VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

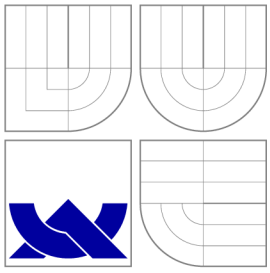
APLIKACE PRO OVLÁDÁNÍ INTELIGENTNÍ ELEKTROINSTALACE INELS PRO PLATFORMU IOS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

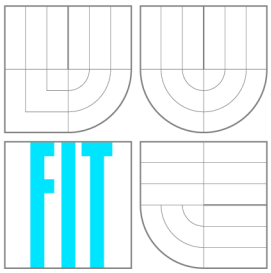
AUTOR PRÁCE
AUTHOR

Bc. LUKÁŠ MATAJ

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE PRO OVLÁDÁNÍ INTELIGENTNÍ ELEKTRO-INSTALACE INELS PRO PLATFORMU IOS

APPLICATION FOR CONTROL OF INTELLIGENT ELECTROINSTALLATION INELS FOR THE PLATFORM IOS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. LUKÁŠ MATAJ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. MARTIN DRAHANSKÝ

BRNO 2013

Abstrakt

Tato diplomová práce má za cíl seznámení se s vývojem mobilní aplikace, která dokáže ovládat inteligentní elektroinstalaci iNELS a její multimediální nadstavbu iMM. Součástí práce je také popis samotného vývoje pro platformu iOS a prozkoumání možností komunikace s oběma ovládanými systémy. Popsány tak budou dva důležité komunikační protokoly – EPSNET a XML-RPC. Výsledný produkt je na závěr zhodnocen a také je popsáno jeho chování v ostrém provozu.

Abstract

This masther's thesis aims at familiarization with the development of mobile application that can control intelligent electrical installation iNELS and its multimedia superstructure iMM. The thesis also describes the common development for the iOS platform and explore the possibility of communication with both controlled systems. Two important communication protocols will be described – EPSNET and XML-RPC. The resulting product is evaluated at the end and its behavior in full operation is described too.

Klíčová slova

Platforma iOS, iOS aplikace, Objective-C, inteligentní elektroinstalace, iNELS, protokol EPSNET, XML-RPC protokol

Keywords

iOS platform, iOS application, Objective-C, intelligent electroinstallation, iNELS, EPSNET protocol, XML-RPC protocol

Citace

Lukáš Mataj: Aplikace pro ovládání inteligentní elektroinstalace iNELS pro platformu iOS, diplomová práce, Brno, FIT VUT v Brně, 2013

Aplikace pro ovládání inteligentní elektroinstalace iNELS pro platformu iOS

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Ing., Dipl.-Ing. Martina Drahanského, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Mataj
27. května 2013

Poděkování

Děkuji panu docentu Martinu Drahanskému za vedení při vytváření této zprávy. Dále děkuji firmě ELKO EP, s.r.o. a jejímu řediteli Jiřímu Konečnému za poskytnuté prostředky a materiály potřebné pro realizaci.

Obsah

1 Úvod **3**

2 Vývoj aplikací pro systém iOS **4**
2.1 Operační systém iOS . . . 4
2.1.1 Architektura systému . . . 4
2.2 Integrované vývojové prostředí XCode . . . 6
2.3 Programovací jazyk Objective-C . . . 7
2.3.1 Třídy a objekty . . . 7
2.3.2 Metody . . . 8
2.3.3 Vlastnosti třídy . . . 8
2.3.4 Práce s pamětí . . . 9
2.3.5 Moderní Objective-C . . . 9
2.3.6 Bloky . . . 10
2.3.7 Grand Central Dispatch . . . 11
2.4 Systémové frameworky . . . 11
2.5 Návrhové vzory . . . 12
2.5.1 Model-View-Controller (MVC) . . . 12
2.5.2 Delegace . . . 12
2.5.3 Target-Action . . . 13
2.6 Základní struktura aplikace . . . 13
2.7 Distribuce aplikace . . . 15

3 Inteligentní elektroinstalace iNELS **16**
3.1 Popis systému iNELS . . . 16
3.1.1 Sběrnice CIB . . . 16
3.1.2 Centrální jednotka . . . 17
3.1.3 Komunikace s centrální jednotkou . . . 17
3.2 Síť EPSNET . . . 18
3.2.1 Obecná struktura protokolu EPSNET . . . 19
3.2.2 Typy odpovědí . . . 20
3.2.3 Komunikační služby . . . 20
3.3 Identifikace zařízení v systému iNELS . . . 22
3.4 Multimediální systém iMM . . . 23
3.4.1 Komunikační protokol XML-RPC . . . 23
3.5 Vizualizace systému iNELS . . . 25

4 Návrh aplikace **26**
4.1 Základní struktura projektu . . . 26
4.2 Klientská aplikace . . . 27
4.3 Grafické uživatelské rozhraní . . . 27
4.4 Síťová komunikace . . . 28
4.5 Zobrazování zařízení . . . 29

5 Implementace **30**
5.1 Síťová komunikace . . . 30
5.1.1 Implementace EPSNET protokolu . . . 30
5.1.2 Implementace XML-RPC protokolu . . . 31
5.2 Třída AppDelegate . . . 32
5.3 Hlavní okno aplikace . . . 32
5.3.1 Konfigurace pokojů a zařízení . . . 32
5.3.2 Základní prvky hlavního okna . . . 33
5.3.3 Zobrazování dlaždic . . . 34
5.4 Rychlé menu . . . 35
5.4.1 Základní prvky okna . . . 35
5.4.2 Zobrazení zařízení . . . 36
5.4.3 Ovládání bezpečnostních kamer . . . 36
5.4.4 Ovládání klimatizací . . . 37
5.4.5 Ovládání Miele spotřebičů . . . 38
5.4.6 Zkratky pro zobrazení sekcí . . . 38
5.5 Multimédia . . . 38
5.5.1 Konfigurace zón . . . 38
5.5.2 Základní prvky okna . . . 39
5.5.3 Zobrazování adresářové struktury . . . 39
5.5.4 Playlist a práce s multimédii . . . 39
5.6 Měření energií . . . 40
5.6.1 Základní prvky okna . . . 40
5.6.2 Zobrazování dat . . . 41
5.6.3 Grafy . . . 41
5.7 Nastavení aplikace . . . 41
5.7.1 Stažení konfigurace ze serveru . . . 41
5.8 Testování aplikace . . . 42
5.9 Distribuce aplikace . . . 42
5.9.1 Promo aplikace . . . 43

6 Popis aplikace a ovládání **44**

7 Závěr **48**
7.1 Možný rozvoj do budoucna . . . 49

A Obsah DVD **52**

B Snímky obrazovek aplikace **53**

Kapitola 1

Úvod

V posledních letech můžeme stále více pozorovat trend ve zvyšování komfortu v obytných budovách. Domy se vybavují kvalitním nábytkem, moderními kuchyněmi a dalším vybavením, ale také například inteligentní elektrickou instalací. Pomocí inteligentní elektroinstalace je možné efektivně spravovat celý dům od ovládání světel, přes ovládání různých spotřebičů až po multimediální funkce. Firmy, které tyto systémy vytvářejí, se samozřejmě snaží uživatelům také nabízet co nejjednodušší a nejpohodlnější ovládání funkcí těchto systémů. V dnešní době, kdy jsou nesmírně populární chytré telefony a podobná mobilní zařízení, se přímo nabízí použít pro ovládání inteligentní elektroinstalace právě je. A přesně vytvoření takového ovladače si klade za cíl tato práce.

V kapitole 2 se seznámíme s principy vývoje pro mobilní operační systém iOS, pro který má být výsledná aplikace vytvořena. Kapitola 3 se věnuje popisu systému inteligentní elektroinstalace iNELS a prostředkům, které jsou použité pro komunikaci s tímto systémem. V kapitole 4 je nastíněn základní návrh aplikace, včetně specifikace komunikačních prostředků. Také přiblížíme grafickou podobu aplikace. V kapitole 5 pak bude podrobně popsána implemetace od základních tříd až po výsledný produkt, který se bude dále distribuovat. Kapitola 6 popisuje vytvořenou aplikaci a ovládání všech jejích částí. Závěrečná kapitola 7 pak shrnuje dosažené výsledky a hodnotí konečný produkt.

Kapitola 2

Vývoj aplikací pro systém iOS

V této kapitole se seznámíme s procesem vytváření aplikací i samotné distribuce výsledného produktu. Budou popsány nástroje používané při vývoji a vše, co se vývoje týká. Kapitola čerpá z oficiální dokumentace [5] a také knih [11] a [10].

2.1 Operační systém iOS

iOS [17] je operační systém vyvinutý firmou Apple, určený pro mobilní zařízení. Poprvé byl představen v roce 2007, společně s první generací mobilního telefonu *iPhone*. Systém má za sebou tedy již 5 let vývoje a v současnosti je k dispozici verze s pořadovým číslem 6. Na rozdíl od konkurence Apple nelicencuje iOS jiným výrobcům, nelze se s ním tedy setkat v jiných zařízeních, než právě v těch od americké firmy.

iOS je založený na operačním systému OS X, který je použitý v osobních počítačích Mac a sdílí s ním důležité frameworky (viz kapitola 2.4). Obsahuje tedy unixové jádro, ovšem na rozdíl od většiny ostatních unixových systémů, iOS není open-source.

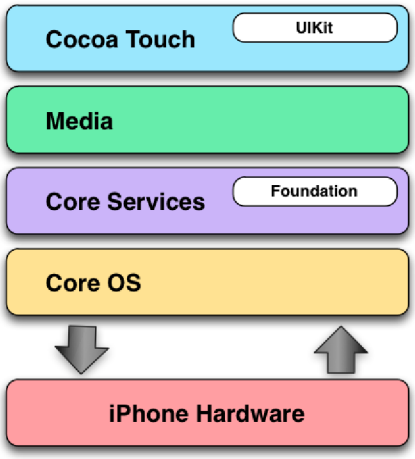
Uživatelské prostředí systému iOS je založeno na přímé interakci s uživatelem, který může vše ovládat dotykem, multi-dotekovými gesty a pomocí vestavěných čidel snímajících polohu a zrychlení pohybu zařízení. Právě díky této skutečnosti a díky celkové jednoduchosti uživatelského prostředí se iOS stal rychle velmi oblíbeným a nastartoval trend, kterým se celý trh s mobilními telefony v současnosti řídí.

2.1.1 Architektura systému

Dříve než popíšeme architekturu systému, definujme pojem *nativní iOS aplikace*. Taková aplikace je určená právě jen pro operační systém iOS a instaluje se přímo na určené zařízení. Do této kategorie tedy nepřipadají různé webové a jiné aplikace. O nativní aplikaci také můžeme říci, že se jedná o *Cocoa* [14] aplikaci. Ta je definována pomocí dvou určujících faktorů:

1. Je složena z objektů, které dědí z třídy `NSObject`, která je definovaná ve *Foundation frameworku* (kapitola 2.4)
2. Je založena na běhovém prostředí jazyku *Objective-C* (kapitola 2.3).

Dalším rozdílem mezi nativní aplikací a jinými aplikacemi je, že nativní aplikace přímo spolupracují s operačním systémem iOS a jeho vestavěnými frameworky. Operační systém se tak chová jako mezičlánek mezi aplikací a hardwarem obsaženým ve fyzickém zařízení.



Obrázek 2.1: Vrstvy architektury systému iOS.

Hlavní výhodou takového mezičlánku je, že se aplikace nemusí sama starat o případné budoucí změny v hardwaru nebo ve specifikaci zařízení. Operační systém poskytuje veškeré potřebné informace o hardwaru a schopnostech zařízení.

Samotná architektura systému iOS se skládá ze 4 oddělených vrstev (obr. 2.1): *Cocoa Touch*, *Media*, *Core Services* a *Core OS*. Takto vrstvená architektura znázorňuje úroveň abstrakce, kde vyšší stupně vrstev jsou více abstraktní a objektově orientované, a nižší stupně vrstev jsou více elementární a svázané s hardwarem zařízení. Vyšší vrstvy také závisejí na těch nižších kvůli jejich funkčnosti. Apple doporučuje pokud možno pracovat nejlépe s co nejvyššími vrstvami, protože nižší vrstvy bývají už hodně komplexní a mohou znepřehlednit samotnou stavbu aplikace. Ale je samozřejmě možné s těmito nižšími vrstvami pracovat, pokud abstraktní vrstvy neposkytnou programátorovi požadovanou funkčnost. Nyní podrobněji popíšeme výše zmíněné vrstvy systému iOS.

- **Cocoa Touch**

  Vrstva Cocoa Touch je nejvyšší vrstva celé iOS architektury. Obsahuje klíčové frameworky, na kterých nativní iOS aplikace stavějí, především nejdůležitější z nich – *UIKit framework*. Vrstva Cocoa Touch definuje základní infrastrukturu aplikací a poskytuje množství důležitých technologií, jako je například *multi-tasking*[1] a uživatelský vstup pomocí dotyků. Na UIKit frameworku iOS aplikace závisí a pokud tato knihovna není k aplikaci přilinkována (společně s Foundation frameworkem) tak aplikace nemůže fungovat.

- **Media**

  Grafika, audio a video jsou ovládány pomocí vrstvy Media. Tato vrstva obsahuje množství důležitých technologií, jako je *Core Graphics* (vykreslování 2D grafiky v aplikacích), *OpenGL ES* (mobilní verze známé 3D knihovny), *OpenAL* (zpracování zvuku) a *AV Foundation* (přehrávání multimediálních souborů v aplikacích). Skrze tuto vrstvu je také možné přistupovat k fotografiím a videozáznamům pořízených zařízením nebo tyto záznamy vytvářet.

[1] Povoluje běh několika aplikací najednou a uživatel mezi nimi může libovolně přecházet.

- **Core Services**

  Tato vrstva slouží pro ovládání základních systémových služeb, které nativní iOS aplikace používají. Nejvyšší vrstva Cocoa Touch na této vrstvě a mnoha jejích funkcionalitách velmi závisí. Vrstva Core Services také poskytuje množství velmi důležitých technologií, jako *blok objekty* (kap. 2.3.6), *GCD* (kap. 2.3.7), *ARC* (kap. 2.3.4), platby v aplikacích nebo úložiště *iCloud*[2].

- **Core OS**

  Většina funkcionality poskytovaná třemi předchozími vrstvami je postavena na Core OS vrstvě. Tato vrstva mimo jiné nabízí i několik frameworků pro přímé použití v aplikacích, například pro práci s vnitřními pohybovými čidly zařízení nebo framework s bezpečnostními funkcemi. Vrstva Core OS zapouzdřuje jádro systému a nízkoúrovňová UNIX rozhraní, ke kterým nativní aplikace nemají přímý přístup. Avšak pomocí knihovny `libSystem` lze přistupovat k některým nízkoúrovňovým funkcím jako jsou BSD sockety, POSIX vlákna nebo DNS služby, tak jak je známe z unixových systémů.

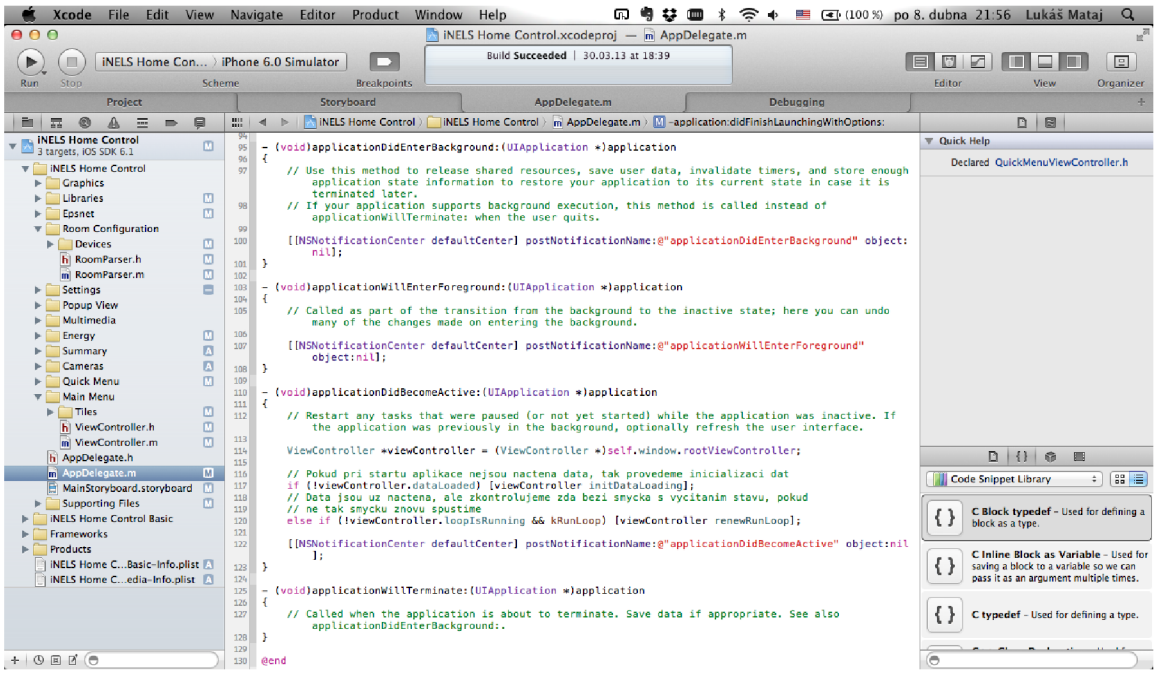
2.2 Integrované vývojové prostředí XCode

Zcela základní podmínkou pro vývoj iOS aplikací je instalace aplikace *XCode* [6], která poskytuje všechny potřebné nástroje v jednom balíku. Základem je integrované vývojové prostředí (angl. *integrated development environment*, zkráceně *IDE*) XCode, které je rozšířené o balík *iOS SDK*. Omezením pro instalaci IDE je nutnost vlastnit počítač od firmy Apple s operačním systémem Mac OS X. Na klasickém PC s operačním systémem Windows nebo s Linuxem vyvíjet aplikace nelze. Samotné IDE (Obrázek 2.2) je velice propracované a poskytuje vše potřebné od návrhu aplikace, vytvoření uživatelského prostředí, testování, optimalizaci až po výslednou distribuci.

Pro vytváření uživatelského prostředí slouží komponenta *Interface Builder*, která dovoluje utvářet vzhled aplikace pouhým vizuálním způsobem, bez nutnosti psaní kódu. Od svých prvních verzí se tento program velice zlepšil a v současné době je téměř hloupost nevyužít jeho služeb. Mnoho programátorů bylo dříve odpůrcem používání těchto „klikacích“ prostředí, ale v současné době dokáže práci velmi zefektivnit. Obvzláště to platí v případě vyvíjení mobilních aplikací. Navíc s příchodem iOS verze 5 byl představen systém takzvaných *Storyboards*, který dovoluje jednotlivé obrazovky aplikace vidět v editoru všechny najednou, včetně způsobů jakými mezi sebou interagují. Takto se proces návrhu aplikace velmi urychlí. Samozřejmě není tento způsob vhodný pro všechny aplikace, ale většinou zcela dostačuje.

Velmi důležitou součástí balíku XCode je *iOS Simulator*, který umožňuje simulaci prostředí reálného prostředí systému iOS. Prostřednictvím simulátoru je tedy možné aplikace testovat bez potřeby připojení fyzického zařízení. Toto má výhodu především při vývoji aplikace, v pozdějších fázích je vhodné aplikaci testovat i na fyzických zařízeních, které se mnohdy mohou chovat mírně odlišně než simulátor.

[2] Služba společnosti Apple, která poskytuje cloudové úložiště a umožňuje uživateli sdílet multimediání obsah mezi všemi jeho zařízeními (více na `www.apple.com/cz/icloud/`).



Obrázek 2.2: Vývojové prostředí XCode.

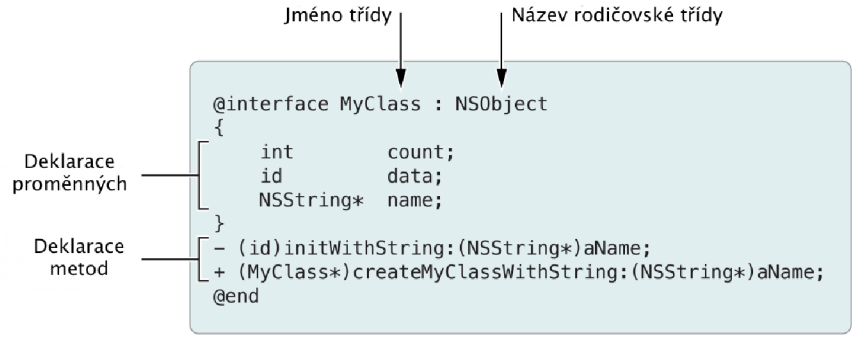
2.3 Programovací jazyk Objective-C

Pro vývoj aplikací na iOS se používá výhradně programovací jazyk *Objective-C* [1]. Je to objektově zaměřený jazyk, který se používá především u firmy Apple a jejich softwarových produktů. Objective-C je nadmnožinou klasického jazyka ANSI C, ke kterému přidává nadstavby jako je vytváření tříd a jejich metod, posílání zpráv mezi třídami, používaní *properties* (viz kap. 2.3.3), statické a dynamické typování a další vlastnosti známé z jiných objektově orientovaných jazyků. Programátor, který tedy zná například jazyk C++ nebo i Java, by neměl mít problémy s pochopením Objective-C, největší rozdíly jsou především v syntaxi. Pro seznámí s tímto jazykem lze doporučit například knihu [13], ze které bylo částečně čerpáno i v této kapitole.

2.3.1 Třídy a objekty

Podobně jako v jiných objektově orientovaných jazycích, třídy v Objective-C zapouzdřují data a metody, které s těmito daty pracují. Instance třídy se nazývá objekt. Objekt v sobě obsahuje kopie proměnných definovaných třídou a ukazatele na metody. Vytvoření objektu probíhá pomocí dvou akcí: *alokace*, kdy se pro nově vytvářený objekt rezervuje místo v paměti a *inicializace*, která objekt nastaví do počátečního stavu, který je předepsaný třídou objektu.

Definice třídy je tvořena dvěma součástmi: *rozhraním* a *implementací*. Pro rozhraní je určen zpravidla hlavičkový soubor se standartní koncovkou `*.h`. Samotné rozhraní musí být označeno direktivou `@interface` a používá se pro deklaraci proměnných a metod třídy, které mají být veřejné a zprostředkovávají komunikaci třídy s okolím. Definice rozhraní (obrázek 2.3) musí být ukončena direktivou `@end`. Za direktivou `@interface` se nachází jméno třídy a za dvojtečkou je uvedena třída, ze které se dědí. U každé třídy musí být nějaký rodič. Základní a kořenovou třídou v jazyce Objective-C je `NSObject`. Samotnou



Obrázek 2.3: Deklarace třídy.

instanci `NSObject` není možné vytvořit, slouží právě jako rodičovská třída pro základní a jednoduché třídy. Uvnitř složených závorek se nacházejí proměnné a poté jsou definovány metody. Implementační část je ohraničena direktivami `@implementation` a `@end`. Za direktivou `@implementation` se nachází název třídy. Implementace se obvykle nachází v souboru s příponou `*.m`.

2.3.2 Metody

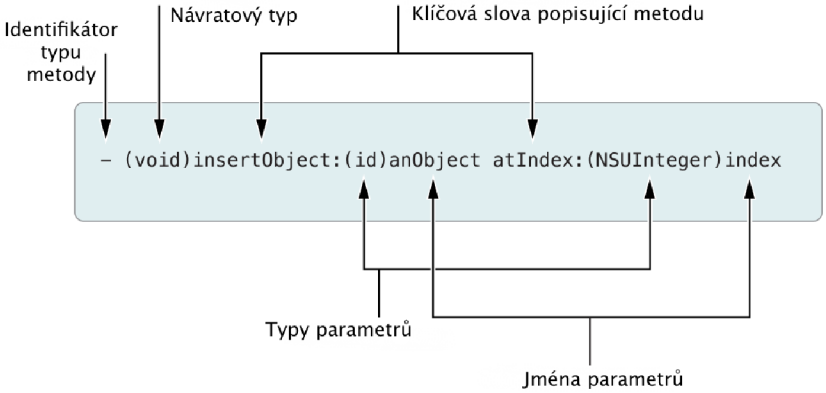
Pro komunikaci mezi jednotlivými třídami slouží metody. V Objective-C jsou dva typy metod:

- **metody instance** – pro použití těchto metod musí být nejprve vytvořena instance dané třídy,
- **metody třídy** – tyto metody se mohou volat bez vytvoření instance třídy.

Na obrázku 2.4 vidíme ukázku deklarace metody i s popisem jednotlivých částí. Počáteční identifikátor typu metody slouží právě pro rozlišení metody instance (znaménko –) a metody třídy (znaménko +). Můžeme si všimnout, že v deklaraci jsou zřetelně oddělené parametry a běžnou konvencí bývá, že se metoda pojmenuje tak, aby bylo jasné jaké parametry metoda přijímá. Volání výše odkazované metody je provedeno následujícím způsobem: `[array insertObject:object atIndex:0]`, kde `array` je instancí třídy pro obecné pole. Tato syntaxe s hranatými závorkami je další specialitou Objective-C. Je ovšem možné používat i tečkovou syntaxi. Například výraz `[array count]`, který vrací počet prvků pole, je ekvivalentní s tímto výrazem – `array.count`. Pro pořádek si ještě uvedeme, jak vypadá volání metody třídy, popsané výše: `[NSArray array]`. `NSArray` je v tomto případě pouze název třídy a nejedná se o instanci.

2.3.3 Vlastnosti třídy

Dalším specifikem jazyka Objective-C jsou *vlastnosti* (angl. *properties*). Ty představují nějaká data, která jsou zapouzdřená a uložená v objektu. Pro vytvoření vlastnosti se používá direktiva `@property`, po které následuje klasická deklarace proměnné. Ovšem díky použití této direktivy kompilátor vytvoří přístupové metody k proměnné, pomocí kterých je možno tuto proměnnou nastavovat (metoda známá jako *setter*) a nebo hodnotu proměnné číst (*getter*). To je velká výhoda oproti některým jiným jazykům, kde programátor



Obrázek 2.4: Popis deklarace metody.

tyto přístupové metody musí psát sám. Vlastnosti se deklarují uvnitř rozhraní třídy. (Pozn.: Dále se v textu bude používat výhradně anglické slovo *property*.)

2.3.4 Práce s pamětí

Objective-C neobsahuje žádný *garbage collector*[3], což na programátora klade větší nároky při práci s pamětí. Každý objekt obsahuje tzv. *čítač uchování* (angl. *retain count*), který značí, jestli se má daný objekt udržovat v paměti. Pokud tento čítač klesne na nulu, je objekt uvolněn z paměti a zaniká. Objekt je vytvořen standardní konstrukcí `[[object alloc] init]`, která alokuje paměť potřebnou pro objekt a uvede ho do počátečního stavu. Nastaví také čítač uchování na hodnotu 1. Čítač může být inkrementován pomocí volání `[object retain]`. Dekrementace čítače je provedena voláním `[object release]`. V současné verzi kompilátoru jazyka Objective-C je dostupná technologie *Automatic Reference Counting* (ARC), která správu paměti převezme částečně za programátora. Neznamená to ovšem, že by byla přidána podpora garbage collectoru, ARC pouze při překladu zdrojového kódu automaticky přidává volání `[object release]`, aby byly všechny objekty korektně uvolněny. Tímto se vývoj aplikací pro iOS velmi zefektivnil, protože programátor již více nemusí podrobně kontrolovat všechny alokace objektů a jejich uvolňování. Nicméně ARC je volitelným prostředkem, nemusí být tedy nutně využíván.

2.3.5 Moderní Objective-C

Společně s šestou verzí operačního systému iOS byly také představeny některé změny v syntaxi jazyka Objective-C. Apple tyto změny sám nazývá jako *Modern Objective-C*. Ačkoliv se jedná spíše o syntaktický cukr[4], jsou to změny, které pomáhají programátorům tvořit kód efektivněji. Nyní si tedy jednotlivé změny popíšeme.

Definice properties

Dříve bylo nutné při používání properties každou definovanou property ještě v implementačním souboru takzvaně *syntetizovat*. To se provádělo pomocí direktivy `@synthetize`

[3]Systém na automatické uvolňování paměti, která je obsazena nepoužívanými objekty.
[4]Pozměnění syntaxe jazyka pro zjednodušení zápisu nějaké konstrukce, ale bez vlivu na funkčnost jazyka.

`jmeno_property`. V moderním Objective-C se toto děje již standardně a direktiva se nemusí v kódu uvádět. Properties jsou pak také přístupné pomocí podtržítka jako prefixu. Jestliže tedy máme například property pole, nemusíme odkaz na něj získat pomocí příkazu `self.pole`, ale stačí psát pouze `_pole`.

Řazení metod

V moderním Objective-C již nyní není nutné brát ohled na to, kde je metoda definována a kde je volána. Bývalo nutností metodu definovat ještě před jejím použitím, případně její jméno uvést do hlavičkového souboru. Nyní je možné metodu volat odkudkoliv a nezáleží na tom, kde je definována.

Použití literálů

Další velmi vítanou změnou v moderním Objective-C je použití literálů, které se vztahuje konkrétně na objekty typu `NSArray` (jednoduché pole), `NSDictionary` (asociativní pole) a `NSNumber` (objekt pro uložení číselných hodnot různých typů, které jsou pak tedy ve formě objektu). U polí je rozdíl při vytváření pole a u přístupu k jeho prvkům. Pole nyní může být vytvořené takto: `NSArray *pole = @[Objekt1, Objekt2]` a přistupuje se do něj pomocí klasické syntaxe s hranatými závorkami (`Objekt *objekt = pole[0]`). Což představuje podstatné zjednodušení oproti předchozímu přístupu (vytvoření pole: `NSArray *pole = [NSArray arrayWithObjects:objekt1, objekt2, nil]`, přístup: `Objekt *objekt = [pole objectWithIndex:0]`). Pomocí přístupu k prvkům přes hranaté závorky je možné prvky také upravovat. U asociativních polí je situace podobná, pouze při vytváření pole je nutné v rámci jednoho prvku uvést klíč i hodnotu, které jsou odděleny dvojtečkou. Při přístupu do pole se pak do hranatých závorek dává přímo objekt, který plní roli klíče. U objektů typu `NSNumber` se literály používají při vytváření hodnot. Je tak možné vytvářet například celočíselné hodnoty (`NSNumber *number = @12`), float hodnoty (`NSNumber *number = @12.5f`), bool hodnoty (`NSNumber *number = @YES`) a mnoho dalších typů. Což opět zrychluje práci, například s porovnáním předchozí definice celočíselné hodnoty (`NSNumber *number = [NSNumber numberWithInteger:12]`).

2.3.6 Bloky

Bloky jsou jazykově založená technologie, přidaná k jazyku Objective-C, která umožňuje vytvářet oddělené spustitelné segmenty kódu. Ty mohou být předávány mezi metodami a funkcemi, stejně jako kdyby byly obyčejné proměnné. V Objective-C se bloky chovají jako normální objekty, mohou tedy být například přidávány do kolekcí jako je `NSArray` a `NSDictionary`. Mají také tu vlastnost, že si pamatují stav proměnných použitých uvnitř bloku, ve chvíli kdy je blok vytvořen. Fungují tedy jako *uzávěry*. Definice bloku je jednoduchá a skládá se ze 4 částí: návratový typ, jméno bloku (uvozené znakem stříšky a obklopené závorkami), parametry (také obklopené závorkami) a samotné tělo bloku, které je uvnitř složených závorek. Ukázku bloku vidíme zde 2.1. Návratový typ i parametry mohou být také samozřejmě prázdné, čili typu `void`.

Při vývoji moderních aplikací pro platformu iOS se bloky prosazují čím dál více, použití nalézají například při rychlém procházení kolekcí dat, kdy může být blok aplikován na každý prvek a tímto způsobem například pole filtrovat. Také dokáže nahradit funkci *delegáta* (kapitola 2.5.2), kde se při asynchronních událostech předává funkci blok, který

```
double (^multiply)(double, double) = ^(double a, double b)
{
    return a~* b;
};
```

Kód 2.1: Ukázka bloku.

```
dispatch_queue_t myQueue = dispatch_queue_create("MyQueue", NULL);
dispatch_async(myQueue, ^{
    // Background process
    });
});
```

Kód 2.2: Spuštění kódu v pozadí pomocí GCD.

je proveden po dokončení operace. Příkladem může být asynchronní stažení dat z internetu a jejich následné zpracování prostřednictvím bloku. Další využití je možné například u různých animací a množství dalších operací, bloky tak dokážou mnohé konstrukce zrychlit a zefektivnit.

2.3.7 Grand Central Dispatch

Grand Central Dispatch (GCD) je v rámci systému iOS další poměrně novou technologií. Je to mechanismus, který pro každý proces využívající GCD spravuje vlákna a jednu nebo více front, které obsahují bloky práce, které je nutno vykonat. GCD na základně znalosti dostupných zdrojů a znalosti front z jiných procesů vybírá bloky z front a přiřazuje je konkrétním vláknům ke zpracování. Pokud tedy programátor v aplikaci narazí na místo, kde by bylo vhodné paralelizovat, nebo provádět delší operaci v pozadí, jednoduše zavolá GCD funkci (viz kód 2.2) a nemusí se o nic jiného starat.

2.4 Systémové frameworky

Tvorba aplikací pro iOS je postavena na používání *frameworků*, což jsou systémové knihovny, které obsahují soubory metod a postupů, které může aplikace využívat. Pro slovo framework neexistuje vhodný překlad do češtiny, proto zůstaneme u anglického výrazu. Každá aplikace může používat zároveň několik frameworků a ty jsou současně sdíleny mezi různými aplikacemi. Frameworky jsou vytvářeny přímo firmou Apple a poskytují přístup k různým funkcím systému, ale umožňují také spolupráci s hardwarem. Uvedeme si zde několik těch nejdůležitějších:

- **Foundation Framework** – nejzákladnější framework, který poskytuje přístup k většině základních objektů a datových typů, umožňuje například pracovat s kolekcemi dat, obrázky, URL objekty, řetězci, číselnými hodnotami atd.,
- **UIKit Framework** – zde jsou sdruženy všechny objekty, které jsou spojené s uživatelským rozhraním a má na starosti také interakci s uživatelem,

- **Core Data Framework** – tento framework slouží pro vytváření persistentního úložiště dat v aplikaci,
- **Core Graphics Framework** – slouží k vytváření jednoduché grafiky v aplikacích, také umožňuje měnit pokročilejší grafické vlastnosti standartních komponent uživatelského rozhraní,
- **Core Animation Framework** – díky tomuto frameworku je možné provádět animace a transformace s objekty.

Existuje velké množství dalších frameworků, například pro práci s multimédii (přehrávání audia a videa, práce s fotoaparátem), vytváření 3D grafiky pomocí OpenGL, ovládání telefonních funkcí (hovory a SMS), propojení s adresářem a kalendářem a další.

2.5 Návrhové vzory

Při programování pro iOS se používá několik návrhových vzorů. Vybereme ty nejdůležitější, používané nejčastěji, které si nyní blíže popíšeme.

2.5.1 Model-View-Controller (MVC)

Tento vzor přiřazuje objektům v aplikaci jednu z následujících 3 rolí: *model*, *view* nebo *controller*. MVC vzor také rozhoduje o tom, jak mezi sebou objekty komunikují. Každý z těchto tří typů objektů je zřetelně oddělen od ostatních typů objektů. Základní princip tohoto vzoru vidíme na obrázku 2.5. MVC vzor je základem pro vytváření iOS aplikací a jeho použití přináší řadu výhod. Nyní ještě blíže vysvětlíme jednotlivé role v tomto návrhovém vzoru.

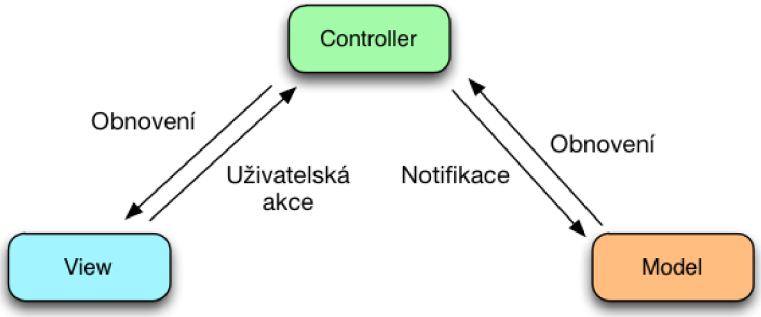
Model (model)
: Tento typ objektu zapouzdřuje data aplikace a definuje jak se s těmito daty bude manipulovat. Obvykle se model skládá z více propojených objektů.

View (pohled)
: Představuje objekt, který uživatel může vidět. Také dovoluje přímou interakci s uživatelem. Tento objekt tedy převádí data reprezentovaná modelem do uživatelského rozhraní.

Controller (kontrolér)
: Tvoří prostředníka mezi dvěma předchozími typy objektů, který reaguje na události vyvolané uživatelem prostřednictvím view objektu a na základě těchto událostí může manipulovat s daty v modelu.

2.5.2 Delegace

Dalším, velmi hojně využívaným, návrhovým vzorem je delegace. Na delegaci můžeme pohlížet jako na součást vzoru MVC, nicméně si jistě zaslouží samostatný popis. V tomto vzoru provádí objekt zvaný *delegát* akce na základě požadavků jiného objektu. V praxi objekt v nějakém okamžiku pošle zprávu delegátovi a vyžádá si data, která jsou potřebná pro další procesy vykonávané objektem. Tímto způsobem docílíme velmi dobrého znovupoužití



Obrázek 2.5: Návrhový vzor MVC.

objektu, který může fungovat nezávisle na ostatních objektech, aniž by bylo nutné vytvářet novou podtřídu. Delegace je velmi často použitá přímo v objektech obsažených ve frameworcích, ale programátor může metody pro delegaci vytvářet i ve svých nezávislých objektech.

2.5.3 Target-Action

Target-Action (cíl-akce) je návrhový vzor, používáný pro posílání zpráv mezi objekty na základě různých vnějších událostí. Událostí, která spouští akci odeslání zprávy může být cokoliv, stejně jako objekt odesílající zprávy může být libovolný objekt. Například objekt rozpoznávající gesta posílá zprávu jinému objektu, když rozpozná sobě nastavené gesto. Nicméně, návrhový vzor target-action nejvíce obyčejně nalezeneme u ovládacích prvků, jako jsou tlačítka a posuvníky. Když uživatel manipuluje s takovýmto objektem, odešle zprávu do určeného objektu. Prvek, který reaguje na vnější akce, musí být podtřídou třídy `UIControl`, uvnitř které je tento návrhový vzor implementován.

2.6 Základní struktura aplikace

Nyní stručně nastíníme, jak vypadá jednoduchá aplikace pro iOS. Součástí každé aplikace je funkce `main`, která ale zůstává neměnná a stará se o spuštění aplikace. Tato funkce je vygenerovaná automaticky při založení nového projektu v XCode, programátor ji nemusí psát sám.

Další pevnou součástí každé aplikace, která je generovaná automaticky, je tzv. *delegát aplikace*. Ten je tvořen třídou `UIApplicationDelgate` a umožňuje definovat jak bude aplikace reagovat na některá základní systémová volání. Nyní si uvedeme jaké funkce každý delegát aplikace standardně obsahuje (pro zestručnění nebudeme rozepisovat jednotlivé parametry metod, mohou být jednoduše vyhledány např. v oficiální dokumentaci [5]):

- `-(BOOL)application:didFinishLaunchingWithOptions:` – tato metoda je volána po startu aplikace a běžně se zde provádějí operace, které je potřeba provést právě po zapnutí aplikace,
- `-(void)applicationWillResignActive:` – v aplikaci dochází k přechodu do neaktivního stavu, což může být způsobeno nějakým systémovým přerušením (telefonní hovor, SMS, notifikace) nebo když uživatel odejde z aplikace,

- `-(void)applicationDidEnterBackground:` – aplikace vstoupila do režimu, ve kterém běží na pozadí a je vhodné zde uvolnit systémové prostředky, uložit uživatelská data apod.,
- `-(void)applicationWillEnterForeground:` – tato metoda se volá, pokud bude aplikace z pozadí přecházet do aktivního stavu,
- `-(void)applicationDidBecomeActive:` – při volání této metody víme, že aplikace úspěšně přešla do aktivního stavu a můžeme zde restartovat procesy, které bylo potřeba zastavit při přechodu do pozadí,
- `-(void)applicationWillTerminate:` – aplikace bude úplně vypnuta, nebude tedy ani na pozadí.

Třída `UIAppDelegate` pak také obvykle obsahuje property `window`, která slouží k přístupu k hlavnímu oknu aplikace a je možné se přes ni dostat ke třídám popisujícím uživatelské rozhraní. Samotné uživatelské prostředí je prezentované v souboru s příponou `*.storyboard`. Ten je při startu aplikace automaticky načten, konkrétní soubor je specifikován v konfiguračním souboru projektu.

Každá aplikace pak zpravidla obsahuje několik tříd `UIViewController`. Každá z nich má zpravidla na starosti jednu obrazovku aplikace. `UIViewController` je základní třída, existuje pak ještě několik rozšiřujících tříd, které z ní dědí a usnadňují práci s některými komponentami (jako příklad uvedeme `UITableViewController`, který má v sobě vestavěné metody pro práci s tabulkou). Tyto třídy tedy plní roli kontroléru, tak jak je definován v návrhovém vzoru MVC (viz kapitola 2.5.1). Obecná třída `UIViewController` má k dispozici také několik metod delegace, které umožňují programátorovi správně reagovat na konkrétní události, které během životního cyklu kontroléru mohou nastat. Opět tedy popíšeme jejich význam:

- `-(void)viewDidLoad` – metoda volaná po načtení obrazovky, používaná většinou pro počáteční nastavení komponent obrazovky nebo pro spuštění procesů, které jsou spjaty s konkrétní obrazovkou,
- `-(void)viewWillAppear:` – tato metoda je volána, když se obrazovka má objevit na displeji. Nevolá se tedy pouze po prvním vytvoření objektu obrazovky, ale pokaždé když se má zobrazit (objekt obrazovky může být načten v paměti a na displeji se střídá několik obrazovek). Zde je vhodné například obnovovat některé komponenty.
- `-(void)didRecieveMemoryWarning` – aplikace přijala systémovou zprávu, že je v systému nedostatek operační paměti, je tedy vhodné uvolnit veškerou nepotřebnou paměť,
- `-(void)viewDidUnload` – objekt obrazovky bude zrušen.

Postupnou úpravou souboru popisujícího uživatelské prostředí a tvorbou odpovídajících kontrolérů můžeme vytvářet aplikaci podle svých představ. Existují samozřejmě i některé jiné konstrukce, ovšem tato práce si neklade za cíl dopodrobna popisovat veškeré možnosti vývoje aplikací, pro tyto účely poslouží například oficiální dokumentace, případně jiné materiály dostupné na internetu. Informace uvedené v této kapitole nicméně stačí k utvoření představy, jakým způsobem jsou aplikace pro systém iOS vystavěné.

2.7 Distribuce aplikace

Distribuce vytvořené aplikace ke koncovému uživateli může být někdy poněkud zdlouhavější proces. Předně musí mít vývojář zaplacenou licenci u společnosti Apple v rámci *Apple Developer programu*. Roční poplatek za tuto licenci je 99 amerických dolarů. Pokud má vývojář tento poplatek zaplacený, dostane přístup do systému *Provisioning Portal*, kde může spravovat svoje aplikace a získat certifikáty nutné pro překlad aplikace. Přeložená aplikace se pak musí odeslat ke schválení. Schvalovací proces může být poněkud zdlouhavý a ne vždy je zaručeno schválení aplikace. Apple totiž praktikuje poměrně přísnou politiku a k zamítnutí aplikace může někdy vést i malá drobnost. Po schválení je aplikace uvolněna k prodeji do obchodu *AppStore*. Ten v současnosti patří k největším obchodům s mobilním obsahem a je zde k dispozici více než 650 000 aplikací, konkurence je tedy velká.

Kapitola 3

Inteligentní elektroinstalace iNELS

V této kapitole se budeme věnovat inteligentní elektroinstalaci iNELS, pro níž vytváříme ovladač. Také popíšeme možnosti komunikace s tímto systémem.

3.1 Popis systému iNELS

Současná generace systému inteligentní elektroinstalace iNELS vznikla za vývojové a výrobní spolupráce firem Teco, a.s., a ELKO EP, s.r.o. Hlavními částmi systému jsou výkonná *centrální jednotka* a dvoudrátová *sběrnice CIB* (Common Installation Bus), která slouží pro komunikaci a napájení všech senzorů a akčních členů rozmístěných v budovách, domech a místnostech. Mezi hlavní vlastnosti iNELS patří:

- spínání osvětlení, regulace intenzity osvětlení,
- řízení pohonu žaluzií, rolet, ale i garážových vrat, vjezdových bran a podobných zařízení,
- detekce pohybu,
- řízení systému vytápění a klimatizace,
- ovládání různých domácích spotřebičů,
- zabezpečovací funkce,
- propojení s multimediálním centrem iMM,
- vzdálený přístup přes internet.

3.1.1 Sběrnice CIB

Největší výhodou CIB [9] sběrnice je její snadné zapojení. Jedná se pouze o zavedení dvouvodičového kabelu, který musí propojovat všechny senzory a akční členy. Tato sběrnice podporuje libovolné topologie a větvení, jediným vyloučeným způsobem zapojení je zapojení do kruhu. Další výhodou je minimalizace počtu vodičů nutných pro napájení akčních členů, protože napájecí napětí a data jsou vedena společně po dvou vodičích sběrnice. Tím odpadá starost s řešením samostatného vedení napětí pro napájení jednotek na sběrnici CIB.

Sběrnice CIB je snadno rozšiřitelná a má velký dosah. Systém založený na této sběrnici je modulární a konfigurovatelný. Na jednu větev může připadat až 32 hardwarových adres, tedy 32 jednotek iNELS. Pokud je potřeba na centrální jednotku připojit více větví, je možné jejich počet rozšířit použitím externích modulů. To umožňuje nejen zvětšit počet akčních členů, ale také významně zvětšit rozsah systému. Celková délka jedné větve CIB sběrnice může být 550 m. Komunikace na sběrnici probíhá v módu master-slave.

Komunikační systém je odolný proti výpadkům a poruchám napájení. Ačkoliv sběrnice má standardní napájecí napětí 24 V DC, doporučuje se použít napětí 27 V DC. Díky tomu je možné trvalé dobíjení připojených akumulátorů 2 × 12 V, které potom při výpadku sítě zajistí trvalý chod centrální jednotky včetně všech jednotek na sběrnici CIB. Samozřejmě nebudou fungovat spotřebiče napájené ze sítě 230 V, ale systém je i nadále schopen vykonávat zabezpečovací a komunikační funkce. Dodejme, že maximální zatížení jedné větve sběrnice je 1A.

3.1.2 Centrální jednotka

Na většinu systémů jako je iNELS je kladen požadavek centralizovaného dohledu a vizualizace a v současné době i dálkového dohledu, správy a popř. i ovládání. Dohled a ovládání prostřednictvím SMS, stejně jako využití internetu pro obdobné funkce, jsou běžně vyžadovány. Uvedeným požadavkům přesně vyhovuje zvolená koncepce, kdy centrální jednotka má bez další dodatečné komunikace se senzory a akčními členy absolutní přehled o veškerých událostech v sítích CIB, a tak může oboustranně předávat data pro vizualizaci.

V případě systému iNELS se používá centrální jednotka CU2-01M [4], která je mozkem celého systému a tvoří prostředníka mezi uživatelským prostředím a všemi akčními prvky připojenými ke sběrnici. K této centrální jednotce je možné připojit přímo dvě sběrnice a pomocí externích modulů další čtyři. Celkový počet připojitelných jednotek je 192 (každá větev může obsahovat maximálně 32 iNELS jednotek libovolného typu). Centrální jednotka zajišťuje také kontrolu napájení systému (PSM) a kontroluje i síťové napětí a stav záložních akumulátorů. Při případném výpadku napájecího napětí jsou všechna data a čas zálohovány po dobu minimálně 72 hodin. Dalším vybavením je také konektor RJ45 ethernet portu, podporující rychlost 10 nebo 100 Mbps. Na čelní straně jednotky je také informační displej, který zobrazuje stav jednotky a funkční tlačítko, při jehož stisknutí a přidržení řídicí jednotka zobrazí nastavení komunikace – IP adresu, masku a bránu.

Konfigurace jednotky, a tím i celého systému, se provádí přes rozhraní ethernet, prostřednictvím konfiguračního software iNELS Designer and Manager (dále IDM), který je určen pro operační systém MS Windows XP a vyšší. CU2-01M lze vzdáleně konfigurovat a ovládat i přes internet (pokud je jednotka prostřednictvím sítě LAN na internet napojena). Prostřednictvím software v IDM se nabízí široká možnost programování funkcí. Samotná jednotka také dovoluje vzdáleně upgradovat firmware jednotek připojených ve sběrnici.

3.1.3 Komunikace s centrální jednotkou

Komunikace s centrálou probíhá přes ethernet rozhrání v podobě speciálních UDP paketů. Protokol, který komunikaci popisuje, se jmenuje EPSNET UDP a vytvořila jej firma Teco, a.s. pro komunikaci s jejími programovatelnými jednotkami Tecomat. Tento protokol má přesně dané číslo komunikačního portu, v případě systému iNELS je to 61682. UDP komunikace je použita proto, aby se předešlo zbytečné režii v podobě potvrzování paketů (jak je tomu u protokolu TCP/IP) a kvůli celkovému zrychlení komunikace. Samozřejmě to přináší rizika nespolehlivého přenosu, ale v tomto systému to až tolik nevadí, díky tomu, že každý

0 . . . 1	2	3	4 . . . 5	6 . . .	. . .	. . . n
MESI	PN	R	DPLEN	EPSNET1	. . .	EPSNETx

Tabulka 3.1: Struktura EPSNET UDP paketu.

paket sám o sobě nese celou informaci a není rozdělena do více paketů. Pokud se tedy stane, že se nějaký paket ztratí, neovlivní to ostatní data.

Data v UDP paketu (tabulka 3.1) zasílaném do centrály začínají záhlavím délky 6 bytů a následuje vlastní zpráva, která má shodnou strukturu jako standardní protokol EPSNET (viz kapitola 3.2). Význam jednotlivých hodnot v paketu je vysvětlen v tabulce (tabulka 3.2).

Byte	**Název**	**Význam**
0,1	MESI	Identifikace jednotlivých zpráv. V odpovědi se vrací stejná hodnota jaká byla v dotazu. Lze tak tedy jednoznačně určit ke kterému dotazu patří která odpověď.
2	PN	Určuje v jakém režimu se příchozí data zpracují (slave nebo multimaster).
3	R	Rezerva.
4,5	DPLEN	Určuje délku následujících dat. Byte 4 obsahuje vyšší byte délky a byte 5 obsahuje nižší byte délky.
6 a více	DATA	Posloupnost EPSNET zpráv.

Tabulka 3.2: Význam hodnot v paketu.

Navíc v jednom UDP paketu lze za jedním záhlavím poslat až 5 zpráv EPSNET, a tím komunikaci zefektivnit (nutnou podmínkou je ovšem dodržet sudý počet celkových bytů případným doplněním nulového bytu). Centrála pak tyto vícenásobné zprávy zpracovává v tom pořadí, v jakém leží za sebou. Odpověď od centrály má stejný formát. Za záhlavím délky 6 bytů následuje příslušný počet odpovědí v protokolu EPSNET.

3.2 Síť EPSNET

Uvnitř EPSNET sítě [12], která je základem komunikace s centrální jednotkou, mohou pracovat stanice ve dvou režimech:

- **nadřízená stanice (master)** - aktivní účastník, který řídí komunikaci,
- **podřízená stanice (slave)** - pasivní účastník, který odpovídá na dotazy od aktivních stanic.

Dále je možné EPSNET síť konfigurovat ve dvou různých konfiguracích:

- **monomaster** - v síti se nachází pouze jedna master stanice, ostatní jsou podřízené,
- **multimaster** - v síti může být větší množství nadřízených stanic.

V systému iNELS je použita konfigurace multimaster. V tomto režimu řídí provoz na sítí vždy jedna nadřízená stanice a po vyřízení požadavků předává řízení další nadřízené stanici. Podřízené stanice se řízení nijak nezúčastňují. Tento způsob dovoluje komunikace libovolné nadřízené stanice s libovolnou podřízenou a vzájemnou komunikaci mezi nadřízenými stanicemi.

3.2.1 Obecná struktura protokolu EPSNET

Nyní popíšeme jak vypadá obecná komunikace v EPSNET síti, tedy strukturu dotazů a odpovědí. Významy všech hodnot jsou v tabulce 3.8.

- **komunikace ve směru master - slave (dotaz)**

  1. *zpráva bez datového pole*

SD1	SA	DA	FC	FCS	ED

Tabulka 3.3: Struktura zprávy bez datového pole.

  2. *zpráva s datovým polem*

SD2	LE	LER	SD2R	DA	SA	FC	DATA	FCS	ED

Tabulka 3.4: Struktura zprávy s datovým polem.

  3. *zpráva typu token (po ní nenásleduje odpověď)*

SD4	SA	DA

Tabulka 3.5: Struktura zprávy token.

- **komunikace ve směru slave - master (odpověď)**

  1. *odpověď bez datového pole (alternativou je také krátké pozitivní potvrzení, které obsahuje pouze hodnotu SAC)*

SD1	SA	DA	FC	FCS	ED

Tabulka 3.6: Struktura odpovědi bez datového pole.

  2. *odpověď s datovým polem*

SD2	LE	LER	SD2R	DA	SA	FC	DATA	FCS	ED

Tabulka 3.7: Struktura odpovědi s datovým polem.

Data odesílaná v síti EPSNET jsou zabezpečena sudou paritou, kontrolním součtem FCS a také správnou sekvencí hodnot popsanou protokolem. Pokud je zjištěna chyba jakýmkoliv z těchto mechanismů, považuje se zpráva za chybnou a dojde k jejímu zahození. Tímto způsobem máme zaručeno, že budou doručeny pouze korektní zprávy.

Název	Význam	Hodnota
SD1	Úvodní znak 1	Pevná hodnota 0x10
SD2	Úvodní znak 2	Pevná hodnota 0x68
SD4	Úvodní znak 3	Pevná hodnota 0xDC
LE	Délka dat	Součet bytů položek DA, SA, FC, DATA
LER	Opakovaná délka dat	Posloupnost EPSNET zpráv.
SD2R	Opakovaný úvodní znak	Stejná hodnota jako LE, slouží pro kontrolu
DA	Cílová adresa	Adresa cílové stanice
SA	Zdrojová adresa	Adresa zdrojové stanice
FC	Řídící byte rámce	Závislé na konkrétním typu zprávy
DATA	Vlastní tělo zprávy	Závislé na konkrétním typu zprávy, maximálně však 246 bytů
FCS	Kontrolní součet všech bytů položek DA, SA, FC, DATA	Pro SD1: $\sum_{DA}^{FC} \mod 256$ Pro SD2: $\sum_{DA}^{FCS-1} \mod 256$
ED	Koncový znak	Pevná hodnota 0x16
SAC	Krátké potvrzení	Pevná hodnota 0xE5

Tabulka 3.8: Popis použitých hodnot.

3.2.2 Typy odpovědí

V sítí EPSNET přijde na každý dotaz odpověď, pokud proběhne samotný přenos zprávy bez obtíží. Podle úspěšnosti provedení požadované akce rozlišujeme 3 typy odpovědí:

- **pozitivní odpověď** - znamená správné provedení služby a liší se podle konkrétního typu dotazu,
- **negativní odpověď** - struktura dotazů a odpovědí je stejná, v případě negativní odpovědi se do hodnoty FC uloží jeden ze tří chybových kódů (0x02, 0x03 nebo 0x09), které ve všech případech hlásí chybnou službu a FC může mít ještě hodnotu 0x0C, která značí chybné parametry služby,
- **žádná odpověď** - pokud se nevrátí žádná odpověď, můžeme předpokládat, že odeslaný dotaz měl chybně sestavený rámec, samozřejmě v tomto případě může být na vině nějaká hardwarová chyba, ale to už bohužel systém není schopný rozlišit.

3.2.3 Komunikační služby

V sítí EPSNET je standardně k dispozici několik komunikačních služeb. Nebudeme je zde popisovat všechny, omezíme se pouze na ty, které najdou využití při vytváření výsledné aplikace.

CONNECT

Pomocí této funkce dochází k zahájení samotné komunikace i inicializaci komunikačních struktur. Syntaxi zprávy vidíme v tabulce 3.9, hodnota FC je v tomto případě nastavena na 0x49 nebo 0x69. V případě pozitivní odpovědi má zpráva stejnou syntaxi, ale hodnota FC je rovna 0x00.

SD1	SA	DA	FC	FCS	ED

Tabulka 3.9: Struktura zprávy CONNECT.

IDENT

Slouží k identifikaci připojeného systému. Můžeme tedy například zjistit verzi softwaru, verzi použitého komunikačního protokolu atd. Struktura zprávy je stejná jako u funkce CONNECT, s tím rozdílem, že hodnota FC je 0x4E nebo 0x6E. Rozdíl nastává u pozitivní odpovědi. Ta má strukturu klasické odpovědi s datovým polem (viz. kapitola 3.2.1), v jejíž datové části jsou zapsané požadované informace.

READN

Tato funkce je určena pro čtení ze zápisníkové paměti. V našem případě bude tedy sloužit pro čtení stavu jednotlivých zařízení. V jedné zprávě je možné zjistit stav více zařízení najednou. Syntaxe zprávy viz 3.10. Význam jednotlivých hodnot v datové je uveden v tabulce 3.11.

SD2	LE	LER	SD2R	DA	SA	FC	0x0B	TR1	IR1L
IR1H	LR1	...	TRn	IRnL	IRnH	LRn	FCS	ED	

Tabulka 3.10: Struktura zprávy READN.

Název	**Hodnota**
LE	$4 + 4n$ (n je počet bloků dat)
LER	$4 + 4n$
FC	0x4C nebo 0x6C
TRn	Označení registru, ze kterého se čte
IRnL	Dolní byte indexu prvního čteného registru
IRnH	Horní byte indexu prvního čteného registru
LRn	Počet čtených registrů

Tabulka 3.11: Význam hodnot použitých ve zprávě READN.

Pozitivní odpověď na tento dotaz má opět shodnou strukturu jako dotaz, ale některé hodnoty jsou rozdílné. FC má hodnotu 0x08, hodnota LE se vypočítá jako součet LR1 ... + ... LRn + 3. A konečně v datové části jsou obsahy čtených registrů.

WRITEN

Tato funkce slouží pro zápis registrů do zápisníkové paměti. V praxi bude tedy sloužit pro zapisování hodnot do různých zařízení. Struktura zprávy je shodná jako u funkce READN s těmito rozdíly: FC má hodnotu 0x43 nebo 0x63 a na konci datové části je ještě blok se zapisovanými registry. Samozřejmě se také liší výpočet hodnot LE a LER, v tomto případě je vzorec následující: $4 + 4n + \text{LW1} + \ldots + \text{LW}n$ (LWn má stejný význam jako LRn v předchozí funkci, jen se jedná o počet zapisovaných registrů). Jako pozitivní odpověď je odesíláno pouze krátké potvrzení s hodnotu SAC.

READB

Používá se pro čtení bitů ze zápisníkové paměti. V systému iNELS slouží především pro přepínání stavu zařízení, která jsou typu BOOL. Strukturu zprávy vidíme v tabulce 3.12. Význam jednotlivých hodnot je popsán zde 3.13

SD2	LE	LER	SD2R	DA	SA	FC	0x0F	TR1	IR1L
IR1H	BR1	...	TRn	IRnL	IRnH	BRn	FCS	ED	

Tabulka 3.12: Struktura zprávy READB.

Pozitivní odpověď má opět stejnou strukturu jako dotaz, rozdílné hodnoty jsou vypočítány následovně: LE = LER = 3 + n, FC = 0x08. Datová část odpovědi obsahuje čtené bity, každý z nich je ale zapsaný v samostatném bytu.

Název	**Hodnota**
LE	$4 + 4n$ (n je čtených bitů)
LER	$4 + 4n$
FC	0x4C nebo 0x6C
TRn	Označení registru, ze kterého se čte
IRnL	Dolní byte čteného registru
IRnH	Horní byte čteného registru
BRn	Index čteného bytu

Tabulka 3.13: Význam hodnot použitých ve zprávě READB.

WRITEB

Služba WRITEB je použita pro zápis bitů do zápisníkové paměti. Zpráva této služby vypadá stejně jako u READB, jen hodnota FC je rovna 0x43 nebo 0x63 a na konci datové části je blok se zapisovanými bity. Pozitivní odpověď je uskutečněna krátkým potvrzením hodnotou SAC.

3.3 Identifikace zařízení v systému iNELS

Abychom mohli se všemi zařízeními v síti komunikovat, potřebujeme o nich mít náležité informace. Již dříve jsme si řekli o softwaru IDM, pomocí kterého je možné nakonfigu-

rovat centrální jednotku. V tomto softwaru si také můžeme vygenerovat soubor nazvaný `export.pub`, který tyto informace o všech akčních členech obsahuje. Struktura tohoto souboru je pevně daná a formát popisu jednoho zařízení je uveden zde 3.14. Každé zařízení je popsané v rámci jednoho řádku a jednotlivé hodnoty jsou oddělené mezerou.

item	REG	CF	ADDR	.B	TYPE	PUB_INOUT

Tabulka 3.14: Formát popisu zařízení.

Nyní popíšeme význam jednotlivých položek uvedených v tabulce 3.14. Položka item je jedinečný název zařízení, REG značí typ registru (hodnota X pro vstupní registr, Y znamená výstupní a R je uživatelský registr). CF je takzvané pole kompatibility, které pro nás nemá význam. ADDR je přesná adresa registru, .B je obsaženo pouze pokud je zařízení typu BOOL a znamená pozici bitu v registru. Pole TYPE může nabývat hodnot BOOL, REAL nebo BYTE podle typu zařízení. A konečně pole PUB_INOUT obsahuje informaci o tom, zda je zařízení určeno pro zapisování, čtení nebo obojího. Na ukázce 3.1 vidíme vzorek souboru `export.pub`. Tento soubor je vygenerován při instalaci systému a zůstává neměnný. Vygenerování nového souboru je třeba pouze v případě přidání nových akčních prvků do systému, což v praxi nebývá moc obvyklým jevem, nicméně možné to je.

```
state_SA2_04M_Sn_RE4 Y B 29 .3 BOOL PUB_OUT
SA2_04M_Sn_RE4_ON R B 17922 .0 BOOL PUB_INOUT
SA2_04M_Sn_RE4_OFF R B 17922 .1 BOOL PUB_INOUT
SA2_04M_Sn_RE4_TRIG R B 17922 .2 BOOL PUB_INOUT
SA2_04M_Sn_RE4 R B 17932 .0 BOOL PUB_INOUT
evnt_Zapnuti_rele R B 17116 .0 BOOL PUB_INOUT
```

Kód 3.1: Ukázka souboru `export.pub`.

3.4 Multimediální systém iMM

V současné době je už téměř nedílnou součástí systému iNELS také multimediální systém iMM. Přidává nové možnosti ovládání multimediálních zařízení umístěných v budově, jako jsou televizory, audio i video přehrávače, umožňuje také připojení bezpečnostních kamer do systému. Hlavním prvkem systému je iMM klient, jehož základem jsou klasické počítačové komponenty a běží na něm linuxový systém. Je k němu možné připojit centrální úložiště dat a mít tak všechny multimediální soubory po ruce. V implementaci je zajištěna komunikace s centrální jednotkou, oba systémy spolu tedy mohou velice úzce spolupracovat. Cílem této práce je tedy i ovládání tohoto multimediálního systému bez nutnosti být přímo u něj, což opět zvýší uživatelský komfort.

3.4.1 Komunikační protokol XML-RPC

Pro spojení a komunikaci se systémem iMM je použit protokol XML-RPC [15]. Ten umožňuje jednoduché volání vzdálených procedur. Je to více méně soubor pravidel, který určuje jak používat známou metodu RPC [16], s tím rozdílem, že tady jsou data reprezentována pomocí značkovacího jazyka XML a odesílána protokolem HTTP. Tímto se docílí toho, že je možné systém ovládat z různých platforem, bez potřeby provádět nějaké změny. Samotná

Datový typ/struktura	Význam
`int`	Celé číslo
`double`	Číslo s plovoucí desetinnou čárkou.
`boolean`	Booleovská hodnota.
`string`	Řetězec.
`array`	Pole prvků.
`struct`	Struktura představující asociativní pole.

Tabulka 3.15: Datové typy a struktury používáné při komunikaci.

komunikace probíhá na principu klient-server, kdy klient posílá dotazy k serveru, který na ně odpovídá. Systém iMM tedy obsahuje vlastní implementaci, realizující protokol XML-RPC, která dovoluje vzdáleně ovládat všechny jeho funkce. XML-RPC má jednoduchou, pevně danou strukturu. Jelikož jsou data odesílána pomocí protokolu HTTP a jeho dotazu POST, je součástí zprávy HTTP hlavička (viz 3.2).

```
POST /server HTTP/1.0
User-Agent: identifikace_klienta
Host: xmlrpc.domena.cz
Content-Type: text/xml
Content-length: pocet_znaku
```

Kód 3.2: HTTP hlavička.

Druhou částí zprávy je tělo (viz kód 3.3), které obsahuje XML struktury s daty. Tělo je uvozeno značkou `methodCall`. Uvnitř je značka `methodName`, která udává jméno volané procedury. Dále je zde značka `params`, uvnitř které jsou jednotlivé parametry procedury uvozené značkami `param`. V tabulce 3.15 jsou uvedeny základní datové typy, které se při komunikaci s iMM serverem používají (výčet tedy není kompletní, nepoužívané datové typy nejsou v tabulce uvedeny). U typu `array` jsou prvky obsažené v poli ohraničené ještě značkou `data`, každý prvek je pak určen značkou `value`. Datový typ `struct` má pak každý prvek ohraničen značkou `member`, uvnitř které je jméno uvozené značkou `name` a hodnota uvozená značkou `value`.

```
<?xml version="1.0"?>
 <methodCall>
   <methodName>trida.jmenoMetody</methodName>
   <params>
     <param>
       <value><int>250</int></value>
     </param>
     <param>
       <value><double>2.31</double></value>
     </param>
   </params>
 </methodCall>
```

Kód 3.3: Tělo XML-RPC zprávy.

Odpověď od serveru dodržuje shodnou strukturu s dotazem až na to, že místo značky `methodCall` se zde nachází `methodResponse`. Pokud se na serveru při zpracování dotazu vyskytla nějaká chyba nebo byl dotaz nekorektní, vrací server chybovou zprávu. Tu poznáme podle značky `error` obsažené v odpovědi. Důvod a jméno chyby jsou pak v těle odpovědi popsány.

3.5 Vizualizace systému iNELS

Nyní se již dostáváme k tomu, jakým způsobem může být celý systém prezentován uživateli, což je pro tuto práci určující. Zde se ukazuje to, že systém iMM už je opravdu úzce spojen se systémem iNELS, protože konfigurace vizualizace se provádí právě přes iMM. Pokud ale není iMM k dispozici, je možné konfiguraci provádět přes vzdálený server. Konfigurační nástroj je v obou případech stejný. Pomocí tohoto nástroje se načte soubor `export.pub` a následně můžeme vytvářet virtuální pokoje, do nichž můžeme rozmisťovat jednotlivé prvky. Logicky zní kopírování reálných pokojů v domácnosti, nicméně vytváření těch virtuálních není nijak omezeno a lze tedy například umístit všechna světla v domě do jednoho pokoje. Pro multimédia jsou určeny jednotky, které se nazývají zóny. Ty jsou dvojího typu – audiozóna a videozóna. Opět do nich můžeme umisťovat libovolná multimediální zařízení. Jediné omezení je v tom, že zóna musí být připojena k nějakému virtuálnímu pokoji, jinak není platná. Po konfiguraci je vygenerován soubor nazvaný `rooms.cfg`, který v jednoduché XML reprezentaci obsahuje všechny vytvořené pokoje a zóny společně se zařízeními, která jsou jejich součástí. Tento soubor tedy reflektuje to, jak si uživatel přeje mít rozdělené jednotlivé prvky systému a je tedy použit pro vizualizaci v ovládacích aplikacích. V naší aplikaci bude tedy pro tento účel používán také.

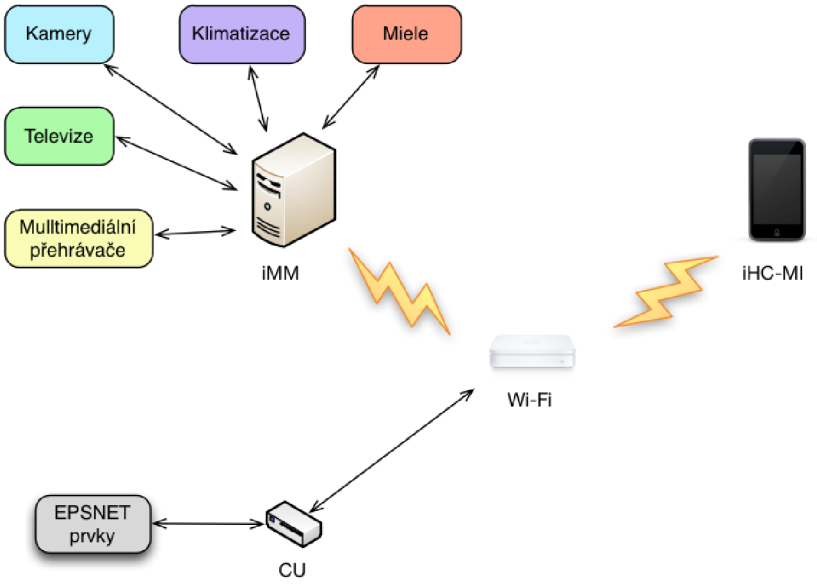
Kapitola 4

Návrh aplikace

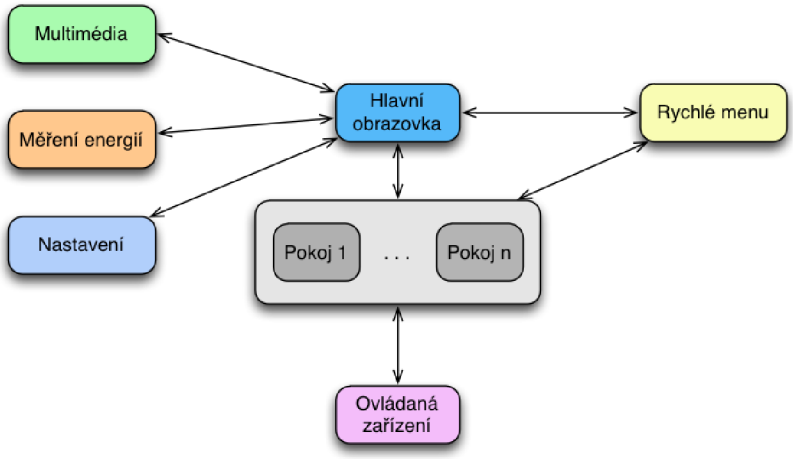
V této kapitole popíšeme předpokládanou podobu aplikace, použité vývojové prostředky a definujeme jednotlivé části projektu. Předmětem vývoje bude tedy aplikace určená pro operační systém iOS, která bude ovládat systém inteligentní elektroinstalace iNELS a také multimediální nadstavbu iMM. Jméno aplikace je iHC-MI (iNELS Home Control – Mobile iPhone) podle specifikace firmy ELKO EP.

4.1 Základní struktura projektu

Celý projekt je tedy tvořený systémem klient-server. Předmětem této práce je vytvoření mobilní klientské části. Stranu serveru představuje multimediální systém iMM a také centrální jednotka CU2-01M, jejichž implementace a specifikace je již pevně daná. Klientská aplikace bude se systémem komunikovat prostřednictvím bezdrátového Wi-Fi routeru, který se musí nalézat ve stejné síti jako oba servery. Základní strukturu propojení jednotlivých částí celého systému vidíme na obrázku 4.1.



Obrázek 4.1: Schéma celého systému.



Obrázek 4.2: Základní struktura aplikace.

4.2 Klientská aplikace

Aplikace bude vyvíjena v prostředí XCode, které je popsané v kapitole 2.2. Zde nebyla možnost jiné volby. Jelikož se jedná o aplikaci, která je vytvářená na základě požadavků firmy ELKO EP, je jasně daná stavba celé aplikace i její funkčnost. Stručně si popíšeme jak bude aplikace vypadat. Po spuštění aplikace bude k dispozici obrazovka, která bude zobrazovat jednotlivé virtuální pokoje (viz kapitola 3.5). V horní části bude název pokoje a pod ním v matici dlaždic zobrazené prvky přiřazené do pokoje. Mezi jednotlivými pokoji bude možné přecházet pomocí gesta *swipe* (rychlé posunutí prstem zleva doprava nebo naopak) přes název pokoje. Po kliknutí na název pokoje se zobrazí rychlé menu, které prvky zvoleného pokoje rozřadí do kategorií (např. zvlášť světla, zvlášť žaluzie, zvlášť klimatizace, atd.). Kliknutím na název pokoje se uživatel vrátí zpět. Pokud bude k dispozici multimediální centrum iMM, bude na hlavní obrazovce také dlaždice Multimédia, která zobrazí jednotlivé audio a video zóny a umožní jejich ovládání. Stejně tak může být k dispozici dlaždice pro přístup do obrazovky měření energií a v neposlední řadě i dlaždice pro přístup do nastavení aplikace. Jednoduchá grafická reprezentace struktury aplikace je uvedena na obrázku 4.2.

Při vývoji bude použita technologie *Storyboards* (viz kapitola 2.2), celé struktura aplikace tak bude přehledně vytvořena v rámci jednoho celku. Pro každou obrazovku pak bude žádoucí vytvořit samostatnou třídu dědící z třídy `UIViewController`, která se bude starat o potřebnou funkčnost.

4.3 Grafické uživatelské rozhraní

Vzhled aplikace byl již předem také určen firmou ELKO EP, grafické podklady budou postupně dodávány. Na obrázku 4.3 vidíme návrh hlavní obrazovky aplikace, podobný vzhled bude dodržen i v ostatních částech aplikace. Veškeré grafické prvky musí být vytvořeny ve dvou vzorcích, protože telefony iPhone mají 2 různá rozlišení displejů. Novější telefony to-

Obrázek 4.3: Navržená podoba hlavního menu aplikace.

tiž mají takzvaný *retina displej*[1]. Ve zdrojích aplikace je tak výhodnější mít každý použitý obrázek v obou rozlišeních, aby na novějších displejích aplikace vypadala dobře a naopak aby se na starších přístrojích nemusel obrázek za běhu zmenšovat, což by mohlo zpomalovat aplikaci.

4.4 Síťová komunikace

Vzhledem k povaze aplikace bude jednou z nejdůležitějších součástí síťová komunikace ze serverem, proto nastíníme jaké technologie budou použity. Ačkoliv knihovny pro síťovou komunikaci jsou již v základních frameworcích, budeme používat jiné open-source knihovny třetích stran, pro zjednodušení a celkové zpřehlednění kódu. Pro komunikaci s EPSNET sítí byla zvolena knihovna `CocoaAsyncSocket` [3]. Její součástí je třída `AsyncUdpSocket`, která, jak již název napovídá, umožňuje komunikaci prostřednictvím UDP paketů. Pro komunikaci s iMM pak bude využit `XML-RPC Framework` [2], který umožňuje jednoduchou komunikaci prostřednictvím protokolu XML-RPC. Pro každou z použitých technologií bude vytvořena specializovaná třída, která bude komunikaci zastřešovat. Tímto dosáhneme snažší rozšiřitelnosti do budoucna a také lepší znovupoužitelnosti. V rámci návrhu aplikace byla již vytvořena samostatná třída `EpsnetClient` (implementace popsaná v kapitole 5.1.1), která se stará o komunikaci prostřednictvím EPSNET protokolu. Pro další vývoj aplikace je implementace EPSNET protokolu určující.

[1] Velmi jemný displej, kde mřížka mezi pixely není pouhým okem téměř pozorovatelná. Slovo retina v češtině znamená rohovka (v oku).

4.5 Zobrazování zařízení

Pro zobrazování zařízení bude nutné vytvořit systém tříd, které budou ukládat informace o zařízeních a díky kterým se budou moci zobrazovat v grafické podobě. S ohledem na to, že aplikace musí ovládat poměrně velké množství různých a někdy velice odlišných zařízení, není pro ně možné vytvořit nějaký univerzální vzor a bude nutné je rozdělit do několika odpovídajících skupin. Pro samotnou konfiguraci aplikace a tedy samotné načtení všech zařízení do vytvořených datových struktur bude také vhodné vytvořit specializovanou třídu, která se bude starat právě o zpracování konfiguračních souborů.

Kapitola 5

Implementace

V této kapitole bude popsána samotná implementace aplikace. Podrobněji budou popsány především důležité a zajímavé části projektu, které mají největší vliv na funkčnost aplikace. Výsledkem bude popis vývoje aplikace od základních kamenů, až po výsledný produkt, připravený k distribuci.

5.1 Síťová komunikace

Nejdříve bude věnována pozornost implementaci síťových protokolů a komunikaci se servery, jelikož díky povaze aplikace tvoří síťová komunikace stěžejní část projektu.

5.1.1 Implementace EPSNET protokolu

Pro komunikaci prostřednictvím EPSNET sítě byla vytvořena třída `EpsnetClient`. Jak bylo řečeno v kapitole 4.4, používá se zde knihovna `AsyncUdpSocket` pro vytvoření komunikačních vláken. Sockety jsou definované dva, jeden slouží pro čtení a jeden slouží pro zapisování. Předejde se tak možným konfliktům při výměně dat s centrální jednotkou. Inicializační metoda objektu vypadá následovně (kód 5.1). Vidíme tedy, že objekt při inicializaci přijímá informaci o tom, k jaké adrese a portu se má připojit. Také je definována ještě další adresa a port, které slouží pro případné připojení k iMM serveru (důvody budou popsány níže). Po inicializaci dojde k připojení socketů.

```
/* Inicializace objektu */
- (id)initWithAddress:(NSString *)IP andPort:(NSInteger)port
    immAddress:(NSString *)immIP andPort:(NSInteger)immPort
{
    if ((self = [super init]))
    {
        // Inicializace dalsich potrebnych objektu
    }

    return self;
}
```

Kód 5.1: Inicializace třídy EpsnetClient.

Nejdůležitějšími funkcemi uvnitř třídy `EpsnetClient` jsou tyto: `writeValue:toDevice:`, která zapíše hodnotu na požadované zařízení a `readAll:devices`, která naopak získá hodnoty ze všech zařízení, která jsou uvedena ve vstupním parametru. Samotná komunikace je pak realizována funkcí `sendAndRecieve:withTag:`, která vytvoří EPSNET packet, označí komunikaci tagem, pošle data a zároveň připraví socket na přijmutí odpovědi. Označení tagem je důležité proto, abychom následně mohli poznat na jaký dotaz nám přišla odpověď. Odpověď je přijímána v delegátské metodě třídy `AsyncUdpSocket`, kde se podle tagu zpracuje a informuje opět prostřednictvím delegáta (tentokrát ovšem směrem ke třídě, která se jako delegát zaregistrovala při vytváření objektu `EpsnetClient`) o výsledku.

Při zapisování hodnot do zařízení se zjistí typ zařízení (`bool` nebo `float`) a podle toho se vytvoří požadavek, který se odesílá na server. U čtení stavu zařízení je situace složitější. Vzhledem k limitům EPSNET protokolu musíme zařízení nejprve rozdělit do skupin po 62. Pro každou skupinu se pak vytvoří kompletní EPSNET pakety, které se pak postupně posílají na server. Vzhledem k možnosti ovládat EPSNET zařízení prostřednictvím iMM serveru (pokud je k dispozici), je součástí třídy `EpsnetClient` také zapisování a čtení hodnot prostřednictví XML-RPC protokolu, k čemuž se používá knihovna XMLRPC. V nastavení aplikace je uložena informace o tom, zda má aplikace k dispozici iMM server. Tato informace se vyvolá při inicializaci objektu `EpsnetClient` a uloží se. Potom při každém zapisování nebo čtení podle této informace objekt určí, zda použije komunikace přes EPSNET nebo přes XML-RPC. Všechny tyto procesy se odehrávají pouze uvnitř objektu, takže delegát nepozná žádný rozdíl. Je nadále o komunikaci informován stejnými metodami s těmi samými parametry. Možnost ovládání EPSNET zařízení pomocí iMM serveru je k dispozici kvůli omezení centrální jednotky, která může přes EPSNET protokol komunikovat pouze se třemi klienty najednou. Ale pokud komunikace jde přes iMM server, toto omezení odpadá, protože s centrálou tak komunikuje pouze tento server. Tento problém se týká samozřejmě pouze rozsáhlejších systémů, v menších domácnostech je většinou limit třech zařízení dostačující.

Důležitým prvkem, o kterém je potřeba se zmínit, je třída `ExportPub`. Ta má za úkol v sobě udržovat informace o všech zařízeních definovaných v konfiguračním souboru (kapitola 3.3). Třída `ExportPub` je řešena jako jedináček, objekt je tedy vytvořen pouze jednou a je sdílený mezi ostatními objekty. Při inicializaci objektu je z nastavení získán soubor `export.pub` a následně rozparsován. Každé zařízení je reprezentováno objektem `ExportPubItem`, ve kterém jsou uloženy všechny potřebné parametry zařízení. Výsledkem je tedy pole těchto objektů. Třída `ExportPub` obsahuje metodu `getItemForName:`, díky které je možné získat objekt popisující dané zařízení.

5.1.2 Implementace XML-RPC protokolu

Jak bylo již řečeno, pro komunikaci přes XML-RPC protokol je použita knihovna XMLRPC. Vzhledem k tomu, že se při použití protokolu nemusejí vytvářet manuálně žádné speciální pakety, jako je tomu u EPSNETu, je zde situace jednodušší. Nicméně pro ještě větší zjednodušení byla vytvořena třída `XMLRPCClient`. Třída samotná je delegátem objektu `XMLRPCConnection`, který je definován v knihovně XMLRPC. Nově vytvořená třída obsahuje metodu `createAndSendRequestForMethod:andParameters:`, která z názvu metody a parametrů vytvoří potřebný požadavek typu `XMLRPCRequest` a odešle ho na server. Odpověď je poté zpracována a o výsledku operace je prostřednictvím delegace informován zaregistrovaný objekt. V konkrétních objektech by se samozřejmě mohla používat i samotná knihovna XMLRPC, nicméně díky vytvoření třídy `XMLRPCClient` je použití přehlednější a dá se lépe přizpůsobit požadavkům aplikace. Při komunikaci s iMM serverem je například

nutná autentizace, která zde probíhá prostřednictvím 3 metod. Ale vzhledem k tomu, že už je obsažená v objektu klienta, nemusí se dále řešit a použití je efektivnější.

5.2 Třída AppDelegate

V této třídě (obecně popsaná v kapitole 2.6) neprobíhá to nejdůležitější, přesto je tu několik záležitostí, které je žádoucí popsat. Po prvním startu aplikace jsou do uživatelského nastavení (sdílený systémový objekt `NSUserDefaults`) uloženy informace o výchozích pořadích některých prvků (prvků na hlavní obrazovce, prvků v rychlém menu). O těchto pořadích a jejich nastavování si více řekneme v odpovídajících kapitolách. V třídě `AppDelegate` aplikace musí reagovat na systémové události. Zde nás bude zajímat přepnutí aplikace do pozadí a naopak vyvolání do aktivního stavu. Při upozadění aplikace musíme zrušit běh smyčky v hlavním okně, kde probíhá neustálé vyčítání stavů jednotlivých zařízení (bude popsáno v kapitole 5.3.3). Pokud bychom tuto smyčku nezrušili, mohlo by se stát, že systém aplikaci úplně vypne, protože v systému iOS není povoleno provádět operce na pozadí (kromě specifických funkcí jako jsou polohové služby nebo přehrávání hudby). Při vrácení aplikace do popředí (případně zapnutí) je pak nutné tuto smyčku znovu spustit. Ovšem pouze v případě, že smyčka už předtím běžela. Pokud ne, znamená to, že aplikaci čistě spouštíme a nemá načtená žádná data, a proto zavoláme metodu třídy hlavního okna pro úvodní načtení dat. Zapnutí smyčky s vyčítáním stavů pak probíhá už v režii třídy hlavního okna.

5.3 Hlavní okno aplikace

V této kapitole popíšeme hlavní okno aplikace, které je tvořeno třídou `ViewController`. Na hlavní obrazovce jsou zobrazeny jednotlivé pokoje a do nich přiřazená zařízení. Dozvíme se také více o reprezentaci jednotlivých zařízení a jejich grafickém zobrazení.

5.3.1 Konfigurace pokojů a zařízení

Začneme spíše pod kapotou a popíšeme jakým způsobem je načítána konfigurace pokojů a jednotlivých zařízení. Pro zařízení bylo vytvořeno několik speciálních tříd, které kopírují to, jak jsou definována v konfiguračním souboru `rooms.cfg`. Popis těchto tříd je uveden v tabulce 5.1.

Třída, která má na starosti zpracování souboru `rooms.cfg`, se nazývá `RoomParser`. V hlavní metodě `getRooms` se nejdříve z uživatelského nastavení načte konfigurační soubor, který je poté postupně zpracováván. Každý pokoj je pak reprezentován objektem typu `NSDictionary`, tedy asociativním polem. Základní strukturu tohoto pole se zpracovaným pokojem vidíme v tabulce 5.2. Výsledkem je pak klasické pole těchto objektů. Při samotném zpracování zařízení také hlídáme pořadí jednotlivých skupin prvků, aby se nemuselo kontrolovat později a výsledkem bylo již nyní seřazené pole. Z nastavení se tedy získá pořadí skupin prvků (v nastavení je uloženo pole seřazených řetězců s přesnými názvy skupin prvků, tak jak jsou pojmenovány v konfiguračním souboru). Jednotlivé prvky jsou ukládány do pomocných polí a ta jsou pak na základě získaného nastavení seřazena do výsledného pole. Mezi zařízení také zařadíme několik „falešných zařízení“, která budou představovat speciální dlaždice, většinou pro přístup do dalších sekcí aplikace. Jako například dlaždice pro vstup do nastavení, ovládání multimédií, sekce s ovládáním spotřebičů Miele nebo sekce s přehledem o energiích.

Název třídy	Popis zařízení
`EpsnetDevice`	Tato třída reprezentuje obecné EPSNET zařízení a slouží pro uložení následujících prvků: světla, jednoduchý spínač, odvlhčení, ovladač dveří, vrat a garáží, topení, zavlažování, větrání.
`HeatControlDevice`	Třída sloužící pro jediné zařízení, a to ovládání vytápění v místnosti.
`MeterDevice`	Zde jsou uložena zařízení, která zobrazují měřič (pro různé hodnoty).
`SceneDevice`	Reprezentace scény, která provede několik akcí najednou.
`ShutterDevice`	Třída reprezentující zařízení pro ovládání žaluzií a rolet.
`ThermalDevice`	Tato třída je vytvořena pro uložení teploměru.

Tabulka 5.1: Třídy pro uložení zobrazovaných zařízení.

Klíč	Hodnota
`name`	Jméno pokoje
`devices`	Pole obsahující zařízení v pokoji
`zones`	Multimediální zóny v pokoji

Tabulka 5.2: Struktura objektu se zpracovanými pokoji.

5.3.2 Základní prvky hlavního okna

Hlavní okno se skládá z několika základních součástí. V horní části je panel tvořený objektem třídy `UIScrollView`. Na něm se zobrazuje název zvoleného pokoje a také tlačítka pro přepínání pokojů. Pod ním je ještě menší pruh, který může zobrazovat různé stavové informace. Hlavní část okna ovšem zabírá prostor pro zobrazení dlaždic, které reprezentují jednotlivá zařízení. Tento prvek je tvořen objektem třídy `SSCollectionView`, která je součástí projektu `S.S. Toolkit` [7]. Jedná se o open-sourcový projekt a tato konkrétní třída je náhradou za třídu `UICollectionView`, dostupnou přímo v iOS SDK. Ovšem ta je k dipspozici pouze pro iOS verze 6 a novější a vzhledem k tomu, že vytvářená aplikace je určena pro iOS od verze 5, byla zvolena výše zmíněná třída. Obě třídy tedy fungují stejně – zobrazují kolekce dat. Pro zobrazení dvou sloupců dlaždic je to ideální. Jejich funkčnost je založená na delegaci. Od delegáta získávají potřebné informace například o tom, kolik má být zobrazeno prvků, jak mají být veliké a podobně. Opačným směrem je pak delegát informován o tom, kdy je která dlaždice stisknuta, aby na tuto událost mohl patřičně reagovat. Každá dlaždice dědí z třídy `SSCollectionViewItem`. Jelikož máme několik různých typů zařízení, bylo vytvořeno i několik různých typů dlaždic. Ty jsou popsány v tabulce 5.3.

Každá třída dlaždice obsahuje metodu `configureTileWithDevice:`, která dlaždici nastaví podle konkrétního zařízení. Parametr se zařízením má obecný typ `id` kvůli univerzálnosti, což je třeba zejména u třídy `TileView`, která zobrazuje více typů zařízení. V iOS SDK pak existuje metoda `isKindOfClass:`, podle které můžeme určit třídu objektu a podle toho přizpůsobit konfiguraci. Kvůli požadované animaci dlaždic při stisknutí byla ještě upravena i samotná základní třída `SSColectionViewItem`. Dlaždice se při stisknutí otáčejí okolo svislé osy o 180 stupňů, čehož je dosaženo aplikací transformace 5.2. Také bylo nutné provést ještě

Název třídy	Popis dlaždice
`HeatControlTileView`	Zobrazuje zařízení třídy HeatControlDevice.
`MeterTileView`	Zobrazuje zařízení třídy MeterDevice.
`ThermalTileView`	Zobrazuje zařízení třídy ThermalDevice.
`TileView`	Univerzální dlaždice, slouží pro zobrazení všech ostatních zařízení.

Tabulka 5.3: Typy dlaždic v hlavním okně.

```
[CATransaction begin];
CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"transform"];
CATransform3D transform = CATransform3DMakeRotation(M_PI_2, 0.0f, 1.0f, 0.0f);
[animation setToValue:[NSValue valueWithCATransform3D:transform]];
[animation setAutoreverses:YES];
[animation setRemovedOnCompletion:YES];
[animation setDuration:0.2];
[[self layer] addAnimation:animation forKey:@"180"];
[CATransaction commit];
```

Kód 5.2: Aplikace 3D rotace.

jednu úpravu – některé dlaždice musí totiž reagovat na dlouhý stisk. Proto byly přepsány patřičné funkce, které odchytávají dotyky a hlídá se délka dotyku. Pokud délka dotyku překročí jednu sekundu, vyšle se o tom informace delegátovi.

5.3.3 Zobrazování dlaždic

Nyní se dostáváme k samotnému načtení a konfiguraci dlaždic. Pokud nejsou data ještě načtena, je volána metoda `initDataLoading`. Ta nejdříve zkontroluje, zda jsou všechny potřebné konfigurační soubory k dispozici, pokud ne, je zobrazena chybová hláška. Jinak jsou pomocí objektu `RoomParser` načteny všechny pokoje se všemi zařízeními a výsledek je uložen do property nazvané `rooms`. Dále je také vytvořen objekt třídy `EpsnetClient` pro komunikaci s centrálou. Pak následuje konfigurace obsahu hlavního okna. Nejdříve je volána metoda `loadCurrentDevices`, která si načte zařízení, jež náleží pouze do aktuálně zvoleného pokoje. Do property `devices` si ale uloží pouze jména iNELS prvků, které zařízení ovládají. Tyto informace pak budou sloužit k vyčítání stavů zařízení, kde potřebujeme právě názvy iNELS prvků. Poté je spuštěna smyčka `runLoop` pro vyčítání stavů zařízení a obnoví se prvek obsahující dlaždice, aby se promítla načtená konfigurace do aktuální podoby hlavního okna.

Smyčka pro vyčítání stavů není doslovně smyčka (jako například neustále se opakující cyklus `while`), ale opakovaně volaná metoda `runLoop`. V ní se pokaždé zavolá metoda EPSNET klienta pro získání stavu požadovaných zařízení. Po přijetí odpovědi jsou získaná data zpracována a pak je volána metoda `[self performSelector:@selector(runLoop) withObject:nil afterDelay:kLoopInterval]`, která mezi systémové události zaregistruje další spuštění metody `runLoop` po uplynutí časového intervalu (standardně 0.3s). Vidíme, že volání smyčky následují po sobě s minimálními časovými intervaly a je tak spotřebováváno nemalé množství systémových prostředků. Bohužel v současnosti je to jediný možný

způsob jak získávat potřebná data. Ideální by bylo řešení ve formě push zpráv, kdy by server zasílal aplikaci sám změny ve stavu zařízení až v tu chvíli, kdy by k nějakým skutečně došlo.

Nyní se ještě vrátíme ke zpracování dat, která získáme od centrální jednotky. EPSNET klient data již předává uložená v podobě asociativního pole, kde klíčem je název iNELS prvku a hodnotou je skutečná hodnota prvku. Díky tomu tak můžeme v poli rychle vyhledávat. Po přijetí dat tedy procházíme všechna zařízení aktuálního pokoje a vyhledáváme v získaném poli hodnotu jejich iNELS prvku. Zařízení pak nastavíme novou hodnotu. Metody pro nastavení hodnoty jsou u tříd představujících zařízení řešeny tak, že mají návratový typ `BOOL`, který značí jestli se hodnota změnila nebo ne. Díky tomu si můžeme zapamatovat indexy pouze těch zařízení, kterým se změnila hodnota a následně obnovit jen jim odpovídající dlaždice. Díky tomu, že pokaždé neobnovujeme všechny dlaždice, je zpracování dat a jejich prezentace o poznání rychlejší.

U dlaždic je důležité patřičně reagovat na uživatelské akce. Jak bylo již řečeno, děje se tak prostřednictvím delegace třídy `SSCollectionView`. Konkrétně se jedná o metody `collectionView:didSelectRowAtIndex:` a `collectionView:didLongSelectRowAtIndex:`. V prvním případě se jedná o jednoduché zvolení dlaždice. Na základě indexu načteme zařízení odpovídající dlaždici. Opět se zde pracuje s obecným typem `id`, skutečnou třídu zjistíme až později. Podle typu zařízení a jeho stavu pak provedeme odpovídající akci. Jsou samozřejmě dlaždice, u kterých se žádná akce neprovádí, jako jsou například ty se zobrazením teploty nebo s měřící stupnicí. Dále také máme speciální dlaždice, které po stisku zobrazí jiné části aplikace. Dlouhý stisk dlaždice má význam pouze u několika typů zařízení. Konkrétně je to zařízení pro ovládání vytápění (třída `HeatControlDevice`), kde se tímto přepíná automatický a manuální režim a také dále obecná EPSNET zařízení, která mají hodnotu typu `REAL`. U nich se dlouhým stiskem zobrazí ovládací prvek, který umožňuje jezdcem vybrat požadovanou hodnotu. Tento prvek je také možné ovládat pomocí náklonu telefonu, k čemuž jsou používána vestavěná čidla iPhonu.

Zbývá se ještě zmínit o přepínání jednotlivých pokojů. To je zprostředkováno horním prvkem hlavního okna, který je tvořen objektem typu `UIScrollView`. Ten funguje tak, že může mít nastavený obsah, který je větší než jeho skutečná velikost. V našem případě je obsahem několik objektů `UILabel`, které zobrazují názvy pokojů (je vždy vidět jenom jeden). Pokud se objektu `UIScrollView` ještě nastaví vlastnost zvaná stránkování (angl. paging), tak je umožněno uživateli pomocí gesta `swipe` posouvat obsah po stejně velkých částech. Tyto uživatelské události jsou v aplikaci odchytávány a pokud dojde ke změně obsahu (tedy zobrazení jména jiného pokoje), voláme metodu `loadCurrentDevices`, která podle zvoleného pokoje uloží aktuální zařízení. Při příštím volání metody `runLoop` se tak už bude posílat žádost o zjištění stavu nových zařízení, které odpovídají zvolenému pokoji.

5.4 Rychlé menu

Tato kapitola bude popisovat takzvané „rychlé menu“, které slouží pro zobrazení zařízení jiným způsobem. Sdružuje je do skupin a uživatel má tak větší přehled o konkrétních zařízeních.

5.4.1 Základní prvky okna

Horní lišta zůstává stejná jako v případě hlavního okna a její funkčnost také zůstává stejná – slouží k přepínání jednotlivých pokojů. Pod horní lištou je pak další lišta, ve které se

zobrazují ikony skupin jednotlivých zařízení. Pro tyto objekty je vytvořena třída nazvaná `QuickMenuTileView`. Pod nimi je pak objekt třídy `UITableView` (klasická tabulka), která slouží pro zobrazení jednotlivých zařízení. Stejně jako v případě hlavního okna musíme pro každé zařízení vytvořit specifickou třídu, v tomto případě buňku tabulky. V tabulce 5.4 jsou uvedeny všechny typy těchto buněk. Podobně jako u dlaždic hlavního okna je i zde k dispozici metoda pro konfiguraci buňky s univerzálním parametrem typu `id` představujícím zařízení.

Název třídy	**Popis**
`LightCell`	Slouží pro zobrazení jednoduchého EPSNET zařízení s hodnotu typu BOOL.
`LightFadingCell`	Zobrazuje EPSNET zařízení s reálnými hodnotami.
`ShutterCell`	V této buňce se zobrazuje zařízení představující rolety nebo žaluzie.
`SwitchCell`	Obecná buňka, která má v sobě obsaženo spínací tlačítko.
`CameraCell`	Buňka pro zobrazení kamery.

Tabulka 5.4: Typy buněk v rychlém menu.

5.4.2 Zobrazení zařízení

Celá logika získávání hodnot zařízení zůstává v režii hlavního okna. To je aktivní i nadále, přestože není vidět na obrazovce. Systém iOS takto zachovává hierarchii vytvořených oken. V rychlém menu tak máme k dispozici jenom odkaz na pole se zařízeními, nevytváříme nové. Hlavní okno pak po načtení hodnot posílá notifikaci, abychom v rychlém menu věděli, že máme obnovit zobrazená zařízení. V závislosti na tom, jakou máme zvolenou skupinu zařízení, se celé pole vyfiltruje pomocí metody `loadCurrentDevices`, aby byla zobrazena pouze zařízení, která do dané skupiny patří. To se tedy děje po každém přepnutí skupiny. Při každé změně pokoje prostřednictvím horní lišty se pak také posílá notifikace hlavnímu oknu, aby mohly být načteny odpovídající zařízení pro vyčítání jejich stavu. Po jednoduchém kliknutí na horní lištu dojde k zavření okna s rychlým menu. Uživatelské akce (zvolení buňky v tabulce), které jsou vyhodnoceny v závislosti na zařízení se provádějí pomocí EPSNET klienta hlavního okna, k němuž máme dostupný ukazatel. Ovšem v rychlém menu už také přicházejí ke slovu některé prvky, které jsou ovládány prostřednictvím iMM serveru a ty si popíšeme v následujících podkapitolách.

5.4.3 Ovládání bezpečnostních kamer

Součástí rychlého menu je i možnost zobrazení a ovládání bezpečnostních kamer. Přístup ke kamerám je možný pouze přes iMM server. Při konfiguraci aplikace získáme ze serveru seznam všech připojených kamer, včetně všech informací, které potřebujeme k ovládání kamery. Při získávání dat z kamery, ale i při jejím ovládání musíme také znát jméno a heslo ke kameře. Ovládání kamer je řešeno přes obyčejné HTTP požadavky, které jsou přesně specifikovány právě v získané konfiguraci. Obraz je z kamer poskytován ve formátu MJPEG (Motion JPEG – každý snímek videa je komprimován pomocí standardu JPEG).

Přímo v rychlém menu jsou zobrazeny náhledy jednotlivých kamer, spolu s jejich jménem a tlačítkem, které umožňuje spustit nebo vypnout záznam. Informace o tom, zda některá

z kamer nahrává, získáváme periodicky ze serveru. Náhledy zde nejsou řešeny jako realtime videa, ale po 4 sekundách se obnovující obrázky. Při pokusu o zobrazení klasického videa v náhledu a větším množství kamer aplikace začala spotřebovávat velké množství systémových prostředků.

Po kliknutí na některý z náhledů kamery se zobrazí další okno, tvořené objektem třídy `CameraViewController`. Při vytvoření mu nastavíme property `camera`, kam uložíme potřebné informace o kameře v podobě objektu `NSDictionary`. Hlavním prvkem objektu `CameraViewControler` je speciálně vytvořená třída `CameraImageView`, která dědí z třídy `UIImageView` (objekt pro zobrazení obrázku). V této třídě probíhá zobrazování videa. Stačí nastavit URL pro video, uživatelské jméno a heslo. Tato třída je vytvořena tak, aby se obrazu dala měnit velikost a je možné jí tak znovu použít na více místech aplikace, kde je potřeba zobrazovat video z bezpečnostní kamery. Pro získávání obrazu je vytvořen objekt `NSURLConnection`, kterému se nastaví požadavek s adresou kamery. Prostřednictvím delegace pak provedeme autorizaci a získáváme data z kamery. Dostáváme čistá data (třída `NSData`) a proto musíme tato data ještě převádět na obrázek, který pak zobrazíme. Tímto tedy získáváme sled obrázků, které snímá kamera. V okně `CameraViewController` pak máme ještě vytvořený jednoduchý ovladač. Kamery je možné posouvat doleva, doprava, nahoru i dolů a také provádět zoom. Ovladač je vytvořený tak, že se zobrazí pokud se uživatel dotkne obrazovky. Po době krátké nečinnosti pak zase zmizí. Kromě tlačítek je také možné ovládat kameru dotykovými gesty. Pro pohyb do konkrétního směru je to `UISwipeGestureRecognizer`, který zachytává gesto *swipe*[1] a pro zoom je to `UIPinchGestureRecognizer`, zachytávající gesto *pinch*[2]. Nakonec i zde je možnost spustit nebo zastavit záznam kamery.

5.4.4 Ovládání klimatizací

Přes iMM server je také možné ovládat některé klimatizace. V současnosti se jedná o klimatizace značky LG, které podporují vzdálené ovládání a LG u nich uvolnilo API. Přehled o připojených klimatizacích získáme ze serveru při počáteční konfiguraci aplikace. Nemusíme pak po každém zobrazení sekce kontrolovat, jaké jsou připojené klimatizace. Díky tomu je tedy nutné v případě připojení nové klimatizace provést novou konfiguraci, ale to se v praxi moc často nestává. Stavy klimatizací se periodicky získávají ze serveru a ukládají se do property `climsDetails`, což je asociativní pole, kde klíčem je název klimatizace a hodnotu objekt `NSDictionary`, kam se ukládají získané informace ze serveru. Po každém přijetí dat pak kontrolujeme jestli se uložená hodnota změnila a pokud ano, tak obnovíme obsah tabulky se zobrazenými klimatizacemi.

Při základním zobrazení vypnuté klimatizace v rychlém menu je zobrazeno pouze jméno klimatizace a tlačítko pro zapnutí. Pokud je klimatizace zapnutá, můžeme u ní zobrazit ještě menu s dalšími možnostmi. Provedeme to tak, že na buňku s klimatizací klikneme, buňka se rozšíří a pod klasickým obsahem se objeví toto menu. Prostřednictvím něj se pak dá ovládat nastavení teploty a různých režimů klimatizace. Zde jsou posílány pouze jednoduché příkazy na iMM server, který obstarává další komunikaci přímo s klimatizací.

[1] Rychlé přejetí prstem po displeji.

[2] Přiblížení dvou prstů položených na displeji k sobě nebo roztažení od sebe.

5.4.5 Ovládání Miele spotřebičů

Posledním typem zařízení, které můžeme zobrazovat v rychlém menu, jsou kuchyňské spotřebiče značky Miele. Narozdíl od klimatizací zde získáváme seznam připojených spotřebičů až po zobrazení dané sekce. Nejdříve tedy voláme metodu `getMieleDevices`. Pokud se vyskytne nějaká chyba, voláme metodu opakovaně. Po získání seznamu všech připojených zařízení pak periodicky voláme metodu `getMieleInfo`, díky které pak server vrací stavy všech spotřebičů. Po přijetí data ukládáme do property `mieleInfo`. Kontrolujeme také, zda se data od předchozího stavu změnila a případně obnovíme obsah tabulky zobrazující spotřebiče. Samotné zobrazení Miele spotřebičů není příliš složité, opět v základu zobrazujeme pouze název spotřebiče a pokud má definovány nějaké akce, tak i tlačítko pro provedení akcí (tyto akce se spouští odesláním klasického HTTP příkazu, protože Miele spotřebiče jsou připojeny přes web server). Na buňku se zařízením můžeme ještě kliknout a pak se nám zobrazí doplňkové info a stavu spotřebiče.

5.4.6 Zkratky pro zobrazení sekcí

Vzhledem k velmi úzkému propojení hlavního okna s rychlým menu máme možnost i vytvoření několika zkratek pro zobrazení specifické sekce rychlého menu. Pro tuto potřebu jsou vytvořeny speciální dlaždice. První a nejjednodušší možností je zobrazení sekce s Miele spotřebiči. Zde je pouze jednoduchá dlaždice, po jejímž stisku se zobrazí rychlé menu se správně zvolenou sekcí. Pak zde máme dlaždice reprezentující klimatizace. Tyto dlaždice již mají více funkcí - jednoduchý stisk slouží pro zapnutí nebo vypnutí klimatizace a dlouhým stiskem se dostaneme do rychlého menu se zobrazenou sekcí s klimatizacemi. Pokud v hlavním okně použijeme některou z těchto zkratek, je nově vytvářené instanci `QuickMenuViewController` nastaven parametr s informací o tom, jakou sekci chceme zobrazit. Na základě tohoto parametru je správná sekce po zobrazení okna nastavena.

5.5 Multimédia

Pokud máme k dispozici multimediální server iMM, můžeme pomocí aplikace také ovládat multimédia. V této kapitole podrobněji popíšeme tuto sekci aplikace a všechny možnosti ovládání.

5.5.1 Konfigurace zón

Všechna multimediální zařízení jsou rozdělené do skupin zvaných zóny. Celkem máme 2 typy zón: audiozóna a videozóna. Jak napovídá název, v audiozóně ovládáme hudbu a ve videozóně přehrávání filmů, televize a navíc prohlížení fotek. Ovšem ve videozóně můžeme ovládat i audio, vše záleží na konfiguraci. Tu můžeme nalézt v souboru `rooms.cfg`. Zpracovaná konfigurace zón se stejně jako u pokojů získává z objektu `RoomParser`. Další potřebná data pak už získáváme přímo ze serveru při běhu aplikace. Hned po startu získáme seznam přehrávačů, které jsou používány v konkrétních zónách. Přehrávač je v tomto případě reprezentován IP adresou a typem (audio, video, foto). Pro přehrávač je vytvořená nová třída `Player` a všechny tyto objekty jsou uloženy v property `players`. Bez těchto přehrávačů sekce nemůže vůbec fungovat, proto při případné chybě informujeme uživatele, necháme okno bez obsahu a po krátké době zkusíme seznam získat znova. Ostatní získávaná data popíšeme dále u konkrétních prvků.

5.5.2 Základní prvky okna

Nyní přiblížíme základní strukturu okna. V horní části aplikace jsou dvě lišty `UIScrollView`, které plní stejnou funkci jako u okna s rychlým menu. V první liště se tedy zobrazuje název zóny a slouží pro přepínání mezi zónami, ve druhé se pak zobrazují až 4 ikony podle toho, jaký multimediální obsah máme v zóně k dispozici. Pod nimi je zde ještě jedna menší lišta, která zobrazuje stav multimediální zóny. Tento prvek je tvořen objektem `UILabel`. Pod informační lištou se vedle sebe nacházejí dvě tabulky `UITableView`, jejichž obsah se mění podle potřeby. Posledním prvkem tohoto okna je ovládací panel, který obsahuje klasická multimediální tlačítka (např. ovládání hlasitosti, spouštění a zastavování přehrávání, spuštění náhodného režimu, smazání playlistu, atd.). Opět máme vytvořených několik typů buněk, které jsou zobrazovány uvnitř obou tabulek. Jejich popis ale necháme až do dalších podkapitol, v tomto případě to bude vhodnější kvůli přehlednosti.

5.5.3 Zobrazování adresářové struktury

Pro účely zobrazování složek a souborů slouží tabulka v levé části okna. Funkce této tabulky se nemění, zůstává pořád stejná. Pouze se mění adresářová cesta, kterou serveru předáváme jako parametr pro získání obsahu složky, podle toho jaký typ multimédií zrovna zobrazujeme. Pro ukládání informací o souborech byla vytvořena třída `File`, jejíž popis vidíme v tabulce 5.5. Pro zobrazení objektů tohoto typu v tabulce je vytvořena buňka `DirectoryCell`.

Název property	**Popis**
`name`	Název souboru/složky.
`isDir`	Příznak, který udává jestli jde o složku.
`path`	Celá cesta k souboru.
`upPointer`	Příznak pro speciální typ souboru, který slouží pro přepnutí o úroveň výš v adresářové struktuře.
`allFiles`	Příznak pro speciální typ soboru, který přesune celý obsah složky do playlistu.

Tabulka 5.5: Struktura objektu `File`.

Pro celou adresářovou strukturu máme vytvořenou property `fileTree` typu `NSArray`, která obsahuje pole s objekty typu `File`. Tato pole se ukládají na index určený tím, jak hluboko jsme zanořeni. Ze serveru nedostaneme najednou celou strukturu kořenové složky, ale načítáme soubory právě po jednotlivých úrovních. Pokud tedy uživatel zvolí složku, ze serveru se stáhne její obsah a pak se zobrazí. Pokud uživatel zvolí konkrétní soubor, provede se akce podle typu multimediálního obsahu.

5.5.4 Playlist a práce s multimédii

Playlist je zobrazen v pravé tabulce v rámci okna s multimédii. Údaje o playlistu získáváme opět ze serveru a to periodicky, aby se změny neustále zobrazovaly. Pro položky playlistu byla vytvořena třída `PlaylistFile`, která ukládá všechny potřebné údaje (jméno, cesta, typ souboru a příznak jestli je položka zrovna přehrávána). Soubory do playlistu můžeme přidávat z vedlejší tabulky souborů tím, že na ně klikneme. Pokud zvolíme položku z playlistu, volá se na serveru metoda pro přehrávání konkrétního souboru. Výjimku nalezneme

pouze u ovládání televize. Tam se po kliknutí na konkrétní kanál zobrazuje v pravé tabulce televizní program získaný ze serveru. Přehrávání programu se provádí delším podržením názvu kanálu. V tabulce je tedy zobrazováno několik typů buněk (viz tabulka 5.6).

Název třídy	**Popis**
`AudioCel`	Slouží pro zobrazení audio soboru, obsahuje název a délku skladby.
`AudioPlayCell`	Buňka pro audio soubor, který je právě přehráván. Obsahuje navíc ukazatel procent odehraných ze skaldby.
`VideoCell`	Zobrazuje náhled videa, jeho název a typ sobuboru.
`VideoPlayCell`	Stejný jako `VideoCell`, ovšem navíc zobrazuje ukazatel procent odehraných z videa.
`PhotoCell`	Buňka zobrazující náhled fotky a její název.
`EPGCell`	Slouží pro zobrazení jedné jedné položky TV programu.

Tabulka 5.6: Typy buňek v tabulce s playlistem.

Pro kompletní ovládání multimédií musíme ze serveru získávat ještě další data. Jako například hlasitost ve zvolené zóně a informaci o tom, jestli je zapnuto náhodné přehrávání nebo opakování přehrávání. Také získáváme status zóny, tedy který soubor je právě přehráván. Tento status, který je v podobě řetězce, je také bohužel jedinou informací o tom, kolik času je odehráno z přehrávaného soboru. Proto tento status musíme zpracovat pomocí regulárních výrazů, abychom u přehrávaného souboru mohli zobrazit jeho procentuální stav. Každou zónu také můžeme vypnout, případně znovu zapnout. Tato akce se provádí dlouhým podržením názvu zóny.

5.6 Měření energií

Jednou z nejnovějších součástí celého systému iNELS a iMM je možnost měření energií a zobrazování jejich stavu. Právě tuto část aplikace popíšeme v následující kapitole. Jedná se především o doplňkovou součást aplikace, která má spíše informační hodnotu. Není možné nějak upravovat či nastavovat energetické systémy. Ale pomocí přehledných grafů a tabulek je uživateli prezentován stav odběru energií a umožňuje mu tak významně optimalizovat chod domácnosti a ušetřit. Veškerá data se opět získávají pomocí XMLRPC metod z iMM serveru. Vzhledem k tomu, že na serveru není k dispozici metoda pro zjištění, jestli je měření energií dostupné, je tato sekce zobrazena uživateli vždy. Pokud nejsou k dispozici žádná data, pouze se zobrazí prázdný obsah.

5.6.1 Základní prvky okna

Hlavní okno sekce (`GraphViewController`) pro měření energií se opět drží zažité struktury. Nahoře se nacházejí dvě známé lišty `UIScrollView`. Tentokrát je ovšem v horní liště uveden pouze název sekce aplikace (tedy Měření energií), přepínání zde žádné není. A ve druhé liště se zobrazují 4 ikony pro přepínání mezi možnými typy měřitelných energií (voda, plyn, elektřina a celkový přehled). Pod lištami se nachází panel s ovládacími prvky, který dovoluje především měnit časové období zobrazených dat. Pro tento panel bylo vytvořeno několik upravených tříd: `EnergyButton` (tlačítko pro volení období) a `EnergyChecmark` (zatržítko

pro volení dalších parametrů). Posledním prvkem tohoto okna je tabulka `UITableView`, která pak prezentuje samotná data.

5.6.2 Zobrazování dat

Data získáváme ze serveru pomocí několika metod, které jsou odlišeny tím, pro jaké časové období data chceme a pro jaké energie. Podle typů energií se také mění podoba ovládacího panelu. Buňky tabulky jsou ve všech případech stejné, tam se rozdíly dělat nemusí. Po přijetí dat ze serveru jsou data zpracována a uložena do datových struktur, odkud mohou být dále získávána. Obnovení těchto dat probíhá pouze při změně parametrů, nestahují se periodicky.

5.6.3 Grafy

Z hlavního okna měření energií se můžeme dostat do sekce se zobrazením grafu. Pro vykreslování grafu byla použita open-source knihovna `CorePlot` [8], která nabízí komplexní zobrazování všech možných typů grafů. Knihovna používá klasický model delegace pro získání obsahu grafu. Data jsou získávána ze serveru v lehce jiné podobě, protože potřebujeme mít data rozdělena po malých časových úsecích, aby mohl být graf podrobně vykreslen. Graf zabírá podstatnou část obrazovky, ale v horní části okna je malé menu, které dovoluje zapínat a vypínat jednotlivé součásti grafu. Můžeme tak mít najednou zobrazené elektřinu, plyn i vodu, nebo mezi nimi libovolně přepínat. Toto menu má zaregistrované rozpoznání gesta *drag*[3]. Pokud tedy dojde k tažení prstem shora dolů, zobrazí se z horní části rozšířené menu, realizované třídou `GraphMenuViewController`, kde je možné podrobněji nastavit parametry.

5.7 Nastavení aplikace

Jednou z nejdůležitějších částí aplikace, bez níž by nemohla fungovat, je sekce s nastavením (`SettingsViewController`). Tato sekce se drží používání klasických komponent ze systému iOS (hlavně tedy tabulek `UITableView`), není nijak graficky upravená. Je zde tedy plně využito potenciálu *Storyboards* (viz 2.2), kde jsou všechny tabulky navrženy prostřednictvím grafického prostředí. V kapitole 6 jsou popsány všechny položky nastavení a jejich možnosti.

5.7.1 Stažení konfigurace ze serveru

Jedná se asi o nejdůležitější část nastavení, proto jí popíšeme trochu podrobněji. Nejprve zkontrolujeme, jestli uživatel správně zadal IP adresu a port serveru, pokud ne, zobrazíme hlášku a necháme uživatele korektně nastavit požadované údaje. Pak již následuje stahování dat. Jelikož se jedná o časově náročnější operaci, je uživateli zobrazen indikátor aktivity, aby aplikace nevypadala nečinně a uživatel neměl potřebu stahování přerušit. Data se získávají klasicky pomocí XMLRPC klienta. Ze serveru postupně stahujeme všechna potřebná data, jmenovitě to jsou: soubor `export.pub`, IP adresa centrální jednotky, všechny pokoje, kamery a klimatizace. První 3 jsou velice důležité a pokud by při jejich stahování došlo k chybě nebo by se nepodařilo zpracovat příchozí data, nebyla by vůbec umožněna nová konfigurace aplikace a uživatel by byl informován o chybě. Data se tedy stahují kontinuálně. Když se stáhne a zpracuje jedna položka, přejdeme na další, dokud nemáme staženou

[3] Prst se přiloží na objekt a táhne se s ním.

kompletní konfiguraci. Získaná data průběžně ukládáme do patřičných datových struktur, které po skončení stahování uložíme do uživatelského nastavení `NSUserDefaults`, odkud si pak můžou konfiguraci načíst jiné komponenty aplikace. Pokud takto úspěšně konfiguraci uložíme, nastavíme si příznak, že musíme načíst novou konfiguraci v aplikaci. Pokud tedy zavřeme okno s nastavením, dojde k načtení úplně nových dat v hlavním okně.

5.8 Testování aplikace

Velmi důležitou součástí vývoje je samozřejmě testování. Proto byla každá část aplikace důsledně testována již při vývoji. Testování probíhalo ve spolupráci s firmou ELKO EP, kde pro to bylo příznivější prostředí, především díky dostupnosti různých testovacích serverů. Pan Konečný (jednatel firmy) pak také testoval aplikaci v prostředí svého domova, kde má systém iNELS nainstalovaný. Aplikace se také průběžně zasílala různým partnerům společnosti, aby měli přehled o stavu vývoje. Díky tomu tedy aplikace procházela důkladným testováním a díky tomu bylo nalezeno množství drobných chyb a odchylek od požadavků. Aplikace byla také později předváděna na několika výstavách (Integrated Systems Europe v Amsterodamu, ELTETA ve Stuttgartu, CONECO v Bratislavě a CONSTRUMA v Budapešti), kterých se firma ELKO EP zúčastnila. Aplikace tedy byla testována v reálném prostředí s reálnými systémy a byla tak pečlivě připravena k distribuci.

5.9 Distribuce aplikace

Vytvořený produkt bylo podle specifikace nutné rozdělit na několik různých verzí podle poskytované šíře funkčnosti. Jsou celkem k dispozici 3 verze aplikace:

- **Basic** - pouze základní ovládání systému iNELS,
- **Multimedia** - k verzi Basic přidává podporu iMM, kamer a klimatizací,
- **Full Control** - k verzi multimédia přidává podporu měření energií a ovládání Miele spotřebičů.

Toto samozřejmě znamenalo zamyslet se nad možností, jak takové rozdělení na verze provést. Vytvořit 3 samostatné projekty by byl nesmysl, pokud by se prováděla nějaká změna základní funkčnosti, prováděla by se pak třikrát. Naštěstí se řešení našlo díky samotné podobě projektu, jak jej ukládá XCode. Každému projektu tak můžeme nastavit několik *targets* (cílů pro překlad) a těm upravit parametry pro překládání. Pro každou verzi tak byl vytvořený jeden *target*. Každému se nastavily preprocesorová makra, díky kterým pak v kódu při překladu můžeme určit, kterou z verzí právě překládáme a podle toho dát překladači správný kus kódu. U každé verze bylo také nutné použít jiné certifikáty pro překlad, protože ve skutečnosti budou výstupem 3 samostatné aplikace, i když v rámci jednoho projektu.

Samotná distribuce pak už probíhala v režii firmy ELKO EP, tedy prostřednictvím jejich vývojářského účtu. V době odevzdání diplomové práce byla už v obchodu s aplikacemi AppStore verze Basic[4] a verze Multimedia byla těsně před vydáním (i když aplikace byla hotová téměř i na úrovni Full Control, jenom se pozdrželo vydávání). Zveřejnění aplikace se neobešlo bez problémů, ukázalo se, že Apple ke schvalování aplikací přistupuje opravdu

[4]Ke stažení na adrese `https://itunes.apple.com/us/app/inels-home-control-basic/id603747883`.

pedantsky. I když k popisu aplikace byl přidán podrobný návod, jak aplikaci otestovat na veřejném serveru, byla aplikace nejdříve zamítnuta. Důvodem bylo to, že si testeři vyžádali video, kde bude ukázána podrobně funkčnost aplikace. Po dodání tohoto videa již byla aplikace schválena a uvolněna do AppStore.

5.9.1 Promo aplikace

Vedlejším produktem při vývoji aplikace byla menší reklamní aplikace pro potřeby firmy ELKO EP. Slouží pro prezentaci systému iNELS široké veřejnosti. Obsahuje ovládání několika málo prvků společně se zobrazením kamery. Ovládá dva pokoje ze showroomů (v Holešově a v Praze). Tato aplikace byla rovněž zveřejněna na AppStore a dosáhla poměrně zajímavých počtů stažení.

Kapitola 6

Popis aplikace a ovládání

V této kapitole popíšeme výslednou aplikaci a uvedeme jak se její jednotlivé části ovládají. Kapitola tedy slouží jako jednoduchý manuál k aplikaci iHC-MI. Nebudou zde uvedeny všechny screenshoty z aplikace, zbývající obrázky se nalézají v příloze.

Hlavní okno

Podobu hlavního okna vidíme na obrázcích 6.1, B.1 a B.2. Podstatnou část obrazovky tedy zabírají dlaždice, které představují jednotlivá ovládaná zařízení. Většina dlaždic reaguje na krátký dotyk, který provede akci přiřazenou k zařízení. Některé dlaždice informačního charakteru (zobrazení teploty apod.) na dotyk nereagují vůbec, jiné dlaždice reagují i na dlouhý dotyk. Jsou to zařízení, kterým je možné nastavit reálnou hodnotu, jejíž nastavení se provede právě po dlouhém dotyku. Krátký dotyk v tomto případě pouze zapne nebo vypne zařízení, ale přesnou hodnotu neovlivní. Dlouhým dotykem se pak ovládá ještě zařízení typu Heat Control (vytápění), kde se volí mezi manuálním a automatickým režimem. Poslední dlaždicí, kde se používá dlouhý dotyk, je ikona klimatizace, kde se přemístíme do ovládacího panelu klimatizace v rychlém menu. V horní části je pak zobrazen název pokoje. Pomocí tlačítek nebo gesta přejetí prstem můžeme jednotlivé pokoje přepínat.

Rychlé menu

V případě rychlého menu (viz obrázek B.3) zůstává ovládaní prakticky totožné, s tím rozdílem, že zařízení mají jinou grafickou podobu a jsou rozdělená do skupin podle typu. Jednotlivé skupiny je možné přepínat pomocí ikon v horní části obrazovky. Navíc nám ovšem přibývají některá speciální zařízení, jako jsou kamery, klimatizace a Miele spotřebiče. Jejich ovládání si nyní popíšeme:

- **Kamery** (obr. B.4) – Jejich ovládání je intuitivní. Na ovládacím panelu máme směrová tlačítka, která můžou kamerou pohybovat a také tlačítka pro přiblížení a oddálení. V horním pravém rohu je tlačítko s křížkem, které zavře okno s kamerou. Pod ním je tlačítko, které mění zobrazení kamery (může tak být zobrazen skutečný obraz, který se celý vejde na obrazovku, roztažený obraz který vyplní celou obrazovku, nebo roztažený obraz, který si zachová poměry stran, takže část obrazu nemusí být vidět). Pokud se telefon překlopí do landscape režimu, kamera se přizpůsobí. Posledním ovládacím prvkem je tlačítko v levém horním rohu, které ovládá nahrávání kamery.

Obrázek 6.1: Hlavní menu.

- **Klimatizace** (obr. B.6) – V základním zobrazení můžeme klimatizace pouze vypnout nebo zapnout. Pokud je ovšem klimatizace zapnutá, můžeme stiskem buňky zobrazit ovládací panel. Tam se v horní části nacházejí dvě tlačítka se symboly šipky, pomocí kterých můžeme nastavit požadovanou teplotu. Aktuální teplota v místnosti i nastavená jsou zobrazeny hned vedle. Dále se pod nimi nachází pětice ikon, určující režim klimatizace. Režimy jsou následující – klimatizace, ventilace, odvlhčení, topení a automatický režim. Ve spodní části se pak nachází další čtyři ikony, které umožňují nastavit různé doplňkové funkce klimatizace jako je rychlost ventilátoru, otáčení lamel nebo zámek klimatizace.
- **Miele** (obr. B.6) – U spotřebičů Miele je většinou možné jenom zapnutí nebo vypnutí. Sekce má spíše informativní hodnotu, protože u zapnutých spotřebičů klikem na ně zobrazíme stavové informace.

Multimédia

Pod prvkem sloužícím pro přepínání zón (stejná funkce jako v případě přepínání pokojů na hlavní obrazovce) jsou ikony sloužící pro přepínání jednotlivých režimů – Audio, Video, Televize a Fotografie. Ve střední části obrazovky se nacházejí dva panely. Levý panel je ve všech režimech stejný, jeho obsahem je adresářová struktura. Zobrazují se zde jak celé složky, tak soubory. Pokud uživatel klikne na složku, zobrazí se její obsah. Jestliže se nenacházíme v kořenovém adresáři, tak je před soubory a složky vložen prvek (ikona trojúhelníku), který slouží pro návrat o úroveň výše. Pokud se nacházíme ve složce, kde jsou

už jenom samostatné soubory, zobrazuje se také prvek (s ikonou hvězdy) pro přidání celé složky do playlistu. Soubory je také možné přidávat do playlistu po jednom, stačí na něj kliknout. Pravý panel slouží právě pro zobrazení položek playlistu. Grafická podoba se liší podle toho, jaký multimediální obsah zrovna přehráváme. Přehrávaná položka playlistu je vždy zvýrazněna. Kliknutím na položku playlistu se spustí její přehrávání. Od popsaného chování se odlišuje pouze režim Televize. Zde žádný playlist není, místo něj se zobrazuje televizní program. Aby se zobrazil, musíme v levém panelu zvolit požadovaný kanál. Televizní program můžeme zobrazovat 4 dny dopředu. Zvolením konkrétního pořadu se také může naplánovat jeho nahrávání. Přehrávání kanálu se provádí delším stiskem názvu kanálu v levém panelu.

V dolní části se nachází několik multimediálních tlačítek (přeskakování souborů, přetáčení souborů, zastavení přehrávání, spuštění přehrávání a také pozastavení). Dále je zde ovladač hlasitosti v zóně a další 4 tlačítka – funkce ztlumení, opakování playlistu, náhodné přehrávání playlistu a vymazání playlistu. Pokud je vymazán playlist, dojde také k zastavení přehrávání souboru. Některá tlačítka můžou být v určitých režimech schovaná – například u přehrávání fotek nemá smysl zobrazovat ovládání hlasitosti apod. Grafickou podobu sekce s multimédii můžeme vidět na obrázcích B.8, B.9 a B.10.

Měření energií

V této sekci si můžeme vybrat ze 4 základních režimů, které jsou reprezentovány samostatnými ikonami – Voda, Plyn, Elektřina a Přehled. Pod těmito ikonami je ovládací panel, kde můžeme volit co si přejeme zobrazovat. Tento panel se mírně odlišuje v závislosti na zvoleném režimu. Pokud je zobrazená elektřina (obr. B.11), můžeme vybírat mezi 5 zónami, které mohou být v domácnosti vytvořené. U vody je možné vybírat mezi zobrazením studené a teplé vody. Ostatní režimy žádné speciální nastavení nemají. Pro všechny režimy je společné volení období, které má být zobrazeno. Můžeme si vybrat den, týden, měsíc nebo celý rok. Je také možné zvolit konkrétní období. Pro tento příklad jsou připravena tlačítka Od a Do, po jejichž stlačení dojde k zobrazení prvku pro výběr datumu. Ve spodní části se v tabulce zobrazují už konkrétní data. Je zobrazena jak spotřeba v jednotkách tak cena.

Po stisknutí tlačítka s ikonou grafu se dostaneme právě do zobrazení přehledných grafů (viz obrázek B.13). Tato obrazovka je připravena tak, že může zobrazovat libovolné hodnoty, tedy klidně i všechny sledované energie najednou. V horní části je připraven informační panel, kde vidíme jaká data se v grafu zobrazují. U každé položky je také zobrazený puntík s barvou, aby bylo jasné kde ji můžeme v grafu nalézt. Přímo na informačním panelu také můžeme přepínat zda chceme zobrazit spotřebované jednotky nebo cenu. Tyto hodnoty jsou v grafu na svislé ose, vodorovná osa je vyhrazena pro časové jednotky. Pomocí stažení informačního panelu dolů zobrazíme menu (viz obrázek B.14), kde můžeme volit, co všechno se má v grafu zobrazovat. Můžeme tedy volit mezi 5 zónami elektřiny, plynem, teplou a studenou vodou. Dále je zde také možné nastavit konkrétní časové období, které bude v grafu zobrazeno. A jako poslední zde máme opět tlačítka pro přepínání mezi zobrazením jednotek spotřebované energie nebo ceny.

Nastavení

V této části aplikace můžeme provádět veškerou konfiguraci aplikace. V následujícím seznamu jsou popsány všechny položky obsažené v nastavení:

- **IP Adresa serveru** - Zde je možné zadat adresu iMM severu. Je samozřejmě možné podle potřeby přidávat další, mazat stávající a vybírat mezi nimi. Je nutné zadat IP adresu a port. Jelikož je možné přidávat více adres, mezi kterými lze potom volit, je každá položka také pojmenována.
- **Vynutit stažení dat** - Slouží jako tlačítko a začne stahování nové konfigurace ze serveru.
- **IP adresa CU jednotky** - Slouží k zadání adresy centrální jednotky iNELS systému. Má smysl pouze tehdy, když nemáme k dispozici iMM server.
- **Stáhnout data z veřejného serveru** - Opět se používá jenom když si nemůžeme konfiguraci provést vlastním iMM serverem. Pro tento případ je pak vytvořen veřejný server, kam můžeme nahrát svoje konfigurační soubory, získáme specifický kód, který v aplikaci zadáme a stáhne se konfigurace ve správném formátu.
- **Nastavení pořadí prvků v zobrazení Dlaždic** - Zde můžeme libovolně upravovat pořadí dlaždic, případně zakazovat a povolovat jejich zobrazení.
- **Nastavení pořadí prvků v zobrazení Rychlý přehled** - Slouží pro změnu pořadí jednotlivých skupin v rychlém menu a také je můžeme libovolně skrývat.
- **Nastavení pořadí pokojů** - Protože v konfiguračním souboru není nijak definováno přesné pořadí pokojů, je zde dána možnost uživateli si toto pořadí přizpůsobit.
- **Sledované zóny** - V této části si můžeme nastavit, u jakých multimediálních zón bude aplikace hlídat stav na hlavní obrazovce a tyto stavy budou také zobrazovány.
- **O aplikaci** - Informační sekce.
- **Nová kamera** - Zde můžeme přidávat dlaždice kamery na hlavní obrazovku a mít k nim tak rychlejší přístup. Vytvářet můžeme libovolné množství kamer. U každé nastavíme v jakém pokoji se bude zobrazovat a dlaždice také může zobrazovat větší počet kamer než jednu (maximálně 4).
- **Odstranit kameru** - Slouží pro vymazání kamer vytvořených v předchozí části.

Kapitola 7

Závěr

Cílem této práce bylo vytvoření aplikace pro systém iOS, která bude schopna ovládat inteligentní elektroinstalaci iNELS. V kapitole 2 jsme se seznámili se základy vývoje pro systém iOS, kapitola 3 pak popisovala možnosti komunikace se systémem iNELS. V kapitole 4 byl představen základní návrh aplikace, podle kterého pak byla aplikace implementována, což je popsáno v kapitole 5. Kapitola 6 stručně popisuje ovládání vytvořené aplikace.

Výsledkem práce je tedy kompletně zpracovaná aplikace, přesně podle požadavků firmy, která práci zadávala. Tato aplikace tedy dokáže pohodlně a efektivně ovládat inteligentní budovy. Celý systém včetně aplikace je dimenzován tak, že inteligentní budovou nemusí být jenom domácnost, ale i celé firmy. Aplikace je v současnoti již nasazená v ostrém provozu a je používána zákazníky firmy ELKO EP. Jako nejnáročnější se ukázala ta skutečnost, že je aplikace velice komplexní. Je to spojení různých technologií dohromady, které mají ve výsledku činit jednolitý celek. Díky tomuto rozštěpení a velkému množství funkcí byl vývoj časově velice náročný. Samozřejmě k tomu přispělo i to, že je aplikace celá zpracovaná vlastní grafikou a nepoužívá standardní komponenty systému iOS. Ovšem díky tomu je aplikace velice uživatelsky přitažlivá.

Nejvíce implementačně náročný byl celý proces prezentace jednotlivých ovládaných zařízení, od zpracování konfiguračního souboru až po zobrazení zařízení uživateli. Musel být vymyšlen systém jak všechny zařízení ukládat v paměti a pomocí nich konfigurovat grafické prvky. Pro tento účel bylo vytvořeno množství speciálních tříd, které dané požadavky splňují. Uživateli je také umožněno v nastavení aplikace měnit libovolně pořadí zobrazovaných prvků, vše tedy musí být prováděno dynamicky s ohledem na tato nastavení. Vzhledem k tomu, že aplikace neustále získává data ze serveru, bez ohledu na to jestli se změnila, musela být aplikace také optimalizována tak, aby to nebylo poznat v plynulosti používání. Ke slovu tak přicházejí různé asynchronní techniky, aby nedocházelo k blokaci grafického uživatelského prostředí a obnovují se průběžně jenom ty prvky, které to potřebují. S neustálou potřebou připojení k serveru se také pojí další problém – aplikace musí korektně reagovat na přesunutí do pozadí, kdy je v neaktivním stavu a na přepnutí zpět do stavu aktivního. Byl tak vytvořen systém, kdy se v každém okně ukončují a obnovují odpovídající procesy, což je při množství ovládaných prvků a použití několika různých komunikačních technologií poměrně komplikované.

Za největší přínos práce považuji spolupráci s velkou firmou s mezinárodním zastoupením a spolupráci se zaměstnanci. Díky tomu tak výsledný produkt bude používán zákazníky firmy a práce nezapadne v archivu. Vývoj samozřejmě také posloužil pro seznámení s novými technologiemi, například studium protokolu EPSNET bylo poměrně zajímavé a neobvyklé.

7.1 Možný rozvoj do budoucna

Jelikož se jedná o aplikaci, jejíž vývojový cyklus není pevně stanoven, je téměř jisté, že se do budoucna bude dále vyvíjet. Do aplikace se mohou přidávat další moduly, stejně jako se do systému iNELS mohou přidávat další zařízení. Výsledná aplikace je především komerční produkt firmy, tudíž se musí dynamicky vyvíjet podle přání zákazníků.

Literatura

[1] Programming with Objective-C [online]. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/, 2012 [cit. 2013-04-08].

[2] The Cocoa XML-RPC Framework [online]. https://github.com/corristo/xmlrpc, 2013 [cit. 2013-01-08].

[3] CocoaAsyncSocket [online]. https://github.com/robbiehanson/CocoaAsyncSocket, 2013 [cit. 2013-01-08].

[4] CU2-01M: Návod k obsluze [online]. http://www.inels.cz/media/pdf/manualy/cz-sk/Manual_CU2_01M_cz_sk.pdf, 2013 [cit. 2013-01-08].

[5] iOS Developer Library [online]. http://developer.apple.com/library/ios, 2013 [cit. 2013-01-08].

[6] XCode 4 [online]. https://developer.apple.com/xcode/, 2013 [cit. 2013-04-08].

[7] S.S. Toolkit [online]. http://sstoolk.it, 2013 [cit. 2013-05-14].

[8] Core Plot [online]. http://code.google.com/p/core-plot/, 2013 [cit. 2013-05-15].

[9] Klaban, J.: iNels a sběrnice CIB: Moderní systém inteligentní elektroinstalace [online]. *Automa*, ročník 12, 2008 [cit. 2013-01-08], ISSN 1210-9592, http://www.odbornecasopisy.cz/index.php?id_document=38218.

[10] Kochan, S. G.: *Výukový kurz programování pro Mac OS X a iPhone*. Computer Press, 2010, ISBN 978-8025126547, 552 s.

[11] Mark, D.; LaMarche, J.; Nutting, J.: *Beginning iPhone 4 Development: Exploring the iPhone SDK*. New York: Apress, 2011, ISBN 978-1430230243, 657 s.

[12] Teco: *Sériová komunikace programovatelných automatů Tecomat - Model 32 bitů*. 15 vydání, 2009.

[13] Čada, O.: *Cocoa: Úvod do programování počítačů Apple*. Praha: Grada, první vydání, 2009, ISBN 978-80-247-2778-3, 200 s.

[14] Wikipedia: Cocoa [online]. http://en.wikipedia.org/wiki/Cocoa_(API), 2012-05-03 [cit. 2013-05-13].

[15] Wikipedia: XML-RPC [online]. http://cs.wikipedia.org/wiki/XML-RPC, 2012-10-25 [cit. 2013-01-08].

[16] Wikipedia: Remote procedure call [online]. http://en.wikipedia.org/wiki/Remote_procedure_call, 2013-04-03 [cit. 2013-04-09].

[17] Wikipedia: iOS [online]. http://en.wikipedia.org/wiki/IOS, 2013-04-07 [cit. 2013-04-08].

Příloha A

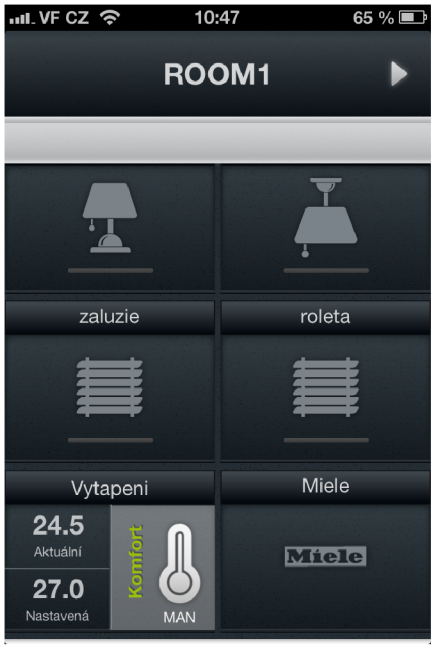
Obsah DVD

- Zdrojové kódy aplikace ve složce `iHC`.
- Zdrojový kód této práce ve složce `Text\src`.
- Tato práce ve formátu pdf ve složce `Text`.
- Obrázky zachycující obrazovky aplikace ve složce `Images`.
- Soubor `README` s popisem možností otestování aplikace.

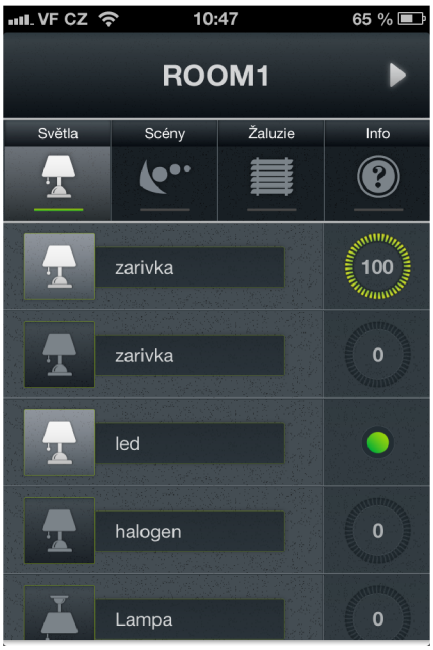
Příloha B

Snímky obrazovek aplikace

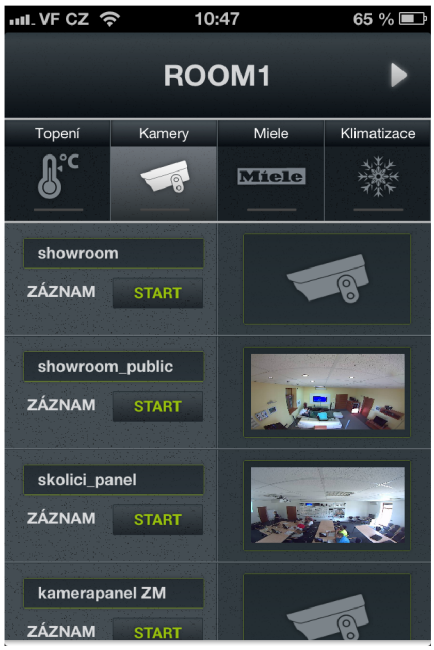
Obrázek B.1: Hlavní menu.



Obrázek B.2: Hlavní menu – další zařízení.



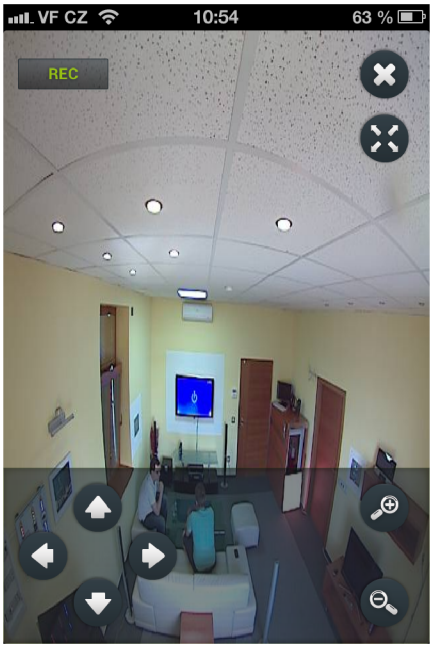
Obrázek B.3: Rychlé menu.



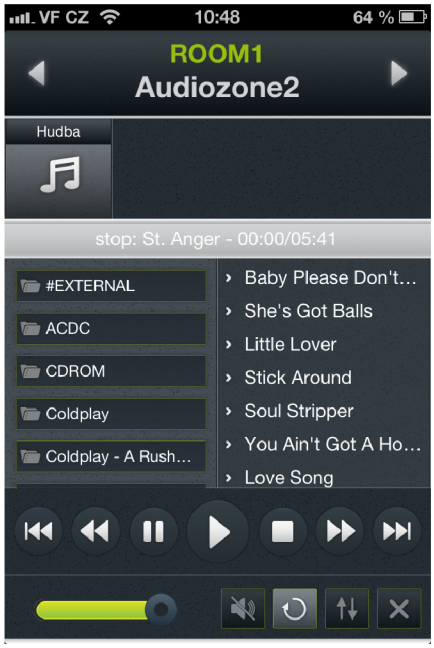
Obrázek B.4: Rychlé menu – kamery.

Obrázek B.5: Rychlé menu – Miele.

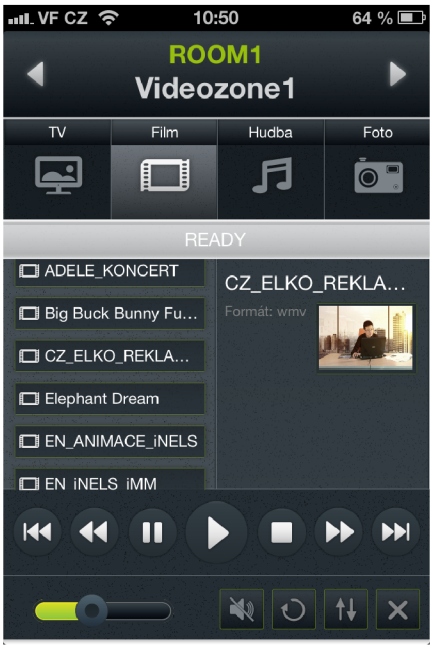
Obrázek B.6: Rychlé menu – klimatizace.



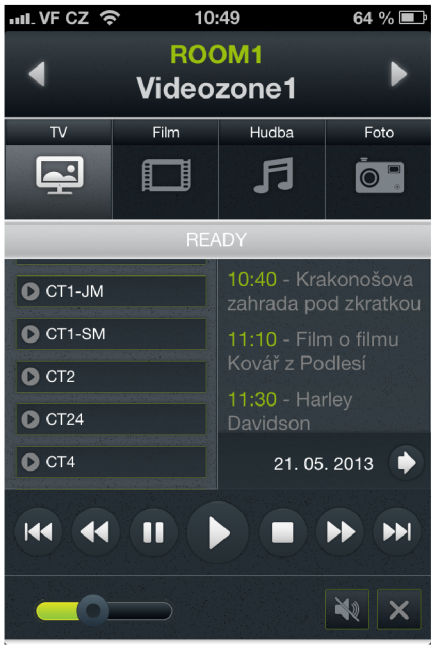
Obrázek B.7: Kamera.



Obrázek B.8: Multimedia – hudba.



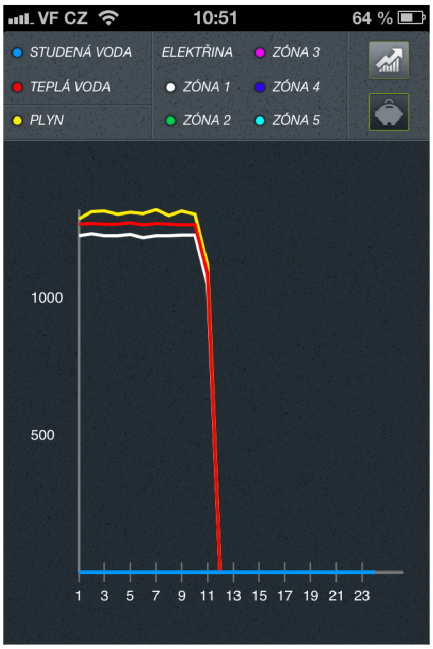
Obrázek B.9: Multimedia – filmy.



Obrázek B.10: Multimedia – TV.

Obrázek B.11: Měření energií.

Obrázek B.12: Měření energií – přehled.



Obrázek B.13: Měření energií – graf.

Obrázek B.14: Měření energií – volby grafu.