>
Sloty
- on_load_BTN_clicked ( ) - on_time_CB_currentTextChanged (text: QString) - on_back_BTN_clicked ( )

Obr. 3.15: Třída `Graphs`

Grafické uživatelské rozhraní



Obr. 3.16: Vzhled GUI třídy `Graphs`

Uživatelské rozhraní je tvořeno boxem `graph_VBL`, ve kterém se zobrazuje graf pro zvolená data. K načtení těchto dat je nutné, aby byla uživatelem zvolena zásuvka, a to s pomocí pole se seznamem `id_CB` a zobrazovaná hodnota s pomocí `value_CB`.

Časový interval, ve kterém se mají data zobrazit, je zvolen pomocí pole se seznamem `time_CB`. Pokud je v tomto poli zvolena možnost „vlastní čas“ dojde k zobrazení datumových polí `since/to_DTE`, kde je možné zvolit vlastní časový interval. Pro načtení dat do grafu, dle zvoleného nastavení, je nutné kliknout na tlačítko `load_BTN`.

Zobrazený graf je možné pomocí šipek na klávesnici libovolně posouvat. Dále také pomocí tlačítka plus a mínus přibližovat a oddalovat. Při najetí kurzorem myši na křivku grafu se zobrazí čas a hodnota v daném bodě.

Metody

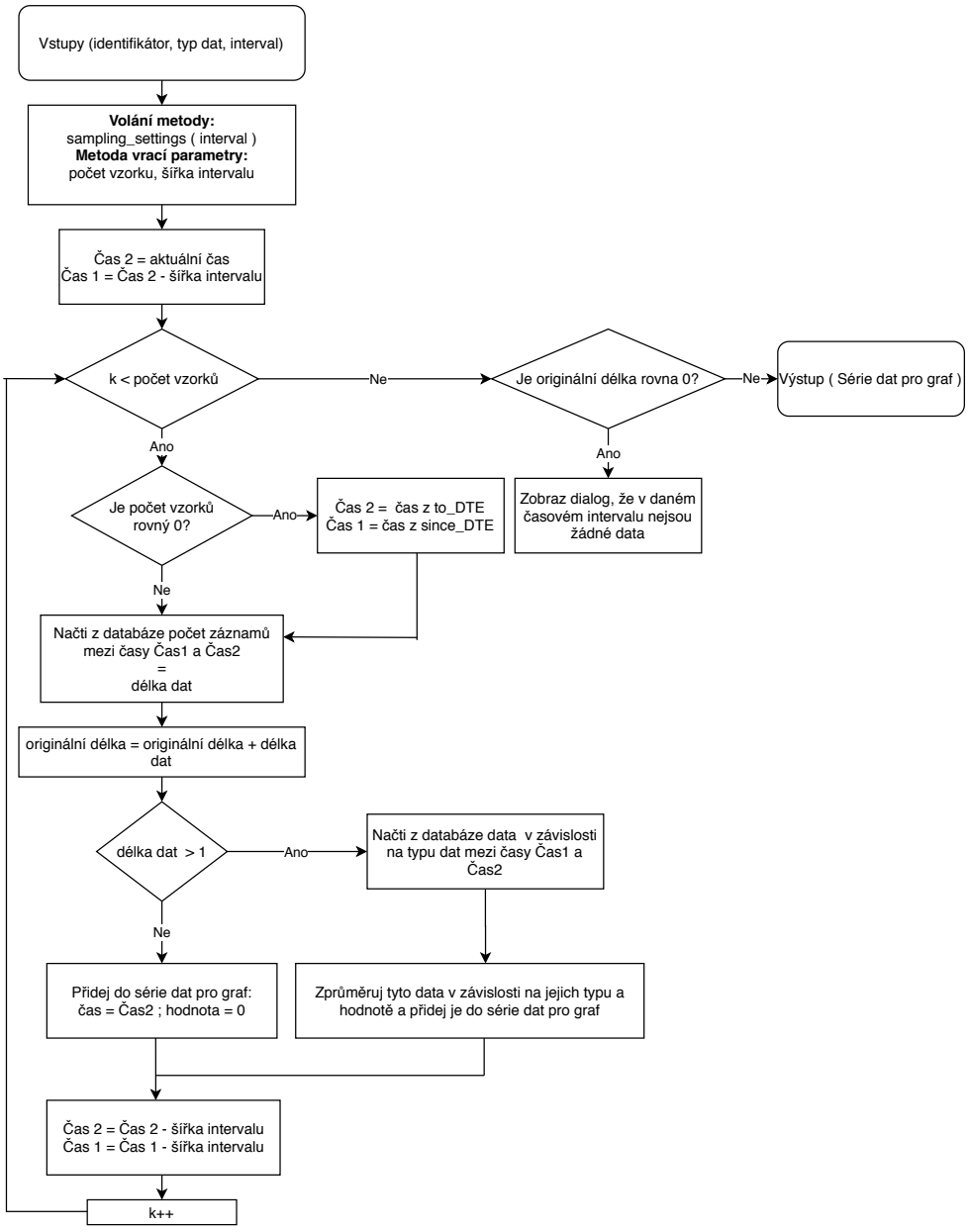
- `sampling_settings` - metoda, která na základě vstupního časového intervalu vrací počet vzorků a šířku intervalu pro metodu `data_processing`. Tyto návratové hodnoty jsou zvoleny tak, aby délka zobrazovaných dat ve výsledném grafu byla téměř neměnná, a tudíž nedocházelo k velkým prodlevám při zobrazování dlouhých časových období. Tabulka vstupu a výstupu této metody je vidět na obrázku 3.1.

Interval	Počet vzorku	Šířka intervalu
Posledních 5 minut	1	360
Posledních 15 minut	1	900
Posledních 30 minut	1	1800
Poslední 1 hodina	1	3600
Poslední 3 hodiny	180	60
Posledních 6 hodin	360	60
Posledních 12 hodin	720	60
Poslední den	1440	60
Poslední týden	1008	600
Poslední měsíc	1008	3600
Poslední čtvrtrok	1008	10800
Poslední půlrok	1008	21600
Poslední rok	1008	43200
Vlastní čas	1	1

Tab. 3.1: Tabulka vstupů a výstupů metody `sample_settings`

- `data_processing` - metoda, která slouží ke zpracování dat z databáze tak, aby je bylo možné přehledně a rychle zobrazit v grafu. Vstupními parametry metody jsou identifikátor, typ dat a časový interval. Na základě těchto vstupních parametrů metoda vrací sérii dat pro graf, které jsou zprůměrovány a upraveny tak, aby nedocházelo k zobrazování nadměrného množství dat a byly viditelně zachovány skokové změny dat, například při změně stavů.
  Podrobněji je při zavolání metody nejprve zavolána metoda `sample_settings`. Ta, na základě zvoleného časového intervalu vrátí počet vzorků a šířku časového intervalu pro průměrování dat. Dále jsou nastaveny časy pro prvotní časový rozsah a volá se cyklus `for`.

V cyklu dochází ke průměrování a úpravě dat v daném časovém rozsahu. Tyto data jsou uložena do série dat pro graf. Pokud v tomto rozsahu nejsou žádná data, je vytvořen umělý nulový vzorek, který je také přidán do série dat pro graf. Také je ukládána originální délka původních dat. Na konci cyklu je celý časový rozsah posunut v čase o šířku intervalu a cyklus se opakuje v závislosti na počtu vzorků. Po skončení cyklu je ověřeno, zda není originální délka dat rovna nule. Pokud tomu tak je, je zobrazen dialog, který upozorní uživatele, že v intervalu nejsou žádná data. Pokud není, vrací metoda sérii dat pro graf. Diagram této metody je možné vidět na obrázku 3.17.



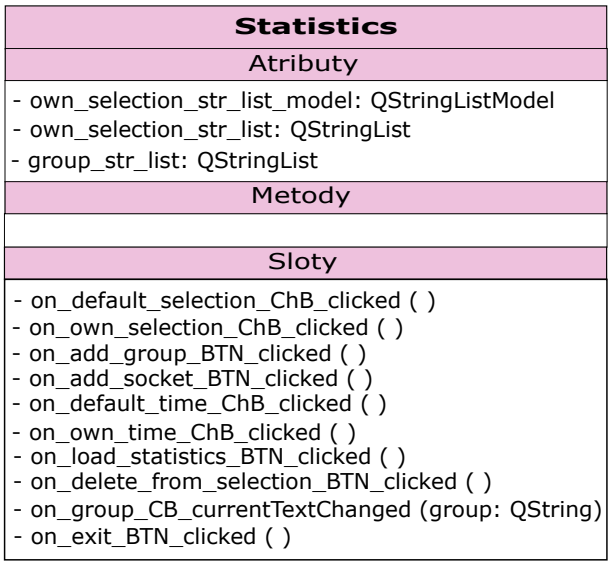
Obr. 3.17: Vývojový diagram `data_processing`

Sloty a signály

- `on_load_BTN_clicked` - slot, který se volá, pokud uživatel klikne na tlačítko `load_BTN`. Při zavolání dojde k uložení nastavení widgetů do proměnných. Toto nastavení je poté předáno metodě `data_processing`, která vrátí sérii dat pro graf, a tato data jsou předány třídě `MyGraphicsView`. Ta se postará o zobrazení příslušného grafu.

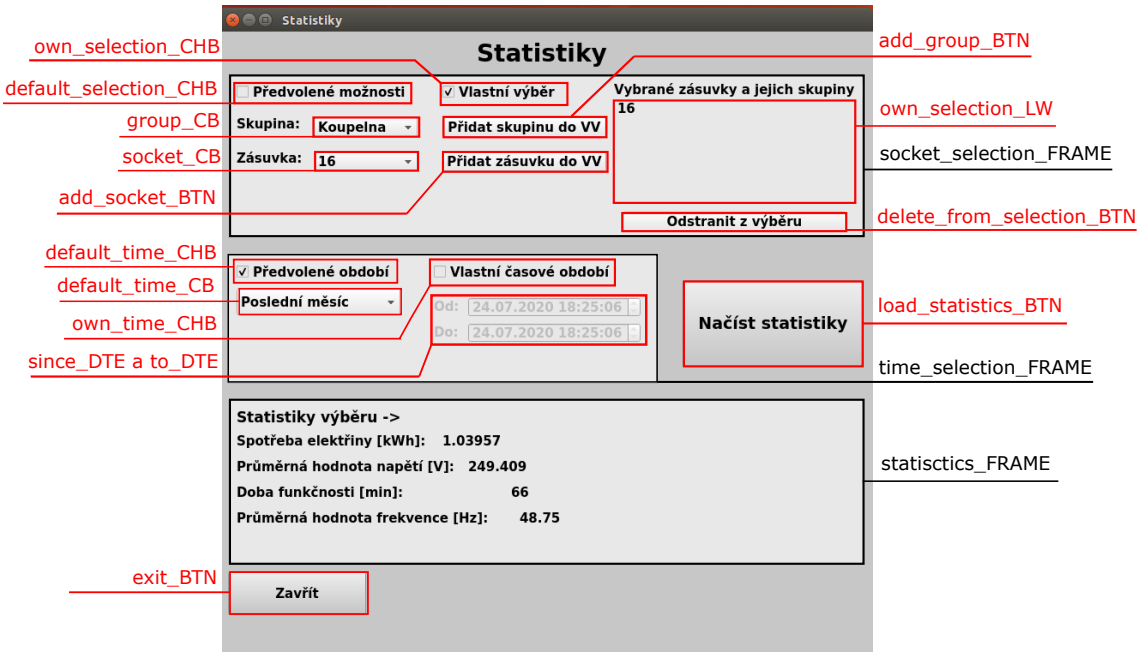
3.5.4 Třída Statistics

Třída `Statistics` slouží k zobrazení statistik zásuvek za zvolený čas. Tyto statistiky lze zobrazit buď v předvoleném nebo vlastním časovém období. Také je možné zvolit, zda se mají zobrazovat statistiky pro předvolený výběr zásuvek a jejich skupin, nebo pro libovolný výběr.



Obr. 3.18: Třída `Statistics`

Grafické uživatelské rozhraní



Obr. 3.19: Vzhled GUI třídy `Statistics`

Uživatelské rozhraní této třídy je rozděleno na tři části. První část tvoří rám `socket_selection_FRAME`. Ten slouží k výběru zásuvek či přímo skupin, ze kterých se mají statistiky zobrazit. Pomocí zaškrtávacích políček `own_selection_CHB` a `own_selection_CHB` lze vybrat, zda se budou volit již předvolené možnosti, nebo uživatel provede vlastní výběr. Při volbě předvolených možností jsou zobrazena pouze seznamová pole `group_CB` a `socket_CB`, ve kterých jsou z databáze nahrány existující zásuvky a skupiny zásuvek. Pokud je zvolen vlastní výběr, je možné pomocí tlačítek `add_group_BTN` a `add_socket_BTN` z již zmíněných seznamových polí přidat zásuvky a skupiny do tabulky `own_selection_LW`, která je zobrazí. Tyto záznamy lze z tabulky mazat pomocí tlačítka `delete_from_selection_BTN`.

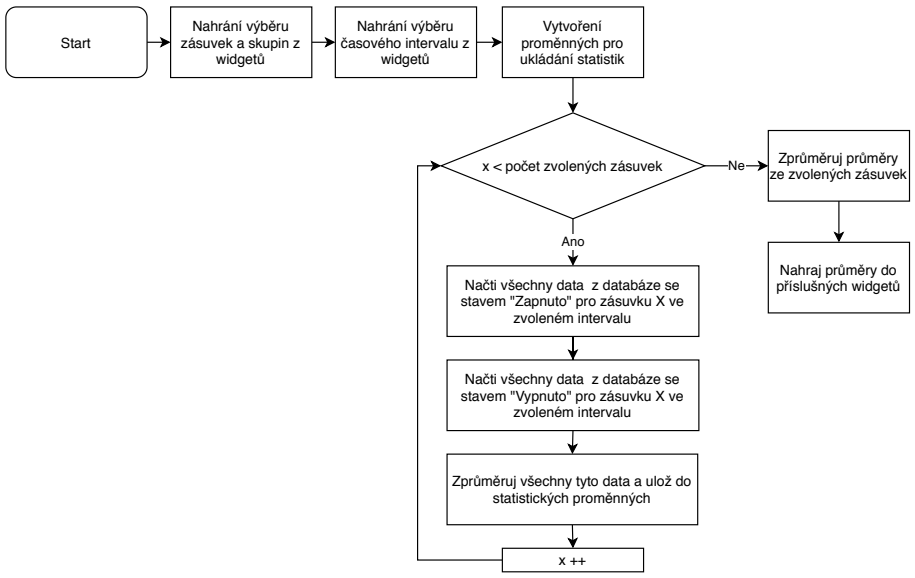
Druhou část tvoří rám `time_selection_FRAME`. Ten slouží k výběru časového intervalu pro zobrazení statistik. Pomocí zaškrtávacích políček `default_time_CHB` a `own_time_CHB` lze vybrat, zda se bude volit z již předvolených časových intervalů, které obsahuje `default_time_CB`. Nebo se zvolí vlastní časový interval pomocí časových vstupů `since_DTE` a `to_DTE`.

Tlačítko `load_statistics_BTN` slouží k nahrání statistik dle výběru uživatele do posledního rámu `statistics_FRAME`, které zobrazuje spotřebu elektrické energie, průměrné napětí a frekvenci výběru. Doba funkčnosti je zobrazena pouze pokud je zvolena jen jediná zásuvka. Pod tímto rámem se nachází tlačítko `exit_BTN`, které slouží k zavření daného rozhraní.

Sloty a signály

- `on_add_group_BTN_clicked` - volá se při stisknutí tlačítka `add_group_BTN`. Vybere právě zvolenou skupinu ze seznamového pole `group_CB` a přidá ji do tabulky vlastních výběrů `own_selection_LW`. Při tomto procesu se kontroluje, zda již tabulka danou skupinu neobsahuje nebo zda se nejedná o skupinu `Všechny`.
- `on_add_socket_BTN_clicked` - volá se při stisknutí tlačítka `add_socket_BTN`. Vezme právě zvolenou zásuvku ze seznamového pole `socket_CB` a přidá ji do tabulky vlastních výběrů `own_selection_LW`. Při tomto procesu se kontroluje, zda již tabulka danou zásuvku neobsahuje.
- `on_load_statistics_BTN_clicked` - slouží k nahrání statistik z vybraných zásuvek a ve zvoleném časovém intervalu do rámu `statistics_FRAME`. Volá se při stisknutí tlačítka `on_load_statistics`.

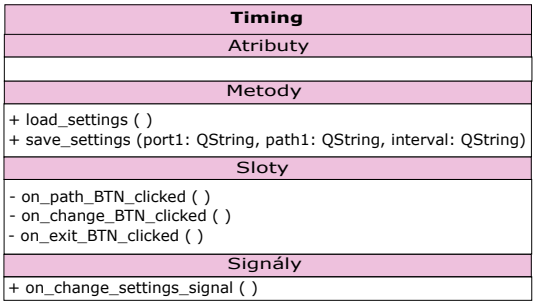
  Při zavolání tohoto slotu jsou nejdříve načteny z widgetů nastavení určená uživatelem. Jmenovitě jsou nahrány vybrané zásuvky, skupiny zásuvek a zvolený časový interval. Poté jsou vytvořeny proměnné pro ukládání zprůměrovaných dat a volán cyklus pro zpracování a průměrování dat. Tento cyklus probíhá pro každou zvolenou zásuvku zvlášť. Nejprve jsou v něm z databáze načteny všechny záznamy v odpovídajícím časovém intervalu, které mají stav „Zapnuto“ nebo „Vypnuto“. Poté jsou tyto záznamy zprůměrovány pomocí aritmetického průměru. Po proběhnutí průměrování pro všechny zvolené zásuvky jsou tyto průměry ještě jednou zprůměrovány a výsledná data zobrazena uživateli. Diagram reprezentující tento algoritmus je vidět na obrázku 3.20.



Obr. 3.20: Vývojový diagram reprezentující zpracování dat pro statistiky

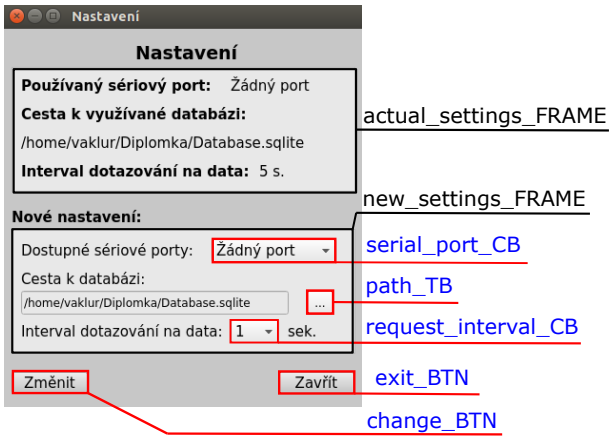
3.5.5 Třída Settings

Třída `Settings` slouží ke změně nastavení aplikace. Konkrétně ke změně používaného sériového portu, databáze a intervalu pro odesílání požadavků do kolektoru dat. Při vytvoření objektu této třídy je načteno stávající nastavení do widgetů zobrazujících aktuální nastavení.



Obr. 3.21: Třída `Settings`

Grafické uživatelské rozhraní



Obr. 3.22: Vzhled GUI třídy `Settings`

Uživatelské rozhraní je tvořeno dvěma rámy. První rám `actual_settings_FRAME` zobrazuje uživateli aktuální nastavení aplikace. Druhý `new_settings_FRAME` slouží ke změně tohoto nastavení, a to konkrétně pomocí pole se seznamem `serial_port_CB`, je možné nastavit jeden z dostupných sériových portů. Tlačítko `path_TB` otevře dialog pro zvolení cesty k nové databázi a pole se seznamem `request_interval_CB` slouží ke zvolení intervalu odesílání požadavků. Pod těmito rámy se nachází dvě tlačítka. Tlačítko `change_BTN`, které slouží ke změně aktuálního nastavení za nové a tlačítko `exit_BTN`, které slouží k zavření uživatelského rozhraní.

Metody

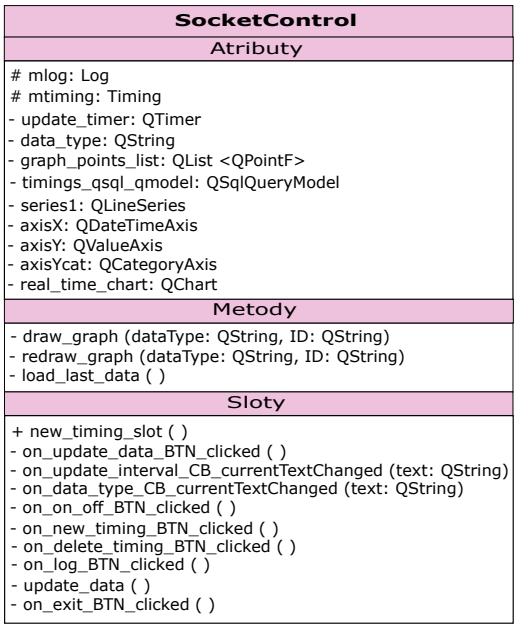
- `load_settings` - metoda, která nahraje do globálních proměnných nastavení přímo z úložiště aplikace.
- `save_settings` - metoda, jejíž vstupní parametry jsou nový sériový port, cesta k databázi a interval odesílání. Metoda tyto parametry uloží do úložiště aplikace.

Sloty a signály

- `on_change_BTN_clicked` - slot, který se volá při stisknutí tlačítka `change_BTN`. Dojde k zobrazení potvrzovacího dialogu, zda chce uživatel změnit stávající nastavení za nové. Pokud tento dialog uživatel potvrdí, je nastavení uloženo pomocí metody `save_settings` a emitován signál `on_change_settings_signal`.
- `on_change_settings_signal` - signál, který je emitován pokud dojde ke změně nastavení aplikace. Slouží k aktualizaci dat ve třídě `MainMenu`.

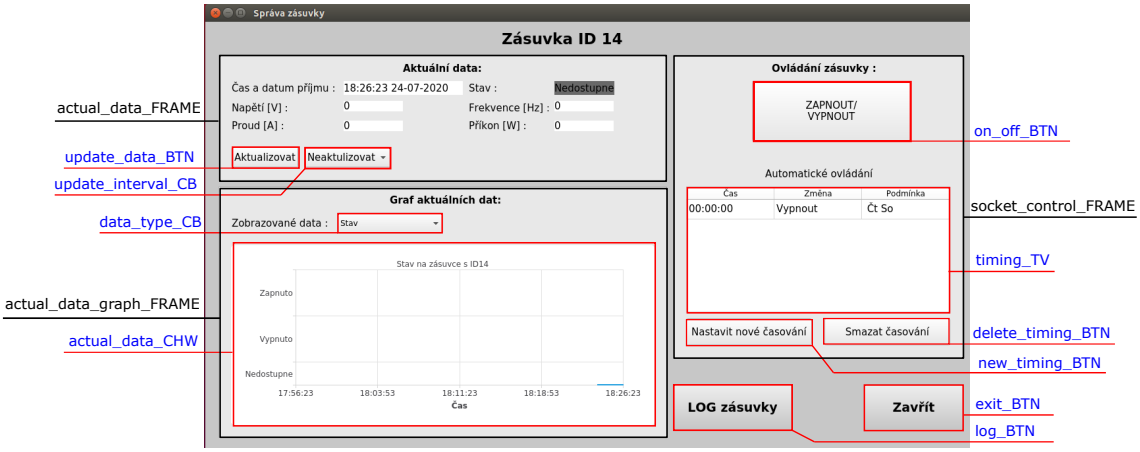
3.5.6 Třída SocketControl

Třída `SocketControl` slouží k zobrazení aktuálních přijímaných dat ze zvolené zásuvky, a to jak textově, tak i graficky. Dále umožňuje manuálně či automaticky změnit stav zásuvky. Při vytvoření objektu této třídy jsou do widgetů načtena data, která byla přijata jako poslední.



Obr. 3.23: Třída `SocketControl`

Grafické uživatelské rozhraní



Obr. 3.24: Vzhled GUI třídy `SocketControl`

Uživatelské rozhraní je rozděleno do tří částí. První část `actual_data_FRAME` zobrazuje textově poslední přijatá data a obsahuje dvě tlačítka. Tlačítko `update_data_BTN` slouží k aktualizaci dat a pomocí pole se seznamem `update_interval_CB` lze navolit časový interval pro automatickou aktualizaci dat.

Druhá část `actual_data_graph_FRAME` se skládá ze seznamového pole `data_type_CB`, které určuje jaké data se zobrazují v grafu `actual_data_CHW`. Tento graf zobrazuje přijatá data za poslední půlhodinu od aktuálního času.

Poslední část `socket_control_FRAME` slouží k ovládání zásuvky, a to buď manuálně pomocí tlačítka `on_off_BTN` nebo lze pomocí tlačítka `new_timing_BTN` přidat příkaz pro automatické ovládání. Tento příkaz je poté zobrazen v tabulce `timing_TV` a lze odstranit pomocí tlačítka `delete_timing_BTN`. Pod touto částí se nachází ještě dvě tlačítka. Tlačítko `log_BTN` slouží k otevření logu se všemi záznamy, které náleží ke zvolené zásuvce a tlačítko `exit_BTN` slouží k zavření uživatelského rozhraní.

Metody

- `draw_graph` - vstupními parametry této metody je identifikátor a typ dat. Slouží ke zpracování a zobrazení dat přijatých za poslední půlhodinu v grafickém widgetu `actual_data_CHW`.
- `load_last_data` - metoda, která slouží k načtení posledních přijatých dat zvolené zásuvky z databáze a jejich zobrazení v rámu aktuálních dat.

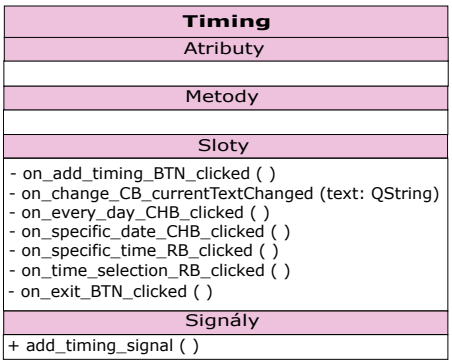
Sloty a signály

- `new_timing_slot` - slot, který se volá, pokud je přidán nový časovač. Zapříčiní aktualizaci tabulky časovačů novými daty.

- `on_on_off_BTN_clicked` - volá se při kliknutí na tlačítko `on_off_BTN`. V závislosti na stavu, který měla zásuvka v posledním přijatém záznamu je do kolektoru dat poslán požadavek na vypnutí nebo zapnutí této zásuvky. Pokud je zásuvka nedostupná, nedochází k žádné akci.
- `on_delete_timing_BTN_clicked` - slot, který je volán při kliknutí na tlačítko `delete_timing_BTN`. Pokud je zvolen některý časovač z tabulky, je zobrazen dialog, zda chce uživatel tento časovač smazat. Potvrzením tohoto dialogu dojde ke smazání časovače z databáze.
- `update_data` - slot, který je volán při vypršení časovače pro aktualizaci dat. Jeho voláním je zapříčiněno překreslení grafu a nahrání nejnovějších dat do widgetu aktuálních dat.

3.5.7 Třída Timing

Třída `Timing` slouží k přidání nového časovače do databáze časovačů. Dovoluje uživateli vybrat čas zapnutí/vypnutí nebo určitý úsek ohraničený dvěma časy. Dále také vybrat možnost opakování v závislosti na určitých dnech nebo datumu.

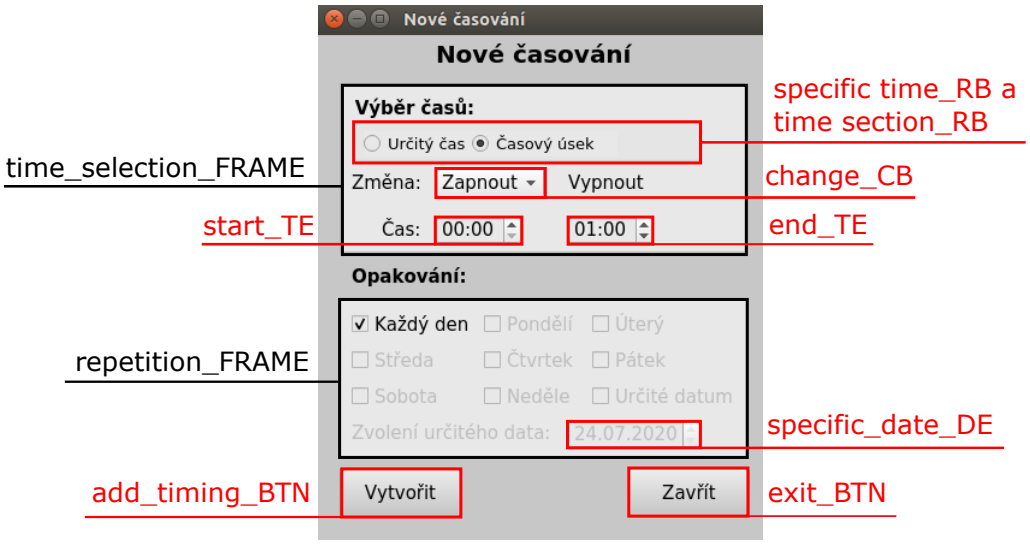


Obr. 3.25: Třída `Timing`

Grafické uživatelské rozhraní

Uživatelské rozhraní je tvořeno dvěma částmi. První, s označením `time_selection_FRAME` slouží k výběru časů, kdy má být změna provedena. S pomocí přepínačů `specific_time_RB` a `time_section_RB` je možné navolit, zda se bude přidávat jen jeden určitý čas nebo časový úsek. Seznamové pole `change_CB` slouží ke zvolení, k jaké změně stavu má dojít a časové vstupy `start_TE` a `end_TE` k volbě časů.

Druhou část tvoří `repetition_FRAME`. Ten slouží k volbě intervalu, v jakém se má časovač opakovat. Je zde možné vybrat buď opakování každý den,v určitý den anebo vlastní datum. Pro volbu vlastního data slouží datový vstup `specific_date_DE`.



Obr. 3.26: Vzhled GUI třídy `Timing`

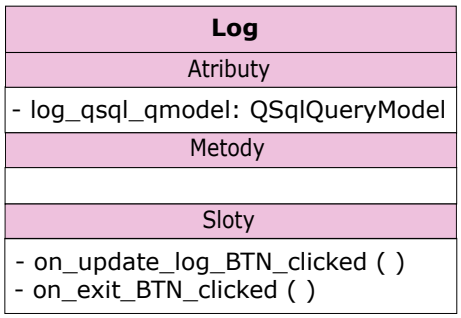
Pod těmito rámy se nachází dvě tlačítka. Tlačítko `add_timing_BTN` slouží k přidání nového časovače v závislosti na nastavených parametrech a tlačíko `exit_BTN` slouží k zavření tohoto rozhraní.

Sloty a signály

- `on_add_timing_BTN_clicked` - slot, který je volán při kliknutí na tlačítko `add_timing_BTN`. Nejprve dojde k prověření, zda není čas pro zapnutí/vypnutí stejný a je zvolena některá z možností opakování. Pokud je vše zadáno v pořádku, je do databáze přidán záznam s novým časovačem.
- `add_timing_signal` - signál, který je emitován, pokud dojde k vytvoření nového časovače. Slouží k aktualizaci tabulky časovačů ve třídě `SocketControl`.

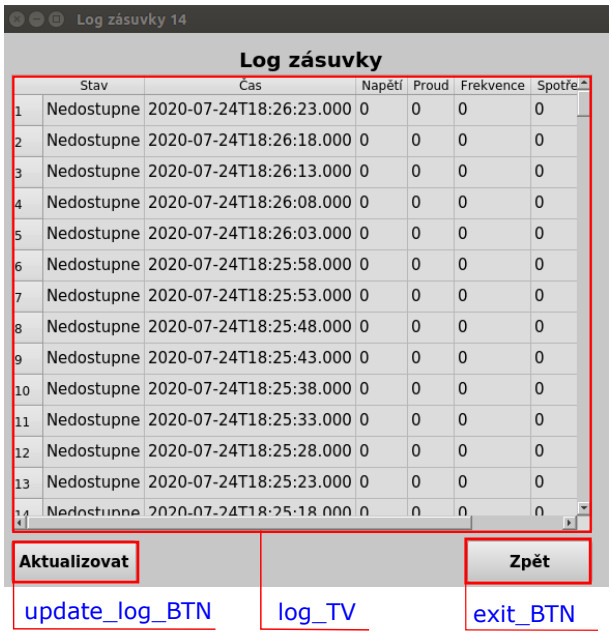
3.5.8 Třída Log

Třída `Log` slouží k zobrazení dat, která jsou uložena v databázi ke zvolené zásuvce.



Obr. 3.27: Třída `Log`

Grafické uživatelské rozhraní

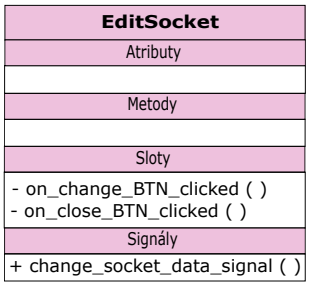


Obr. 3.28: Vzhled GUI třídy `Log`

Uživatelské rozhraní je tvořeno tabulkou `log_TV`, která zobrazuje datové záznamy uložené v databázi pro zvolenou zásuvku. Dále tlačítko `update_log_BTN`, které slouží k aktualizaci dat v tabulce a `exit_BTN`, které slouží k zavření aktuálního rozhraní.

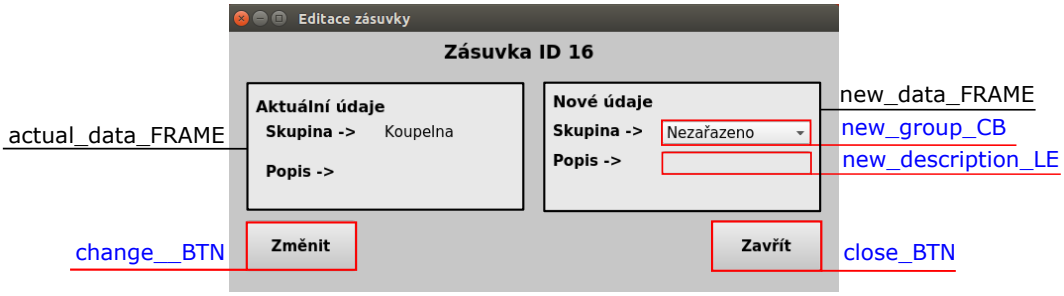
3.5.9 Třída EditSocket

Třída `EditSocket` slouží k editaci uživatelem zvolené zásuvky. Při vytvoření objektu této třídy jsou z databáze načteny aktuální údaje zvolené zásuvky a do widgetu `new_group_CB` nahrány všechny vytvořené skupiny.



Obr. 3.29: Třída `EditSocket`

Grafické uživatelské rozhraní



Obr. 3.30: Vzhled GUI třídy `EditSocket`

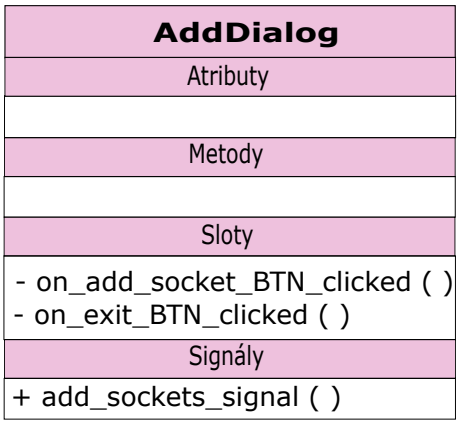
Uživatelské rozhraní je rozděleno na dva rámy. V prvním `actual_data_FRAME` jsou zobrazeny aktuální údaje zvolené zásuvky a ve druhém `new_data_FRAME` jsou widgety pro nové nastavení. Pole se seznamem `new_group_CB` slouží ke zvolení nové skupiny a textové pole `new_description_LE` pro zadání nového popisu. Pod těmito rámy se nachází tlačítko `change_BTN`, které slouží pro potvrzení změny údajů a tlačítko `close_BTN`, které slouží k zavření okna.

Sloty a signály

- `on_change_BTN_clicked` - slot, který je volán při stisknutí tlačítka `change_BTN`. Zapříčiní přepsání aktuálních údajů v databázi za nové, zadané uživatelem.
- `change_socket_data_signal` - signál, který je emitován, pokud dojde ke změně údajů zásuvky. Slouží k aktualizaci dat ve třídě `SocketManagment`.

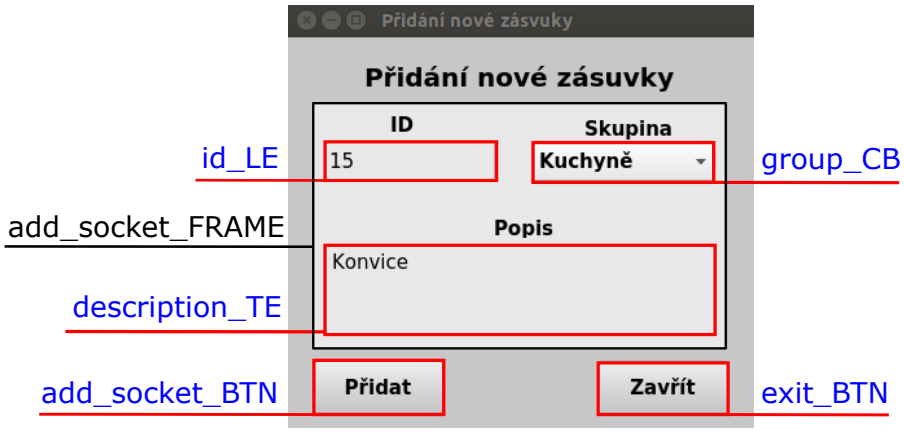
3.5.10 Třída AddDialog

Třída `AddDialog`, slouží k přidání nové zásuvky do databáze zásuvek. Při vytvoření objektu této třídy jsou do widgetu `group_CB` nahrány všechny vytvořené skupiny z databáze.



Obr. 3.31: Třída `AddDialog`

Grafické uživatelské rozhraní



Obr. 3.32: Vzhled GUI třídy `AddDialog`

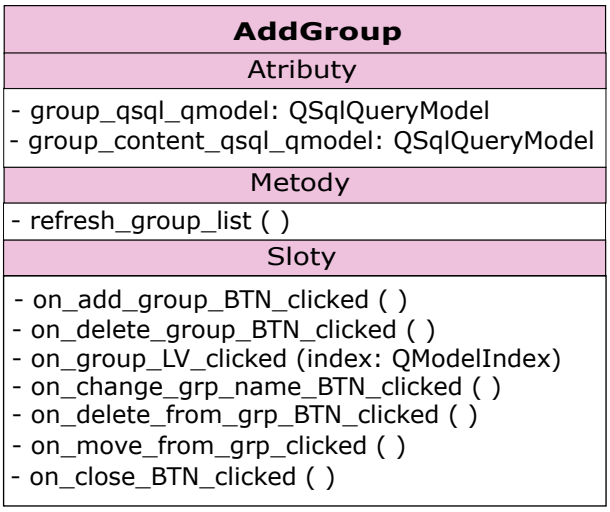
Grafické rozhraní je tvořeno widgety, které slouží pro nastavení parametrů nové zásuvky a dvěma tlačítky. Textové pole `id_LE` slouží k zadání identifikátoru zásuvky. Textové pole `description_TE` slouží k zadání popisu k dané zásuvce a pole s výběrem `group_CB` umožňuje vybrat ze stávajících vytvořených skupin. K přidání nové zásuvky poté slouží tlačítko `add_socket_BTN`. Při kliknutí na tlačítko `exit_BTN` je dané uživatelské rozhraní zavřeno.

Sloty a signály

- `on_add_socket_BTN_clicked` - slot, který je volán při stisknutí tlačítka `add_socket_BTN`. Nejprve dojde k ověření, zda zadaný identifikátor je zadán ve správném rozsahu a zda zásuvka s tímto identifikátorem už není vytvořená. Pokud jsou tyto podmínky splněny, je zásuvka s danými parametry přidána do databáze. Pokud tomu tak není, jsou zobrazeny chybové dialogy.
- `add_sockets_signal` - signál, který je emitován, pokud dojde k bezchybnému přidání nové zásuvky do databáze. Slouží k aktualizaci dat ve třídě `SocketManagment`.

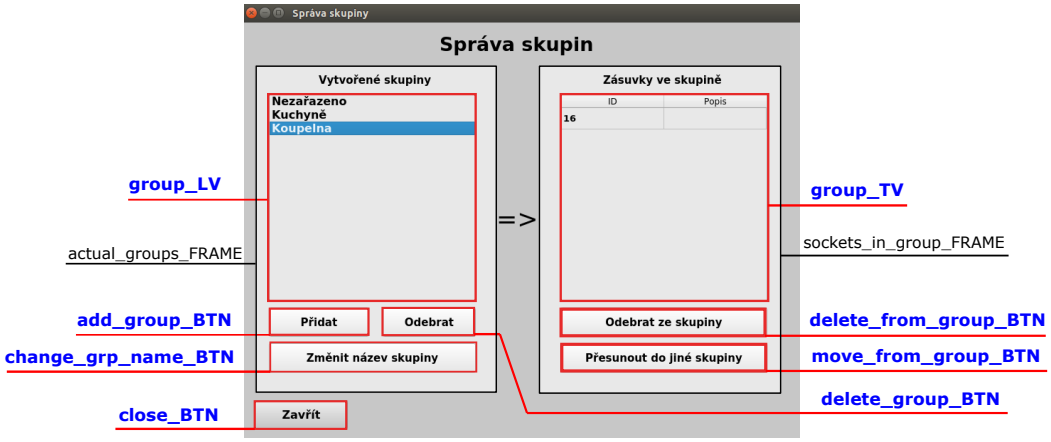
3.5.11 Třída AddGroup

Třída `AddGroup` slouží ke správě skupin. V této třídě je možné skupiny přidávat, mazat a měnit jejich jména. Dále je možné zásuvky přesouvat z jedné skupiny do druhé a mazat zásuvky ze skupiny, ve které se nachází. Při vytvoření objektu této třídy jsou do widgetu `group_LV` nahrány všechny vytvořené skupiny z databáze.



Obr. 3.33: Třída `AddGroup`

Grafické uživatelské rozhraní



Obr. 3.34: Vzhled GUI třídy `AddGroup`

Uživatelské rozhraní je rozděleno na dva rámy. První `actual_groups_FRAME` slouží ke správě skupin, kde již vytvořené skupiny jsou zobrazeny v tabulce `group_LV`. Pod touto tabulkou se nachází tři tlačítka. Pomocí tlačítka `add_group_BTN` lze přidat novou skupinu. Tlačítko `delete_group_BTN` slouží k odstranění vybrané skupiny a při kliknutí na tlačítko `change_grp_name_BTN`, je možné změnit název zvolené skupiny.

Druhý `sockets_in_group_FRAME` slouží ke správě zásuvek ve zvolené skupině. Zásuvky, které skupina obsahuje, jsou zobrazeny v tabulce `group_TV`. Pod touto tabulkou jsou dvě tlačítka. První z nich `delete_from_group_BTN` slouží k odebrání označené zásuvky ze skupiny a druhé `move_from_group_BTN` slouží k přesunutí zvolené zásuvky do jiné z vytvořených skupin.

Metody

- `refresh_group_list` - při volání této metody je aktualizována tabulka `group_LV` novými daty z databáze.

Sloty a signály

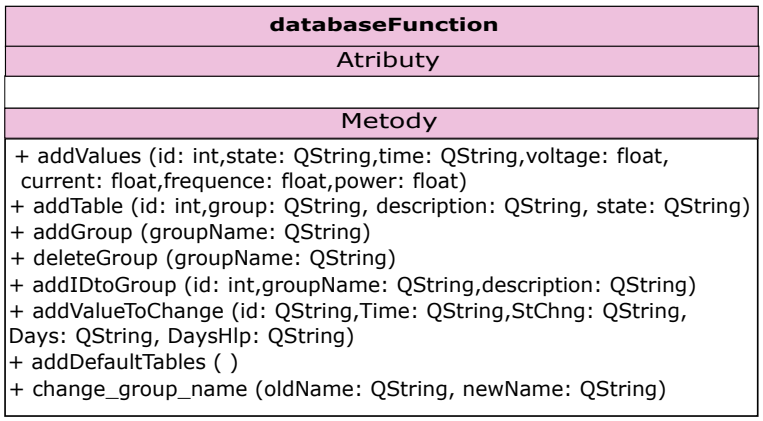
- `on_add_group_BTN_clicked` - volá se při stisknutí tlačítka `add_group_BTN`. Vyvolá dialog pro přidání nové skupiny, který obsahuje textové pole pro zadání názvu nové skupiny a tlačítko pro potvrzení přidání. Při potvrzení dialogu je zkontrolováno, zda je zadaný název skupiny validní a doposud neobsažený v databázi. Pokud je vše v pořádku, dojde k přidání nové skupiny do databáze.

- `on_delete_group_BTN_clicked` - tento slot je volán při stisknutí tlačítka `delete_group_BTN`. Nejprve se zkontroluje, zda je uživatelem vybrána některá ze skupin v tabulce a pokud není, zobrazí se chybová zpráva. Poté dochází k další kontrole, a to zda uživatel nemá označenou skupinu `Nezařazeno`, protože tato skupina nemůže být odstraněna, a také dochází k zobrazení chybové zprávy. Pokud nenastala ani jedna z předchozích možností a je zvolena smazatelná skupina, otevře se dialogové okno. Slouží pro potvrzení smazání uživatelem. Při smazání skupiny nejsou smazány zásuvky v ní obsažené. Tyto zásuvky se pouze přesunou do skupiny `Nezařazeno`.
- `on_group_LV_clicked` - volá se při kliknutí na některý ze záznamů v tabulce skupin `group_LV`. Dochází k nahrání obsahu zvolené skupiny, tedy zásuvek v ní registrovaných do tabulky `group_TV`.
- `on_change_grp_name_BTN_clicked` - slot, který je volán při stisknutí tlačítka `change_grp_name`. Nejprve dojde ke kontrole, zda je zvolena některá ze skupin a není to skupina `Nezařazeno`. Pokud ano, zobrazí se chybová zpráva. Jestliže tato podmínka není splněna, dojde k otevření dialogu s textovým polem, ve kterém může uživatel zadat nové jméno skupiny. Při potvrzení tohoto dialogu se zkontroluje, zda nové jméno neobsahuje nedovolené znaky nebo zda již není v databázi. A pokud je vše v pořádku, dojde ve všech místech databáze k přejmenování skupiny.
- `on_delete_from_group_BTN_clicked` - tento slot je volán při stisknutí tlačítka `delete_from_group`. Pokud je vybrána některá ze zásuvek, dojde k jejímu smazání ze stávající skupiny a přesunutí do skupiny `Nezařazeno`. Pokud se zvolená zásuvka nachází ve skupině `Nezařazeno` , neprovede se žádná akce.
- `on_move_from_group_BTN_clicked` - slot, který se volá, pokud uživatel klikne na tlačítko `move_from_group`. Pokud je zvolena některá ze zásuvek z tabulky, dojde k otevření dialogu, který umožňuje vybrat si některou z dostupných skupin. Potvrzením tohoto dialogu a zvolením cílové skupiny, dojde v databázi k přesunu zásuvky do této skupiny.

3.6 Popis ostatních tříd

3.6.1 Třída databaseFunction

Třída obsahuje metody pro ukládání, úpravu a mazání dat v databázi.



Obr. 3.35: Třída `databaseFunction`

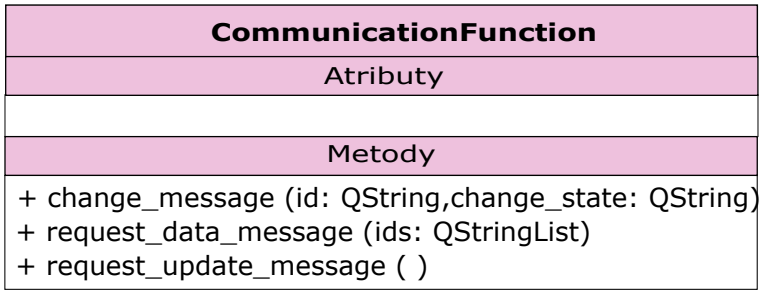
Metody, které třída obsahuje jsou detailně popsány níže.

- `addValues` - metoda, která slouží k přidání nového datového záznamu do databáze. Vstupními parametry je identifikátor zásuvky, ke které se mají data přidat a samotná data přijatá z kolektoru dat.
- `addTable` - slouží k přidání nové zásuvky do databáze. Vstupními parametry je identifikátor, skupina, popis a stav nové zásuvky. Vytvoří se nová tabulka pro ukládání dat této zásuvky. Dále se údaje o nové zásuvce vloží do tabulky `id0` a zásuvka je přiřazena do výchozí skupiny `Nezařazeno`.
- `addGroup` - metoda, pomocí které je možné přidat do databáze záznam o nové skupině. Vstupním parametrem je jméno nové skupiny. Nejdříve se vytvoří tabulka nesoucí jméno skupiny, do které se budou ukládat zásuvky patřící pod tuto skupinu. Dále se skupina přidá do tabulky `grouplist`, která nese informace o všech vytvořených skupinách.
- `deleteGroup` - tato metoda slouží k vymazání skupiny ze všech záznamů v databázi. Vstupním parametrem je jméno skupiny určené k vymazání.
- `addIDtoGroup` - metoda, která slouží k přidání nové zásuvky do zvolené skupiny. Vstupními parametry je identifikátor, skupina a popis zásuvky.
- `addValueToChange` - metoda, která přidá do tabulky `change` v databázi záznam o novém časovači. Vstupními parametry je identifikátor zásuvky, ke které se časovač vztahuje. Dále čas, charakter a den, popřípadě datum, ve kterém má k použití časovače dojít.

- `addDefaultTables` - metoda, která je určená k vytvoření základních tabulek při prvním spuštění aplikace. Mezi základní tabulky patří `grouplist`, `id0`, `change` a `Nezařazeno`.
- `change_group_name` - metoda slouží k přejmenování skupiny v databázi. Vstupními parametry je aktuální a nové jméno zvolené skupiny.

3.6.2 Třída CommunicationProtocol

Třída, která obsahuje metody sloužící k odesílání zpráv do kolektoru dat. Struktura těchto zpráv je dána vytvořeným komunikačním protokolem, který je popsán v kapitole 5.



Obr. 3.36: Třída `CommunicationProtocol`

- `change_message` - metoda slouží k odeslání zprávy typu - žádost o změnu dat. Vstupními parametry této metody je identifikátor zásuvky a typ změny, který má být na dané zásuvce proveden.
- `request_data_message` - metoda, která slouží k odeslání zprávy typu - žádost o data. Vstupním parametrem je seznam identifikátorů zásuvek , ze kterých požadujeme data.
- `request_update_message` - tato metoda slouží k odeslání zprávy - žádost o aktualizaci.

4 Aplikace TrafficGenerator

4.1 Popis aplikace

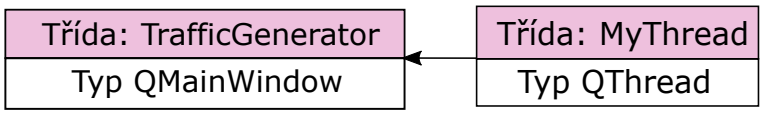
Aplikace TrafficGenerator slouží pro ověření funkčnosti aplikace IController. Reprezentuje a simuluje tzv. kolektor dat, který v reálném provozu vyčítá data z inteligentních zásuvek, udržuje informaci o dostupných zásuvkách a reaguje na požadavky z inteligentního kontroléru.

Pro vývoj aplikace, stejně jako pro aplikaci IController byl zvolen programovací jazyk C++ s knihovnou Qt. Také využívá databázi SQLite pro ukládání dat a sériové rozhraní UART pro komunikaci s inteligentním kontrolérem.

Aplikace umožňuje uživateli přidávat, odstraňovat a měnit stav dostupných zásuvek. Také automaticky reaguje na požadavky přijaté po sériové lince a na základě použitého komunikačního protokolu na ně patřičně odpovídá. Uživatel tuto komunikaci může sledovat v logu událostí, do kterého jsou automaticky zapisovány všechny události, ke kterým došlo v aplikaci. Poslední funkcí aplikace je automatické odesílání dat z dostupných zásuvek nezávisle na přijatých požadavcích.

4.2 Struktura

Aplikace je tvořená pouze dvěma třídami, a to třídou `TrafficGenerator`, která dědí vlastnosti ze třídy `QWidget` a třídou `MyThread`, která dědí vlastnosti ze třídy `QThread`. Struktura aplikace je znázorněna na obrázku 4.1.

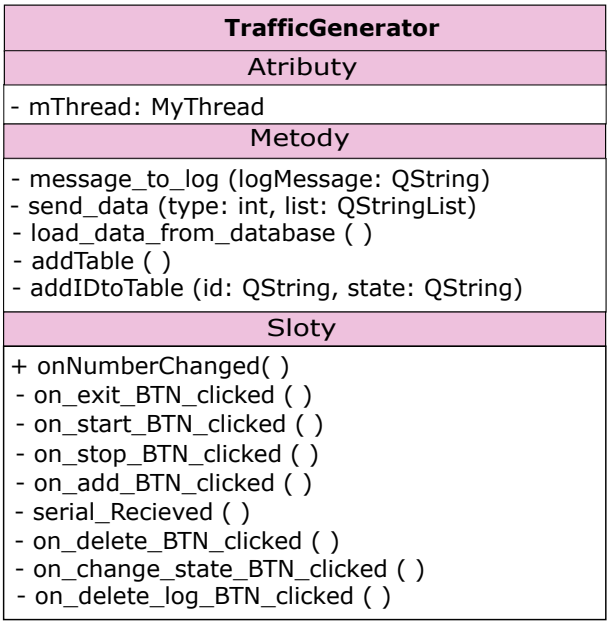


Obr. 4.1: Struktura aplikace TrafficGenerator

4.2.1 Třída TrafficGenerator

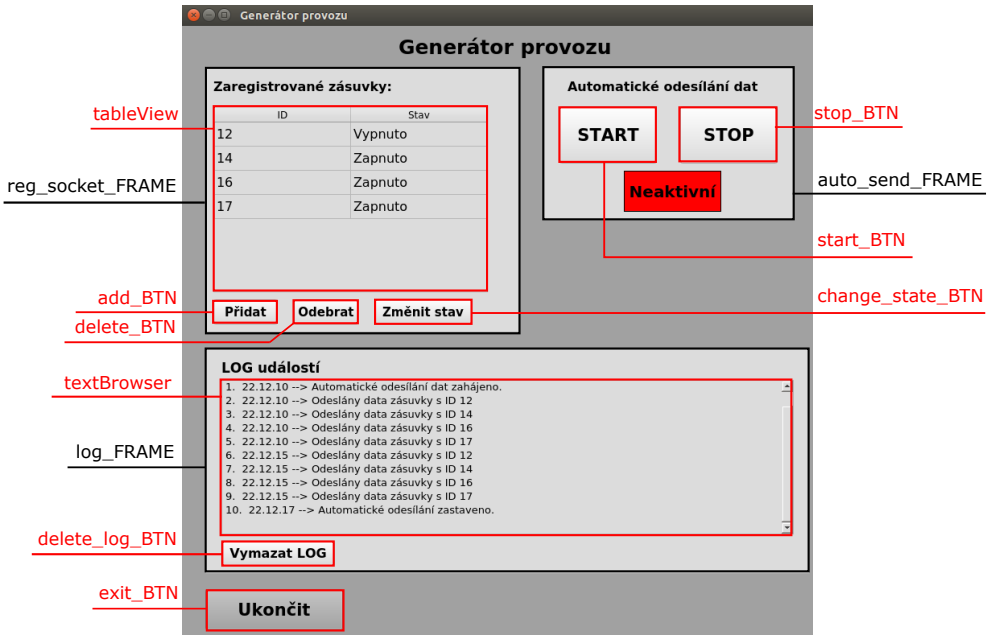
Třída `TrafficGenerator` reprezentuje hlavní třídu aplikace. Po spuštění aplikace je tato třída a její GUI načteno jako první. Slouží k přidávání, mazání a úpravě dostupných zásuvek uložených v databázi. Dále se stará o příjem požadavků ze sériové linky a odpovídání na ně. Umožňuje automatické odesílání datových zpráv a zprostředkovává uživateli log, který informuje uživatele o dění v aplikaci.

Při vytvoření objektu této třídy dochází k inicializování sériového portu a používané databáze. Pokud dojde k tomu, že sériový port není otevřen nebo se nepodařilo spojit s databází, je uživatel upozorněn dialogy na to, že funkčnost aplikace je značně omezena.



Obr. 4.2: Třída `TrafficGenerator`

Grafické uživatelské rozhraní



Obr. 4.3: Vzhled GUI třídy `TrafficGenerator`

Uživatelské rozhraní je rozděleno na tři části. První část `reg_socket_FRAME`, slouží k zobrazení registrovaných zásuvek. Tyto zásuvky jsou zobrazeny v tabulce `tableView` a pod touto tabulkou se nachází tlačítka pro správu těchto zásuvek. Tlačítko `add_BTN` slouží k přidání nové zásuvky, tlačítko `delete_BTN` k odstranění vybrané zásuvky a tlačítko `change_state_BTN` ke změně stavu zvolené zásuvky.

Druhá část `auto_send_FRAME`, slouží ke spouštění a vypínání automatického posílání datových zpráv. Pomocí tlačítka `start_BTN`, je posílání spuštěno a tlačítko `stop_BTN` toto posílání zastaví. Zda je odesílání aktivní, či ne, je zobrazeno indikátorem pod již zmíněnými tlačítky.

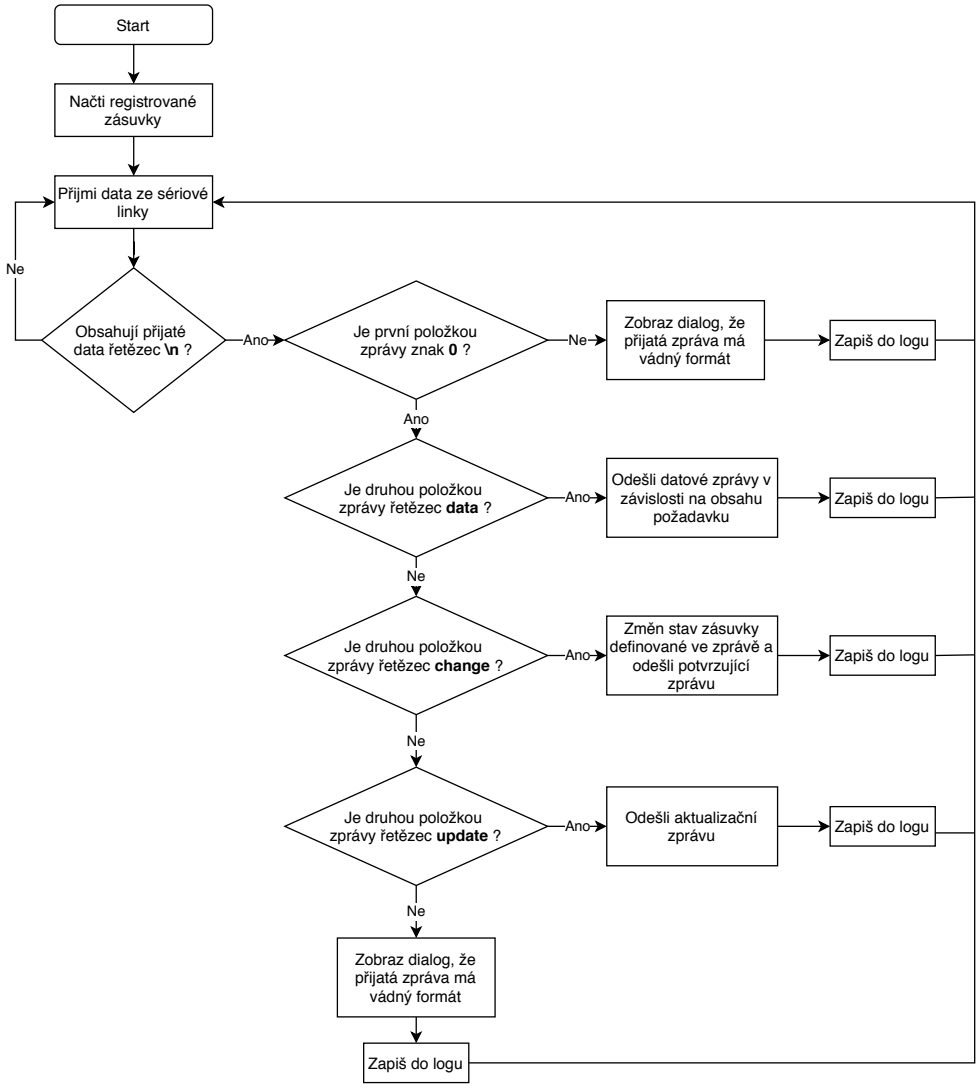
Třetí část `log_FRAME`, slouží k zobrazení logu událostí, které v aplikaci nastaly. Tento log je možné vymazat pomocí tlačítka `delete_log_BTN`. Pod touto částí se nachází tlačítko `exit_BTN` sloužící k ukončení celé aplikace.

Metody

- `send_data` - metoda slouží k odeslání datové zprávy do inteligentního kontroléru. Vstupními parametry jsou typ zprávy a list zásuvek. Typ zprávy určuje, zda bude poslána zpráva s daty nebo zpráva informující o nedostupnosti zásuvky. List zásuvek obsahuje identifikátory zásuvek, pro které mají být zprávy odeslány.
- `load_data_from_database` - slouží k nahrání aktuálních dat z databáze do lokálních proměnných.
- `message_to_log` - přidá do logu, záznam jehož obsah je vstupním parametrem této metody.

Sloty

- `onNumberChanged` - slot, který je volán při přijmutí signálu z vlákna `MyThread`. Načte z databáze všechny zásuvky v ní obsažené a pro tyto zásuvky odešle datové zprávy do inteligentního kontroléru.
- `serial_Recieved` - slot, který je volán při příjmu dat na sériovém portu. Nejdříve je zkontrolováno, zda přijatá data obsahují znak \n, který znamená, že byla přijata celá zpráva. Pokud tomu tak není, pokračuje se v příjmu dalších dat. Pokud ano, dochází ke kontrole, zda první znak přijaté zprávy je nula. Pokud tato podmínka není splněna, vyhodnotí se zpráva jako chybná.
  Dále je kontrolována druhá položka zprávy. Pokud obsahuje řetězec data, je vyhodnoceno, že přišel požadavek na data a v závislosti na dalším jeho obsahu jsou odeslány patřičné zprávy do inteligentního kontroléru. Pokud obsahuje řetězec `change`, je vyhodnoceno, že přišel požadavek na změnu stavu zásuvky definované v další části přijaté zprávy. Dojde ke změně stavu této zásuvky a je odeslána potvrzující zpráva. A pokud obsahuje řetězec `update`, je odeslána aktualizační zpráva. Jestliže nedojde ke splnění žádné z těchto podmínek vyhodnotí se zpráva jako chybná.
  Tento proces zpracování přijatých dat je vidět na obrázku 4.4.



Obr. 4.4: Vývojový diagram slotu `serial_recieved`

4.2.2 Třída MyThread

Třída `MyThread` slouží k automatickému odesílání zpráv do inteligentního kontroléru nezávisle na přijatých požadavcích. Aby při tomto odesílání nedocházelo k pádům či zamrznutí celé aplikace, běží třída v jiném vlákně než zbytek aplikace.

5 Navržený komunikační protokol

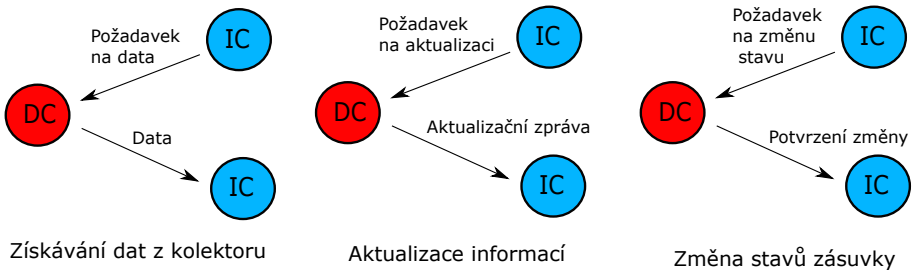
Pro komunikaci mezi aplikací IController a TrafficGenerator byl vytvořen vlastní komunikační protokol. Tento protokol definuje typy a strukturu jednotlivých zpráv, dále popisuje modely komunikace, které mohou nastat. Komunikace s pomocí tohoto protokolu probíhá mezi aplikacemi po sériovém rozhraní UART. Pro správnou funkčnost je nutné, aby parametry sériové linky byly pro obě aplikace stejné. Uživatel se o tento fakt nemusí starat, jelikož parametry sériové linky jsou pevně nastaveny v aplikacích a není možné je měnit.

5.1 Modely komunikace

Modely komunikace popisují jak probíhá komunikace mezi aplikacemi při zaslání různých požadavků. Komunikační protokol definuje tři různé situace, které mohou při této komunikaci nastat:

1. **Získávání dat z kolektoru** - pro získání dat ze zásuvek je nutné poslat z kontroléru do kolektoru zprávu s požadavkem na data. Kolektor se následně pokusí získat data ze zvolené zásuvky. Pokud je získání dat úspěšné, odešle se do kontroléru zpráva s aktuálními daty. Pokud tomu tak není, odešle se zpráva o tom, že tázaná zásuvka je aktuálně nedostupná.
2. **Aktualizace informací** - pro aktualizaci informací o dostupných zásuvkách je nutné odeslat z kontroléru do kolektoru požadavek na aktualizaci. Tento požadavek se odesílá každou minutu nezávisle na frekvenci datových požadavků. Kolektor zkontroluje dostupnost zásuvek k němu přihlášených a odešle do kontroléru aktualizační zprávu.
3. **Změna stavu zásuvky** - pro změnu stavu zvolené zásuvky je z kontroléru odeslán požadavek na změnu stavu. Kolektor se pokusí o změnu stavu zásuvky. Pokud je změna stavu úspěšně provedena, je do kontroléru poslána datová zpráva, která slouží pro potvrzení proběhnutí této akce. Pokud nedojde ke změně stavu, je odeslána zpráva o nedostupnosti dané zásuvky.

Tyto modely jsou graficky znázorněny na obrázku 5.1.



Obr. 5.1: Modely komunikace mezi zařízeními

5.2 Typy a struktura jednotlivých zpráv

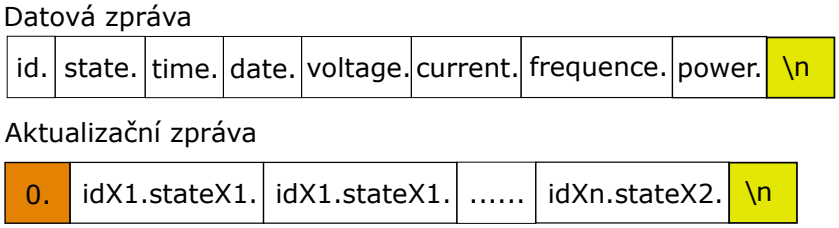
Komunikační protokol definuje pět různých typů zpráv. Zprávy jsou tvořeny řetězcem znaků, který je posílán po sériové lince. Pro rozpoznání těchto zpráv je definováno, že všechny řídící zprávy začínají znakem 0 a datové zprávy libovolnými číselnými znaky. Jednotlivé datové položky, které zpráva obsahuje, jsou odděleny tečkami a konec zprávy je určen řetězcem znaků \n.

5.2.1 Zprávy vysílané kolektorem dat

Kolektor dat vždy vysílá zprávy v reakci na řídící zprávy z inteligentního kontroléru. Mezi tyto zprávy patří:

- **Datová zpráva** - tato zpráva začíná identifikátorem zásuvky, ke které se data v ní obsažená vztahují. Další položky obsahují stav zásuvky, datum a čas získání parametrů zásuvky a následně jednotlivé parametry.
- **Aktualizační zpráva** - tato zpráva je rozpoznána pomocí znaku 0, kterým začíná. Za tímto znakem se nachází položky nesoucí informaci o dostupných zásuvkách. Každá z těchto položek se skládá z identifikátoru zásuvky a jejího aktuálního stavu.

Struktura těchto zpráv je znázorněna na obrázku 5.2.



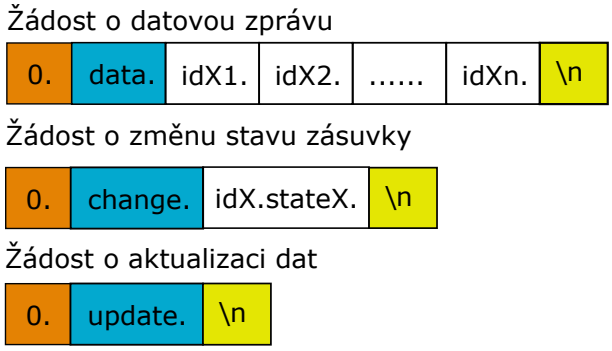
Obr. 5.2: Zprávy vysílané kolektorem dat

5.2.2 Zprávy vysílané inteligentním kontrolérem

Všechny zprávy vysílané inteligentním kontrolérem jsou řídící. Mezi tyto zprávy patří:

- **Žádost o datovou zprávu** - tato zpráva je rozpoznána pomocí klíčového slova data. Za tímto slovem jsou naskládány identifikátory zásuvek, pro která mají být data získána.
- **Žádost o změnu stavu zásuvky** - tato zpráva je rozpoznána pomocí slova change. Za tímto slovem je položka obsahující identifikátor zásuvky u které má dojít ke změně stavu a informaci, o jakou změnu stavu jde.
- **Žádost o aktualizaci dat** - tato zpráva obsahuje pouze klíčové slovo update a slouží k vyžádání aktualizace dostupných zásuvek.

Struktura těchto zpráv je znázorněna na obrázku 5.3.



Obr. 5.3: Zprávy vysílané inteligentním kontrolérem

6 Zhodnocení a testování vytvořené aplikace

6.1 Zhodnocení řešení

Výsledná aplikace realizovaná pomocí knihovny Qt a databáze SQLite, splňuje požadavky, které byly na počátku vývoje stanoveny. Konečné řešení tvoří multiplatformní aplikace, která umožňuje uživateli jednoduché a přehledné ovládání inteligentních zásuvek. Také jejich správu. Aplikaci je možné provozovat na jakékoli platformě, která podporuje Qt a je schopna komunikovat pomocí rozhraní UART.

6.1.1 Výhody

Hlavní výhodou vytvořené aplikace je přehledné a uživatelsky přívětivé GUI, které poskytuje uživateli přehled o všech registrovaných zásuvkách a jejich parametrech. Také poskytuje mnoho možností pro zobrazení vývoje těchto parametrů v čase, a také sledování jejich dlouhodobých průměrů. Nakonec umožňuje jednoduché manuální i automatické ovládání stavu, ve kterém se zásuvky nachází.

Dalšími výhodami je to, že aplikace nepotřebuje přístup k síti, tedy funguje zcela offline, a má malé nároky na výpočetní výkon zařízení, na kterém je provozována.

6.1.2 Nevýhody

Mezi nevýhody lze zařadit to, že aplikace funguje zcela offline, jak již bylo zmíněno výše. Kvůli tomu se je k datům ze zásuvek a jejich nastavení možné dostat pouze v zařízení, na kterém aplikace běží. Jedinou možností jak se k aplikaci dostat jinak, by bylo vzdálené připojení ke grafickému uživatelskému rozhraní zařízení pomocí sítě. Samozřejmě by dané zařízení, na kterém aplikace běží, muselo být v dané síti připojené .

6.1.3 Problematika jednoho řídícího prvku

Výsledná aplikace je navržena pro provozování na jediné centrální řídící stanici.
Z toho plyne, že se v této aplikaci odehrávají veškeré podstatné práce s daty. Jejich příjem, ukládání, zobrazování, a dále odesílání řídících požadavků zásuvkám. Jedná se tedy o centrální řídící systém.

Výhodou využití pouze jedné řídící jednotky tedy je, že veškerá komunikace probíhá jen přes ni. To zajišťuje nemožnost kolize dat a eliminaci konfliktních situací. Naproti tomu velkou nevýhodou je ztráta dat a nemožnost ovládání zásuvek, při její poruše.

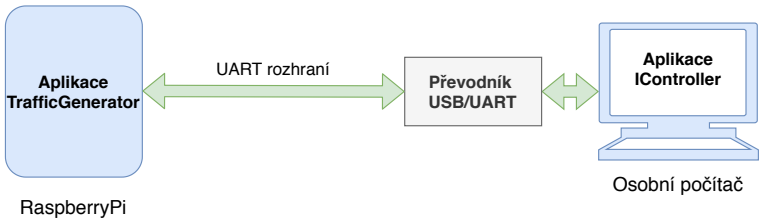
Pro zabránění ztrát dat a možnosti ovládat inteligentní zásuvky by musela aplikace běžet například na dvou paralelně zapojených zařízeních. Jedno z nich by sloužilo jako primární a staralo se o příjem dat a odesílání požadavků. Druhé, sekundární, by pouze naslouchalo a při výpadku by plnohodnotně zastoupilo všechny funkce primárního. Tato zařízení by mezi sebou sdílela databázi s daty, a také by musel být navržen protokol, který by poznal, že došlo k výpadku primárního zařízení.

6.1.4 Zabezpečení

V aplikaci není nijak řešeno zabezpečení dat proti jejich zneužití. Hlavním důvodem, že nebylo implementováno žádně šifrování, je malá možnost zneužití dat. Jelikož celý systém běží offline, a tudíž k získání dat je nutná fyzická přítomnost uživatele u zařízení, na kterém je aplikace provozována. Dále data, tedy parametry zásuvek nemají pro případného útočníka velkou vypovídající hodnotu. Slouží především pro uživatele domu, který systém s touto aplikací využívá.

6.2 Testování funkčnosti

Pro testování funkčnosti aplikace IController byla navržena vlastní aplikace Traffic generator. Při testování byla hlavní aplikace spuštěna na počítači s OS Linux a testovací aplikace běžela na minipočítači Raspberry Pi. Tato zařízení byla mezi sebou propojena pomocí rozhraní UART, které sloužilo pro obousměrnou komunikaci mezi nimi. Zapojení pro testování aplikace je vidět na obrázku 6.1. Při testování byla ověřena funkčnost všech funkcí, které aplikace uživateli poskytuje. Testování probíhalo po dobu pěti hodin a testy nejdůležitějších funkcí aplikace jsou popsány níže.



Obr. 6.1: Zapojení pro testování funkčnosti

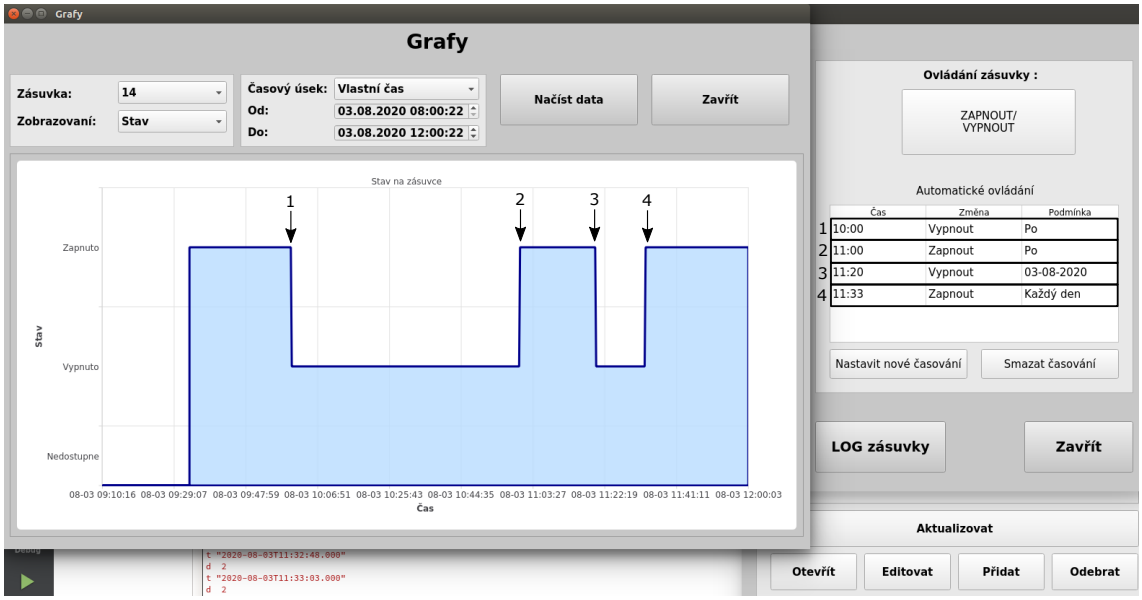
6.2.1 Test automatického ovládání zásuvek

Pro tento test byly pro zásuvku s identifikátorem 14 nastaveny čtyři časovače. Každý z časovačů byl nastaven jinak:

1. Na čas 10:00, změnu stavu na „Vypnout“ a podmínkou pouze v pondělí.
2. Na čas 11:00, změnu stavu na „Zapnout“ a podmínkou pouze v pondělí.
3. Na čas 11:20, změnu stavu na „Vypnout“ a podmínkou pouze dne 3.8.2020.
4. Na čas 11:33, změnu stavu na „Zapnout“ a podmínkou opakovat každý den.

Rozdílné nastavení sloužilo k tomu, aby byly otestovány všechny možné podmínky časovačů, které lze v aplikaci nastavit.

Výsledek tohoto testu lze vidět na obrázku 6.2. V pravé části obrázku je spuštěno GUI třídy `SocketControl` s nastavenými časovači a v levé části obrázku je GUI třídy `Graphs`, kde je ukázaný graf. Tento graf je vykreslen pro časový interval, který bere v potaz všechny nastavené časovače. Z tohoto grafu je patrné, že se časovače aktivovaly přesně v nastavený čas a provedly požadovanou změnu.



Obr. 6.2: Ukázka testu automatického ovládání zásuvek

6.2.2 Test komunikačního protokolu

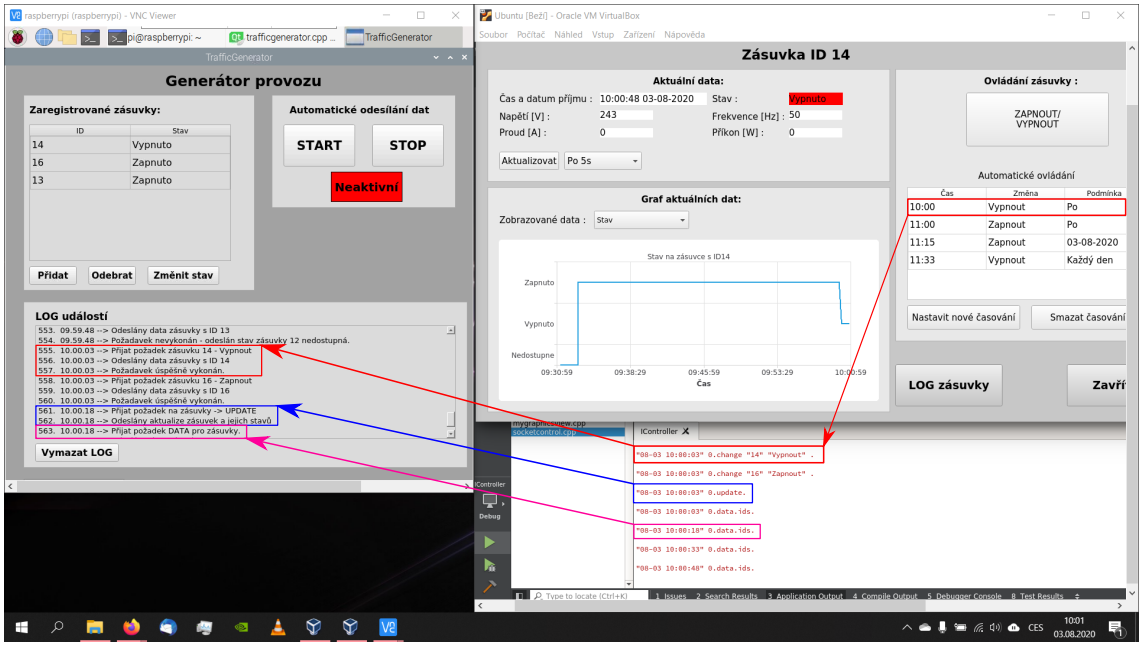
Test komunikace mezi oběma aplikacemi probíhal, jak již bylo zmíněno, přes pět hodin, a za tuto dobu bylo posláno mezi aplikacemi přes šest tisíc zpráv. Za celou dobu testování nedošlo v komunikačním řetězci k žádné chybě.

Ukázku této komunikace je možné vidět na obrázku 6.1. V levém okně je spuštěna aplikace TrafficGenerator na zařízení Raspberry Pi a v pravém okně je spuštěna aplikace IController na počítači. Pod touto aplikací je zobrazen ještě debugovací výstup aplikace v IDE Qt Creator.

Komunikace sloužící ke změně stavu zásuvky je na obrázku zvýrazněná červeně. Nejprve dojde ke spuštění časovače na čas 10:00 s příkazem „Vypnout“, poté je tato zpráva poslána přes sériové rozhraní do kolektoru dat. Ten zprávu příjme, zpracuje a odešle potvrzovací zprávu.

Komunikace sloužící k aktualizaci dostupných zásuvek je zvýrazněna modře. Nejprve dojde k odeslání požadavku u aktualizaci dat přes sériové rozhraní. Kolektor dat tuto zprávu příjme, zkontroluje všechny dostupné zásuvky a odešle aktualizační zprávu. Požadavek o aktualizaci je posílán každou minutu.

Posledním případem je komunikace, sloužící k vyžádání dat z dostupných zásuvek. Tato komunikace je zvýrazněna růžově. Inteligentní ovladač pošle v uživatelem zvoleném intervalu požadavek o data z registrovaných zásuvek. Kolektor dat tento požadavek příjme, vyčte data ze zásuvek a pro dostupné zásuvky odešle jejich parametry a pro ty nedostupné, odešle zprávy o nedostupnosti.



Obr. 6.3: Ukázka testu funkčnosti komunikace mezi aplikacemi

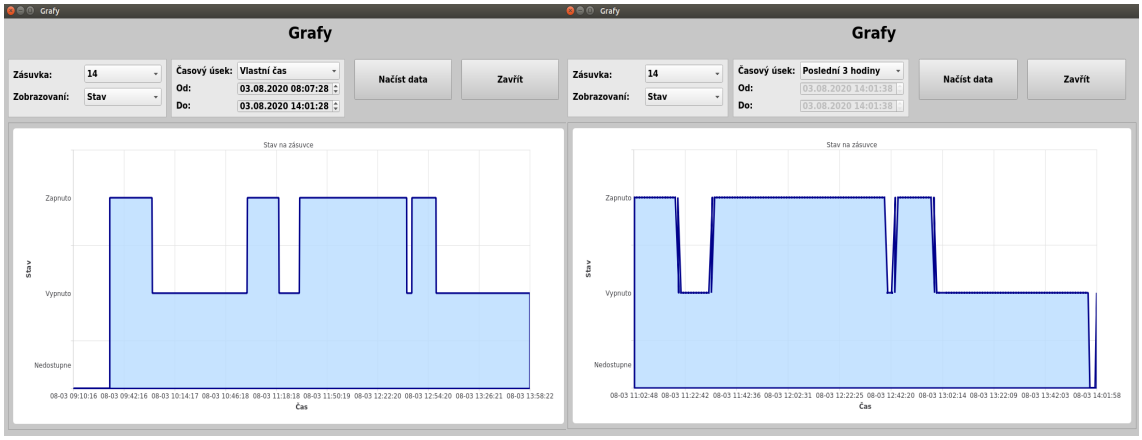
6.2.3 Test algoritmu pro zobrazování grafů

Algoritmus na zpracování dat pro zobrazovaný graf byl otestován na datech vygenerovaných za dobu testu. Grafy vytvořené z dat v tomto intervalu, lze vidět na obrázku 6.4. Z obrázku je patrné, že vykreslené grafy kopírují stejné časy změn stavů dané zásuvky.

Hlavním účelem algoritmu pro zpracování dat je zmenšení objemu dat zobrazovaných v grafu. To tak, aby se při dlouhých časových intervalech nebo velkém množství záznamů za krátký čas vykresloval graf v uživatelsky přijatelném čase. Výsledky komprese jsou zaznamenány v tabulce 6.1. Z této tabulky je patrné, že při nastavení uživatelem zvoleného časového intervalu dochází ke značné kompresi, jelikož algoritmus bere v potaz celý tento interval. Při předem nastavených časových intervalech je využíván odlišný algoritmus, který si daný interval rozdělí na několik menších, ve kterých teprve zpracovává data. V tomto případě dochází zhruba ke třetinovému zmenšení objemu dat. Míra komprese je ovlivněna počtem skokových změn ve zpracovávaných datech.

Výsledky použití algoritmu pro zpracování dat			
Zvolený časový interval	**Původní délka**	**Nová délka**	**Komprese [%]**
Vlastní čas	664	12	98
Poslední 3 hodiny	534	361	32
Posledních 6 hodin	836	562	33
Posledních 12 hodin	1032	705	32

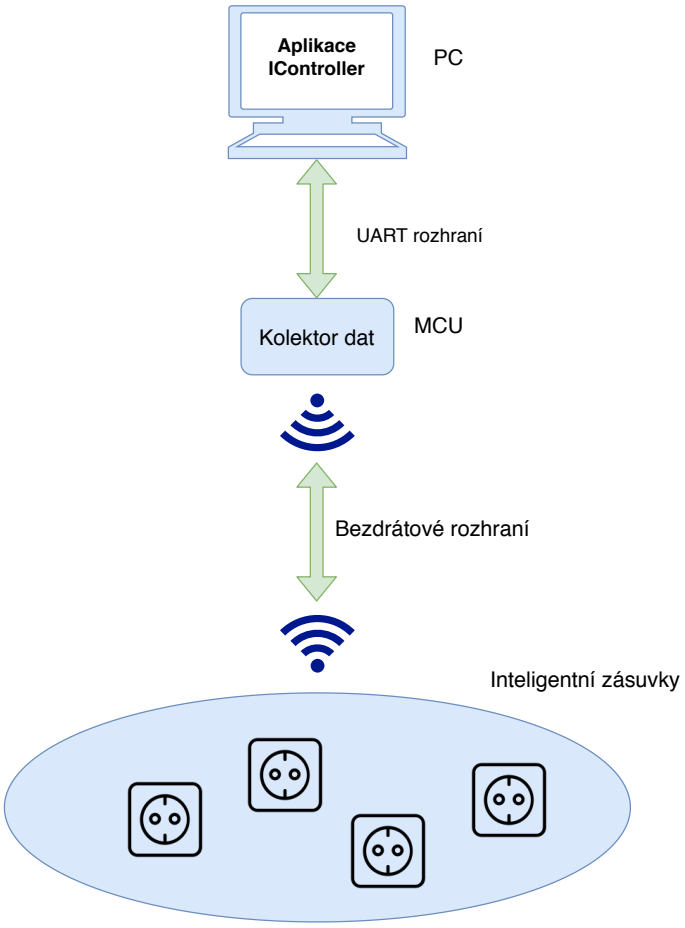
Tab. 6.1: Tabulka účinnosti komprese dat pro rozdílné časové intervaly



Obr. 6.4: Ukázka testu funkčnosti algoritmu pro zpracování dat

6.3 Možnosti reálného použití

Vzhledem k využití knihovny Qt, která je multiplatformní, lze aplikaci spustit na velkém množství zařízení. K tomu, aby mohla správně fungovat, potřebuje, aby zařízení, na kterém běží, disponovalo rozhraním UART. K tomuto rozhraní lze poté připojit jakékoliv zařízení sloužící jako kolektor dat z inteligentních zásuvek a využívající navržený komunikační protokol. Zapojení v reálném provozu by tedy obsahovalo zařízení s vytvořenou aplikací spojené přes UART s kolektorem dat. Kolektor dat by bezdrátově komunikoval se zásuvkami, například s pomocí některého ze standardů IEEE 802.11 nebo IEEE 802.15. Takovéto zapojení je vidět na obrázku 6.5.



Obr. 6.5: Příklad možnosti reálného zapojení

6.4 Možnosti dalšího rozvoje aplikace

Důležitým krokem pro další vývoj aplikace, by bylo vytvoření webového rozhraní, pomocí kterého by bylo možné ovládat aplikaci vzdáleně. Také by dané rozhraní umožňovalo data uložené v databázi, patřící aplikaci, zobrazovat a přehledně zpracovávat. Také jednou z možností by bylo to, že databáze přímo v zařízení, na kterém aplikace běží, by sloužila pouze k zálohování dat. Například při výpadku připojení. Hlavním místem pro ukládání dat by byla některá ze serverových databází.

Dalším důležitým krokem by bylo předělání všech vytvořených GUI do moderního uživatelského vzhledu, který by umožnil ještě větší uživatelský komfort.

V poslední řadě by mohla být aplikace rozšířená na ovládání a správu více typů zařízení, než jen inteligentních zásuvek. Například inteligentních žárovek nebo teplotních čidel.

Závěr

Cílem této práce bylo vytvoření aplikace pro ovládání a správu inteligentních zásuvek v domácnosti. Práce byla rozdělena do dvou částí.

Teoretická část popisovala technologie, které byly použity při tvorbě aplikace. První kapitola popsala aplikační platformu Qt a její hlavní vlastnosti. Dále moduly, které využívá a nástroje, které slouží ke tvorbě aplikací využívající tuto platformu. Druhá kapitola se zabývala relačním databázovým systémem SQLite, zejména byly popsány jeho nejdůležitější vlastnosti, výhody, nevýhody a praktické využití.

Praktická část nejprve popisuje vlastnosti a strukturu navržené aplikace. Poté jsou popsány jednotlivé tabulky databáze SQLite, které aplikace využívá pro ukládání a třídění přijatých dat. Dále jsou popsány jednotlivé třídy, ze kterých se aplikace skládá. U každé z těchto tříd jsou popsány nejdůležitější metody, signály a sloty. U tříd, které disponují grafickým uživatelským rozhraním je dané rozhraní podrobně popsáno a také jsou vysvětleny funkce jednotlivých widgetů, ze kterých se dané rozhraní skládá.

V druhé kapitole praktické části je popsána aplikace vytvořená pro testování hlavní aplikace. Podrobně jsou popsány její funkce, vzhled a třídy, které využívá.

Kapitoly zabývající se klíčovými částmi aplikace byly doplněny vývojovými diagramy důležitých metod, obrázky ilustrující grafická uživatelská rozhraní a strukturu tříd.

Ve třetí kapitole je popsán navržený komunikační protokol pro komunikaci s vytvořenou aplikací. Nejprve jsou představeny modelové situace, ke kterým může při komunikaci dojít. Dále jsou uvedeny a popsány typy zpráv, které komunikační protokol využívá pro komunikaci z a do aplikace.

Závěrečná kapitola se věnuje zhodnocení realizovaného řešení a jeho testování pomocí testovací aplikace. Dále jsou navrhnuta možná vylepšení, či rozšíření finální aplikace do budoucna.

Literatura

[1] A Brief History of Qt. In: *Grafická uživatelská prostředí a knihovny* [online]. 2014 [cit. 2020-08-04]. Dostupné z: https://rtime.felk.cvut.cz/osp/prednasky/gui/the-qt-story/

[2] Qt (Knihovna). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-08-04]. Dostupné z: https://cs.wikipedia.org/wiki/Qt_(knihovna)

[3] Knihovna Qt. In: *Is.mendelu.cz* [online]. [cit. 2020-08-04]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=32514

[4] THELIN, Johan. *Utiliser CMake pour compiler des projets Qt* [online]. In: . 2012 [cit. 2020-08-04]. Dostupné z: https://qt-quarterly.developpez.com/qq-34/cmake/

[5] Application Development. *Qt Documentation* [online]. Finland: The Qt Company, 2020 [cit. 2020-08-04]. Dostupné z: https://www.qt.io/licensing/

[6] Knihovna Qt. In: *Hobrasoft* [online]. Praha: Hobrasoft, 2016, 4.4.2016 [cit. 2020-08-04]. Dostupné z: https://www.hobrasoft.cz/cs/blog/bravenec/qt

[7] Signals and Slots. *Qt Documentation* [online]. Finland: The Qt Company, 2020 [cit. 2020-08-04]. Dostupné z: https://doc.qt.io/qt-5/signalsandslots.html

[8] Qt Creator. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-08-04]. Dostupné z: https://en.wikipedia.org/wiki/Qt_Creator

[9] Qt Designer Manual. *Qt Documentation* [online]. Finland: The Qt Company, 2020 [cit. 2020-08-04]. Dostupné z: https://doc.qt.io/qt-5/qtdesigner-manual.html

[10] KUMST, Martin. Seznámení s SQLite. In: *Root* [online]. Praha: Internet Info, 2016, 6. 4. 2016 [cit. 2020-08-04]. Dostupné z: https://blog.root.cz/maertienuv-obcasny-blog/seznameni-s-sqlite/

[11] MARTINEK, Michal. Lekce 1 - Úvod do SQLite a příprava prostředí. *Itnetwork.cz* [online]. 2019 [cit. 2020-08-04]. Dostupné z: https://www.itnetwork.cz/sqlite/sqlite-tutorial-uvod-a-priprava-prostredi

[12] DB Browser for SQLite: The Official home of the DB Browser for SQLite. *DB Browser for SQLite* [online]. [cit. 2020-08-04]. Dostupné z: https://sqlitebrowser.org/

Seznam symbolů, veličin a zkratek

GPL	obecná veřejná licence
LGPL	menší obecná veřejná licence
GUI	grafické uživatelské rozhraní
GNU	počítačový svobodný operační systém
GDB	GNU debugger
CDB	konzolový debugger
SDK	soubor nástrojů pro vývoj software
UART	univerzální asynchronní přijímač-vysílač
XML	rozšiřitelný značkovací jazyk
QML	Qt modelovací jazyk
MVC	model/view/kontroler návrhové schéma
SCM	správa zdrojového kódu
IDE	vývojové prostředí
RDBMS	systém řízení báze dat
CSV	souborový formát určený pro výměnu tabulkových dat
OS	operační systém
HTML	hypertextový značkovací jazyk
SQL	standardizovaný strukturovaný dotazovací jazyk

Seznam příloh

A Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `Databáze` ..... Databázové soubory pro obě vytvořené aplikace
    - `IControllerDatabase.sqlite`
    - `TrafficGeneratorDatabase.sqlite`
  - `Zdrojové kódy` ..... Zdrojové kódy obou vytvořených aplikací
    - `IController` ..... Složka se zdrojovými kódy aplikace IController
    - `TrafficGenerator` ..... Složka se zdrojovými kódy aplikace TrafficFenerator
    - `Spuštění_aplikací.txt` ..... Návod na spuštění aplikace
  - `Diplomová_práce_Michálek.pdf` ..... Samotná diplomová práce ve formátu PDF