TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zpracování a vizualizace dat analyzátorů v OS Android

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jan Moravec**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Processing and visualisation of analyzer data in Android OS

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Jan Moravec**
Supervisor: Ing. Jan Kraus, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Moravec**
Osobní číslo:	**M14000053**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Zpracování a vizualizace dat analyzátorů v OS Android**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s problematikou návrhu aplikací pro zvolená mobilní zařízení s OS Android, vyberte vhodné nástroje pro vývoj vlastní aplikace pro záznam a zobrazení hodnot měření.
2. Seznamte se s formáty odečítaní měřených dat z paměti elektroměrů a analyzátorů kvality SMx 2. generace a se způsoby jejich archivace v souborech CEA.
3. Navrhněte s využitím stávajících knihoven a vytvořte nebo doplňte aplikaci pro nastavení, odečet a vyhodnocení aktuálních a archivních dat z výše uvedených analyzátorů.
4. Na vybraných zařízeních a jejich různých konfiguracích ověřte správnou funkci aplikace.
5. Uveďte konkrétní možnosti dalšího rozvoje systému a shrňte možnosti a omezení jeho využití v praxi.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **LACKO, L'uboslav. Vývoj aplikací pro Android. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.**

[2] **DEITEL, Paul J. Android 6 for programmers: an app-driven approach, 3/E. Third edition. ISBN 9780134289366.**

[3] **KRAUS, Jan a Martin BLÍŽKOVSKÝ. KMB SYSTEMS, S.R.O. Uživatelská příručka aplikace ENVIS v. 1.2 [online]. 2015. [cit. 2015-10-08]. 1.2. Dostupné z: http://www.kmb.cz/**

[4] **Eisler, David. Aplikace pro konfiguraci a správu dat měřicích přístrojů v prostředí Android. Liberec, 2016. Bakalářská práce. Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci**

[5] **Smolík, Adam. Aplikace pro zpracování měřených dat pro mobilní platformu Android. Liberec, 2016. Diplomová práce. Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci**

Vedoucí bakalářské práce: **Ing. Jan Kraus, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

Technická univerzita v Liberci
L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14. 5. 2017

Podpis: Jan Moravec

Abstrakt

Cílem této bakalářské práce je vytvořit aplikaci pro operační systém Android. Aplikace by měla komunikovat a pracovat s daty z elektroměrů a analyzátorů kvality. Úvodní část práce pojednává o problematice tvorby aplikací pro platformu Androidu. Jsou stručně popsána měřící zařízení, s kterými má aplikace komunikovat, a použitý komunikační protokol. Také jsou vysvětleny použité třídy a datové struktury pro reprezentaci konfiguračních a archivních dat. Aplikace je programována v jazyce Java s využitím MVVM návrhového vzoru. Hlavní funkcí aplikace je stahování a reprezentování konfiguračních dat z měřících zařízení a také modifikování a nahrávání těchto dat zpět do zařízení.

Klíčová slova

programování android aplikací, Java, vývoj grafického rozhraní, vizualizace dat, analyzátory kvality a elektroměry

Abstract

The aim of this bachelor thesis is to create an application for the Android operating system. Application should communicate and work with data from electricity meters and quality analyzers. The first part of the thesis deals with the issue of application development for the Android platform. This thesis briefly describes the measuring devices which the application should communicate with and communication protocol used for application. This thesis also explains which classes and data structures have been used to represent configuration and archive data. The application is programmed in Java using the MVVM design pattern. The main function of the application is to download and represent the configuration data from the measuring devices and also to modify and record the data back to the device.

Keywords

android application programming, Java, development of graphical interface, data visualization, power quality analyzers and electrometers

Obsah

Seznam zkratek

SDK	Software Development Kit, sada vývojových nástrojů
MVVM	Model-View-ViewModel, návrhový vzor
XML	Extensible Markup Language, obecný značkovací jazyk
RS-232	Standard sériové komunikace
RS-485	Standard sériové komunikace
USB	Universal Serial Bus, univerzální sériová sběrnice
LCD	Liquid crystal display, displej z tekutých krystalů
LED	Light-Emitting Diode, polovodičová elektronická součástka vyřazující světlo
OS	Operating system, operační systém
CRC	Cyclic redundancy check, cyklický redundantní součet
PC	Personal computer, osobní počítač
IP	Internet protocol, základní protokol síťové vrstvy
UDP	User Datagram Protocol, uživatelský datagramový protokol
TCP	Transmission Control Protocol, přenosový řídící protokol
FPS	Frames per seconds, snímková frekvence
RAM	Random access memory, paměť s libovolným přístupem
NTP	Network Time Protocol, síťový protokol pro synchronizaci času
DHCP	Dynamic Host Configuration Protocol, protokol pro automatické přidělování síťových údajů
JDK	Java Development Kit, nástroje pro vývoj v jazyce Java
ZIP	ZIP je formát pro komprimaci souborů

Seznam ilustrací

Úvod

V současnosti jsou velice rozšířené mobilní zařízení jako chytré telefony a tablety. Díky své přenositelnosti, dotykovým ovládáním a zvyšujícímu se výkonu mohou být pro určité aplikace vhodnější než počítače. Mobilní aplikace se využívají, jak v pracovních odvětvích, tak za účelem zábavy. Z těchto důvodů jsou mobilní aplikace dnes velice žádané.

Cílem této práce bylo vytvořit právě aplikaci pro operační systém Android. Aplikace by měla komunikovat s měřícími zařízeními řady SMx 2. generace od firmy KMB systems. Pomocí aplikace by mělo být možné stahovat, číst a měnit jednotlivé konfigurační parametry a také umožňovat pracovat s aktuálními a archivními daty, přesněji vyhodnocovat a zobrazovat aktuální data a umožnit stahování archivních dat. Dalším cílem je také to, aby aplikace správně fungovala při komunikaci s různými zařízeními ze zmiňované řady, které budou mít odlišnou konfiguraci. K předloze sloužila desktopová aplikace ENVIS.Daq, cílová aplikace by měla mít stejné funkce, ale měla by být jednodušší a přizpůsobená operačnímu systému Android.

Na začátku teoretické části jsou popsáné práce, které mají podobné zadání, zejména diplomové práce Ing. Adama Smolíka, z které tato práce vychází. V dalších částech práce je krátce popsán operační systém Android a výběr nástrojů pro tvorbu aplikací pro tento operační systém. Stručně jsou popsána zařízení, s kterými by aplikace měla komunikovat. Dále jsou vysvětleny použité návrhové vzory, struktura CEA souboru pro ukládání archivních dat a použité datové struktury.

Praktická část popisuje návrh a implementaci samotné aplikace, jak ze stránky grafické, tak logické. Je dělena do kapitol podle jednotlivých grafických a funkčních částí. V závěrečné části práce je uvedeno, jakými testy byla aplikace podrobena a celkové vyhodnocení práce.

1 Teoretická část

1.1 Podobné řešení

Práce navazuje na diplomovou práci Ing. Adama Smolíka [1]. V původní aplikaci byla vytvořena hlavní aktivita pro přidávání a editování položek představující jednotlivé zařízení. Dále byla vytvořena aktivita zobrazující aktuální data měřícího zařízení a aktivita pro stahování archivních dat, která ještě nebyla kompletně funkční, a byla předělána. Více informací o provedených změnách je uvedeno v kapitole 2.5. Původní aplikace kompletně postrádala rozhraní pro správu nastaveni. Z toho důvodu je tato práce zaměřená hlavně na vytvoření aplikace, pomocí které bude možné stahovat, číst a měnit konfiguraci zařízení. V původní aplikaci také byla knihovna poskytující funkce ke komunikaci se zařízením a práci se stáhnutými daty. Knihovna sice usnadnila práci na této bakalářské práci, ale obsahovala chyby, a některé funkce bylo nutné opravit a doplnit.

Podobnou práci také řešil Bc. David Eisler [2]. Jeho práce se také zabývala vytvořením aplikace pro mobilní zařízení, která komunikuje s elektroměry a analyzátory kvality energie. Pro svou aplikaci použil framework Xamarin, který poskytuje vývoj pro více mobilních operačních systémů [3]. Jeho výsledná aplikace poskytovala funkční stahování a reprezentování několika kategorií konfiguračních dat pro systém Android.

1.2 Operační systém Android

Android je operační systém pro mobilní platformu. V současnosti (duben 2017) má Android největší podíl na trhu mobilních zařízení. Celkový podíl OS Android činí 81,7 % (konec roku 2016). Druhý největší podíl 17,9 % patří operačnímu systému iOS. Na třetím místě jsou operační systémy od společnosti Microsoft, které mají 0,3 % na trhu. Ostatní operační systémy zastupují pouze 0,1 % na trhu mobilních operačních systémů [4]. Právě kvůli vysoké rozšířenosti systému Androidu na trhu byla aplikace vyvíjena pro tento operační systém.

Mezi potřebné nástroje k programování aplikací pro platformu Androidu jsou takzvané SDK a JDK. JDK je soubor nástrojů určený pro vývoj aplikací v jazyce Java [5].

SDK je sada nástrojů pro Android, v této sadě jsou knihovny pro programování, debugování a také obrazy systémů, ukázkové aplikace a dokumentace.

Každé zařízení používající systém Android využívá určitou verzi API, to je rozhraní, které definuje, jaké funkce zařízení poskytuje. Každá aplikace pak má určené jakou minimální verzi API podporuje. Pro aplikaci byla zvolena minimální podporovaná verze API 15, což odpovídá verzi Androidu 4.0.3 (Ice Cream Sandwich MR1) [6]. Aplikace přesto využívá knihovny z novějších API, to je umožněno pomocí Support knihovny, která zajišťuje zpětnou kompatibilitu pro starší zařízení [7]. Jako minimální podporovaná verze API byla zvolena úroveň 15 z toho důvodu, protože některé použité knihovny v kódu aplikace nejsou zpětně kompatibilní. Přesto by aplikace měla být spustitelná pro 98 % současně používaných Android zařízení (duben 2017). [8].

K vývoji aplikace bylo použito vývojové prostředí Android Studio. Alternativou by mohla být například prostředí Eclipse nebo NetBeans. Výhodou Android Studia je, že je to oficiální vývojové prostředí [9]. To znamená, že Android Studio je jako první aktualizováno a přizpůsobeno novým verzím a funkcím systému Android.

1.2.1 Aktivita

Základním stavebním prvkem Android aplikací jsou aktivity. Vlastní aktivita je vytvořena děděním od třídy Activity. Měla by definovat prezenční logiku jedné obrazovky nebo jinak řečeno pohledu (view), který je popsaný v systému Android pomocí XML souboru. V aplikaci může být takových aktivit několik. Všechny použité aktivity v aplikaci jsou uvedeny v souboru „AndroidManifest.xml“, který je součástí každé Android aplikace[10]. V tomto souboru je také definované, jaká aktivita se bude spouštět po zapnutí aplikace jako první.

Aktivita má několik stavů, které jsou popsané životním cyklem aktivity [11]. Základní jsou čtyři stavy. První z nich je, když je aktivita viditelná, běží a reaguje na vstup od uživatele. Druhý stavem je, když je aktivita pozastavená, už není na popředí a neraguje na vstup od uživatele, ale je stále viditelná. Aktivita je ve třetím stavu, pokud je zastavena, to znamená, že není viditelná, ale je její stav je stále uložen v paměti. Poslední čtvrtý

stav značí, když už aktivita neexistuje, byla kompletně ukončena a nenachází se v paměti. Životní cyklus aktivity je definován pomocí callbacků, ty jsou volány, když dochází k přechodu mezi jednotlivými stavy. Pomocí callbacků si uživatel může definovat vlastní chování aktivity při změně jakéhokoliv stavu [11].

1.2.2 Fragment

Fragment je podobný aktivitě, jedná se o jakousi podaktivitu. Vlastní fragment je vytvořen při dědění od třídy Fragment [12]. Má vlastní vzhled definovaný stejně jako aktivita pomocí XML souboru. Může být včleněn přímo do XML souboru definující vzhled aktivity, nebo zdrojového kódu aktivity programově. Fragmentů může jedna aktivita obsahovat více a také lze fragmenty použít ve více různých aktivitách. Fragment má podobně jako aktivita svůj vlastní životní cyklus, navíc je vždy ovlivněn životním cyklem nadřazené aktivity.

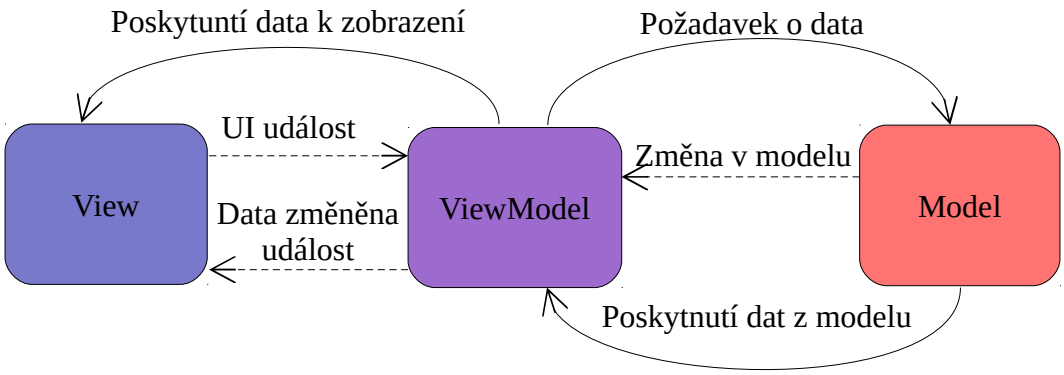
1.2.3 AsyncTask

Třída AsyncTask neboli asynchronní úloha poskytuje jednoduché řešení pro provedení kódu mimo hlavní vlákno. Asynchronní úloha by měla být použita jen na rychlé operace trvající řádově několik sekund [13]. Úloha je prováděna ve čtyřech krocích. Nejprve je provedena metoda „onPreExecute“, která, jak název naznačuje, je spuštěna ještě před samotnou úlohou. Potom je spuštěna metoda „doInBackground“, ve které by měl být umístěný kód trvající delší dobu a ten už bude běžet na pozadí. Dále je možné využít metodu „onProgressUpdate“, pomocí které je možné vracet do hlavního vlákna postup úlohy. Poslední krok představuje metoda „onPostExecute“, ta slouží k vrácení výsledku zpět do hlavního vlákna po dokončení metody „doInBackground“.

1.3 Návrhový vzor MVVM

Aby byly oddělené jednotlivé části logiky aplikace, byl využit návrhový vzor Model-View-ViewModel. Výhodou tohoto návrhové vzoru je také to, že se aplikace lépe testuje, rozšiřuje a kód je celkově čitelnější [14]. Pro efektivní využití tohoto vzoru se používá data binding. Vzor rozděluje logiku do tří vrstev. První z nich je model, ten

popisuje data, s kterými aplikace pracuje. Model by neměl mít žádnou vazbu s grafickým rozhraním. V této aplikaci model představuje zejména KMB knihovna, kde jsou funkce pro komunikace s měřícím zařízením a datové struktury pro ukládání dat.



Ilustrace 1: Diagram návrhového vzoru MVVM

View reprezentuje uživatelské rozhraní. Může to být například okno aplikace nebo určitý ovládací prvek. V Android aplikaci je view tvořen pomocí XML jazyku, který určuje rozvržení grafických prvků.

Viewmodel představuje vrstvu mezi modelem a view, která drží stav aplikace a pracuje s oběma těmito vrstvami. Viewmodel používá data a funkce z modelu. Data ve viewmodelu jsou propojený s view pomocí bindingu. Pokud nastane změna v uživatelském prostředí, projeví se v viewmodel. Pokud nastane změna v viewmodelu, tak se zase promítne do uživatelského rozhraní.

Konkrétně je viewmodel implementován pomocí aktivity nebo fragmentu. Tato aktivita nebo fragment jsou potomkem třídy viewmodel, která má informace o jejich životním cyklu. ViewModel je ukončen (odebrán z paměti) až ve chvíli, kdy je ukončen životní cyklus aktivity či fragementu. Aby bylo možné tento návrhový vzor jednodušejí využít společně s technikou data binding, byla implementována knihovna Android-ViewModelBinding, která je volně dostupná ke stažení z github repozitáře [15].

1.4 Data binding

Data binding je mechanismus, který synchronizuje data mezi view a viewmodel vrstvami. Tento mechanismus je založen na událostech, pokud je hodnota jednoho objektu změněna, spustí se událost a hodnota druhého objektu se aktualizuje [16].

Princip lze jednoduše demonstrovat na příkladu. Existuje aktivita, která má proměnnou typu řetězec a grafické rozhraní s jedním textovým polem. Proměnná je spojená (bind) s textovým polem pomocí data bindingu. Pokud uživatel změní obsah textového pole, změna se projeví i v proměnné. Toto funguje i obráceně, pokud logika aplikace změní hodnotu této proměnné, změna se promítne v grafickém rozhraní.

Nejjednodušší cestou, jak použít data binding v Androidu je pomocí ObservableField [17]. To je tzv. wrapper, kam lze vložit jakýkoliv druh objektu a zároveň se tato třída stará o synchronizaci.

1.5 Měřící zařízení

Zařízení od společnosti KMB systems slouží pro měření spotřeby elektrické energie, sledování, záznam aktuálních veličin a trvalé sledování kvality elektrické energie v různých úrovních napájecí sítě. Také poskytuje možnost archivace dat v paměti zařízení [18].

Zařízení jsou buď jen s jednoduchými LED indikátory, nebo mají LED, LCD displeje. S přístroji je možné komunikovat přes rozhraní Ethernet, USB nebo sériovou linku (RS-232/RS-485) [18]. Pokud je zařízení vybavené rozhraním Ethernet, tak má standardně zařízení zabudovaný nativní webserver, na kterém lze sledovat všechny hlavní měřené hodnoty, čítače a nastavení přístroje. Pro zobrazení většího množství informací je možné zařízení připojit k PC a využít aplikaci Envis. Nyní lze také s využitím wifi se zařízeními komunikovat pomocí mobilní aplikace, kterou popisuje tato práce.

1.6 Protokol KMB Long

KMB Long je proprietární aplikační protokol. Pomocí tohoto protokolu je možné komunikovat s měřícím zařízením přes sériovou linku a Ethernet. Všechny přenášené data jsou ve tvaru big-endian. Protokol používá osm datových bitů s jedním stop bitem. Nepoužívá paritu, místo toho se k detekci chyb přenosu používá cyklický redundantní součet. Měřící zařízení funguje jako slave. PC nebo mobilní zařízení je v roli master a posílá příkazy či dotazy , na které slave (měřící zařízení) odesílá odpovědi [19].

N – celková délka zprávy

Bajt:	1	2,3	4	5 až N − 2	N − 1, N
Význam:	Adresa zařízení	Délka zprávy	Typ zprávy	Tělo zprávy	CRC

Ilustrace 2: KMB Long zpráva

Všechny KMB Long zprávy mají stejný formát. Jak lze vidět na obrázku první bajt zprávy určuje adresu zařízení, druhý a třetí bajt délku zprávy a čtvrtý bajt určuje typ zprávy. Potom následuje tělo, které má různý počet bajtů podle typu zprávy. Na konci jsou 2 bajty pro kontrolní součet CRC [19].

Alternativním protokolem, který lze použít ke komunikaci se zařízením přes rozhraní Ethernet je protokol Modbus TCP. Ten není aplikací podporován kvůli jeho malé maximální velikosti rámce, která je pouze 260 bajtů [20].

1.7 CEA soubor

Archivní data lze ukládat na paměťové médium jako tzv. CEA soubory. Jedná se o strukturu několika složek a souborů, komprimovaných pomocí zip komprese. Aplikace popisující tato práce umí archivní data ukládat pomocí CEA souborů. Tyto vytvořené soubory je pak možné nahrát do desktopové aplikace ENVIS a dále s nimi pracovat.

Jak lze vidět na ilustraci č. 10 na nejvyšší úrovni struktury se nachází složky s názvy Info a UNI. Složka s názvem Info obsahuje dva soubory. První z nich je XML

soubor TreeList , který obsahuje informace o zařízení. Druhý je textový soubor VersionHistory, ten obsahuje pouze jeden řádek textu. Na řádku je nejdříve název a verze aplikace, která vytvořila daný CEA soubor, dále datum a čas vytvoření a typ akce.

Ve složce UNI se nachází dvě složky. Jedna z nich má název podle identifikátoru záznamu a druhá ArchDefs. Ve složce s identifikátorem záznamu se nachází různý počet složek. Ten je určen tím, kolik bylo staženo typů archivních dat. Každá složka má název podle identifikátoru, který je určen podle druhu archivu. V každé z těchto složek se nachází dva binární soubory s hlavičkou a daty jednotlivých záznamů. Hlavička obsahuje číslo s velikostí hlavičky, čas prvního a posledního záznamu, periodu, která určuje, po jaké době jsou vytvářeny záznamy a časové indexy určující mezery mezi záznamy.

Data s jednotlivými záznamy jsou uloženy v souboru s názvem podle času posledního záznamu. Jsou uložena sekvenčně v jednom souboru pro daný typ archivu.Ve složce ArchDefs se nachází xml soubory popisující záznamy jednotlivých druhů archivů. Názvy jsou vytvořeny pomocí identifikátoru typu archivu a CRC hašovací funkce [1].

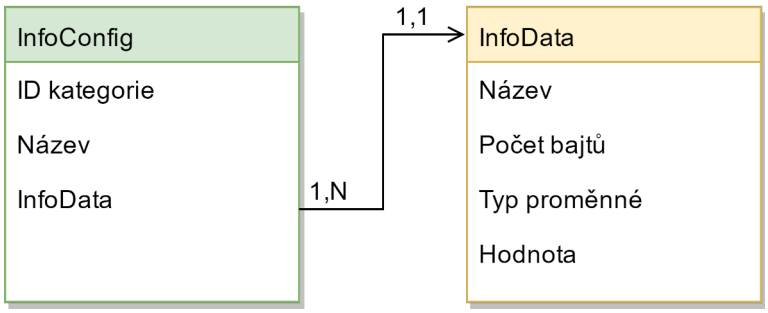
1.8 KMB Knihovna

Aplikace je rozdělena na dva celky, na knihovnu a aplikaci implementující tuto knihovnu v prostředí Android. Obě tyto části byli součástí původní práce, tato práce tyto dva celky dále rozšiřuje. V následujících podkapitolách jsou popsané třídy z této knihovny, které byly využity v aplikaci.

1.8.1 UniversalConfigs struktura

Třída UniversalConfigs obsahuje strukturu pro ukládání stažených konfiguračních dat ze zařízení a metody pro práci s těmito daty. Daty jsou myšleny například informace a parametry nastavení zařízení nebo jednotlivých archivů. Třída UniversalConfigs obsahuje arraylist s datovým typem InfoConfig, kde jsou uloženy konfigurační data. InfoConfig a InfoData jsou vnitřní třídy UniversalConfigs třídy. InfoConfig určuje kategorii pomocí identifikátoru a názvu a také obsahuje list s InfoDaty. Třída InfoData obsahuje údaje o konkrétním konfiguračním parametru jako název, typ proměnné

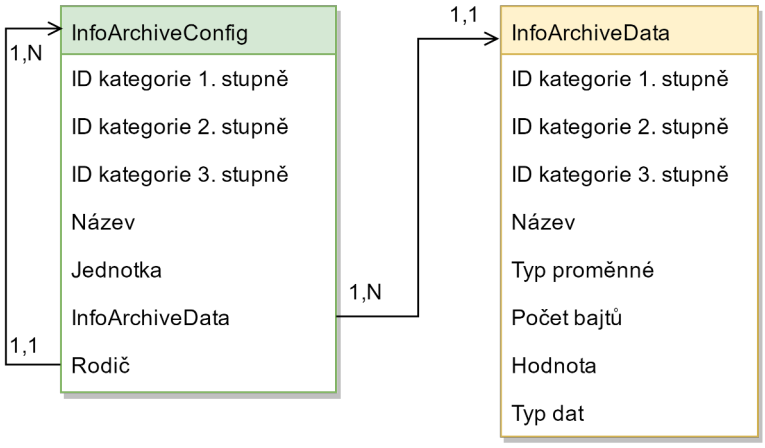
hodnoty, velikost proměnné a samotnou hodnotu. Tato třída také poskytuje metody pro práci s touto strukturou.



Ilustrace 3: Znázornění struktury pro konfigurační data

Mezi nejdůležitější metody patří metoda pro zpracování a načtení pole bajtů do datové struktury. Potom metoda pro získání a nastavení konkrétního parametru podle názvu a kategorie. Dále obsahuje metody pro práci s konfiguračními soubory, pro ukládání a nahrávání. Třída neslouží jen pro práci s konfiguračními daty, ale také pro práci s archivními daty. Archivní data se ukládají do jiné struktury. Struktura pro archivní data je popsána pomocí třídy InfoArchiveConfig, ta obsahuje až 3. stupně kategorií a může se na sebe odkazovat a obsahuje arraylist s datovým typem InfoArchiveData.

InfoArchiveData má 3 atributy určující kategorii, název, typ proměnné hodnoty, počet bajtů, číslo určující typ dat a samotnou hodnotu dat. Tyto dvě třídy jsou stejně jako u předchozí vnitřní a třída UniversalConfigs.



Ilustrace 4: Znázornění struktury pro archivní data

1.8.2 Třída DeviceManager

DeviceManager je třída reprezentující zařízení. Mezi jeho atributy patří URL a číslo portu. URL může být ve tvaru webové adresy nebo IP adresy. Dále má tato třída referenci na objekt třídy zmíněné v předchozí kapitole UniversalConfigs. Důležitou součástí této třídy je také metoda requestAndAnswer. Tato metoda slouží pro komunikaci se zařízením. Přijímá jako vstupní parametr zprávu v podobě pole bajtů. Předanou zprávu do zařízení odešle a čeká několik sekund na odpověď, pokud přijme odpověď je vrácena jako návratová hodnota metody. Může také vrátit zprávu s chybou a v případě, kdy dojde k chybě se spojením, je vyhozena výjimka, kterou musí ošetřit nadřazená metoda používající tuto metodu [1].

2 Praktická část

2.1 Hlavní aktivita

Hlavní aktivita už byla vytvořená v původní aplikaci. Došlo však k několika změnám, pro porovnání slouží ilustrace č. 16, 17. Každá položka v seznamu představuje jedno zařízení. Jak lze poznat z obrázků byli přidány ikony signalizující dostupnost zařízení. Každá tato ikona má 3 stavy, červená dioda značí nedostupnost zařízení. Pokud probíhá zjišťování, zdali je zařízení dostupné, je na místě ikony zobrazena animace vyjadřující průběh a zelená dioda signalizuje, že je zařízení dostupné. To, jestli je zařízení dostupné, je zjištěno v jiném vlákně pomocí paralelního odeslání zpráv s požadavkem o identifikační data na každé zařízení v seznamu. Pokud se od zařízení vrátí validní odpověď, zařízení je považované za dostupné a změní se barva ikony na zelenou. Tato operace je provedena hned po spuštění aplikace nebo lze využít tlačítko v pravém horním rohu aplikaci, které aktualizuje dostupnost všech zařízení v seznamu.

2.1.1 Lokátor

Další přidanou funkcí je lokátor, který vyhledává dostupná zařízení na lokální síti. Pro spuštění hledání zařízení na lokální síti slouží tlačítko v levém dolním rohu viditelné na ilustraci č. 17. Lokátor je spuštěn na samostatném vlákně, aby neblokoval hlavní vlákno. Běh lokátoru je v grafickém rozhraní zobrazen pomocí jednoduché animace. Tato animace překrývá původní tlačítko, aby nebylo možné zbytečně zahlcovat síť. Po dokončení operací lokátoru je tlačítko opět zobrazeno a je možné ho znovu použít.

Hledání zařízení funguje pomocí odeslání UDP broadcastu s cílovým síťovým portem 23, na kterém poslouchají zařízení a jako odpověď odesílají své identifikační data. Původně zařízení odesílali odpověď na port 23, který patří mezi vyhrazené porty. V tom byl původně problém, protože v systému Android jsou nutná k poslouchání na vyhrazených portech administrátorská práva, které běžný uživatel nemá. Tento problém byl vyřešen v novém firmwaru zařízení. S novým firmwarem měřící zařízení posílají

zprávy i na port 2323, který už nepatří mezi porty vyhrazené. Lokátor tedy po odeslání broadcastu poslouchá na portu 2323 po dobu 10 sekund a přijímá zprávy od zařízení. Přijatou zprávu je z pole bajtů dekódována, a pokud je validní je předána do hlavního vlákna.

Na hlavním vlákně je ověřeno, zdali se nalezené zařízení už v databázi nenachází. Pokud zařízení není v databázi, zobrazí se dialogové okno tázající se uživatele, zdali chce přidat nalezené zařízení viditelný na ilustraci č. 14. Pokud uživatel souhlasí s přidání zařízení, tak je abstraktní zařízení ze získaných parametrů vytvořeno a přidáno do databáze. V dialogu je také možné zaškrtnout možnost, aby se při příštím vyhledávání tento dialog nezobrazoval a zařízení se automaticky přidalo do seznamu. Tato informace je uložena jako boolean hodnota do Android Preferences, které uloží informaci do interní paměti mobilního zařízení [21].

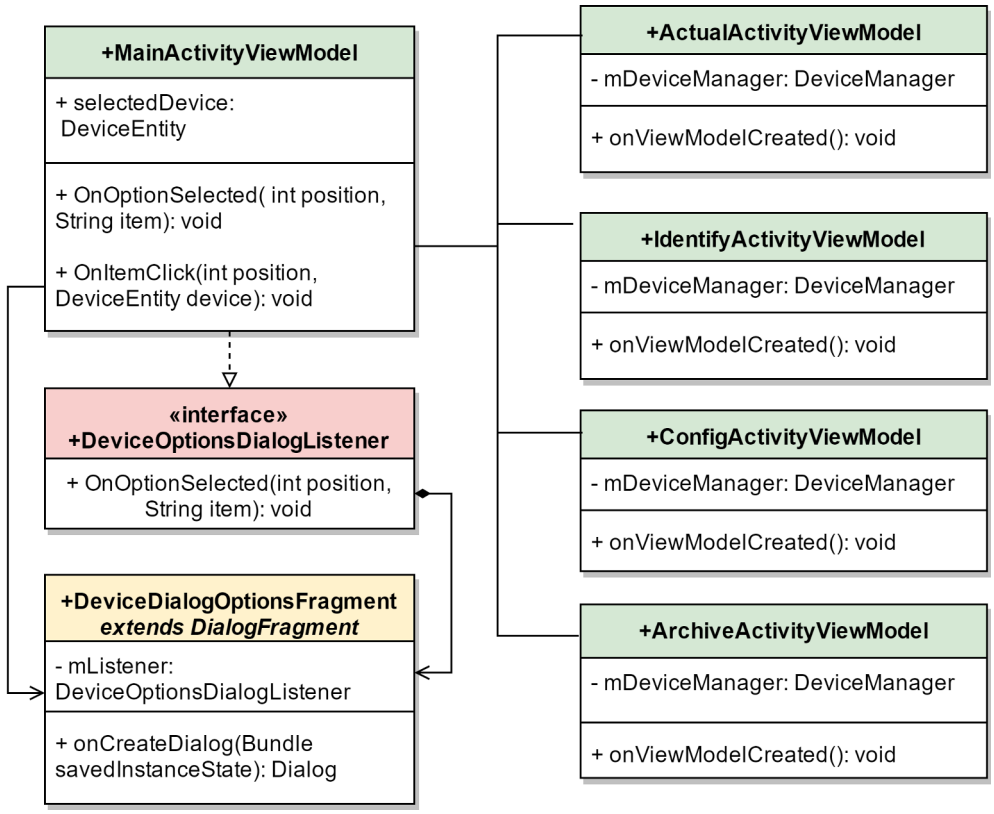
Lokátor tak usnadňuje uživateli práci před přidáváním všech zařízení manuálním způsobem pomocí zadávání IP adresy a portu. Vytvořená abstrakce zařízení má přiřazený název podle typu skutečného zařízení, který si může případně uživatel změnit.

2.2 Navigace

V původní aplikaci se po vybrání zařízení uživatel dostal do aktivity s detailem zařízení, kde bylo k dispozici stahování archivních dat a dále bylo možné se přesunout do aktivity s aktuálními daty. Jak vypadala aktivita s detailem zařízení je možné vidět v příloze na ilustraci č. 18. Z toho důvodu, že byla aplikace rozšířena o práci s konfiguračními a identifikačními daty, bylo potřeba změnit původní grafické rozhraní, aby se dalo mezi jednotlivými aktivitami rychle přesouvat a orientovat.

V hlavní aktivitě se nyní po výběru zařízení zobrazí dialog s možnostmi, pomocí kterých je možné se přesunout do dalších částí aplikace. Dialog je zobrazen na ilustraci č. 19. Každá položka představuje samostatnou aktivitu. Po vybrání položky se uživatel dostane do dalších aktivit, které jsou popsané v dalších kapitolách.

2.3 Hlavní logika aplikace



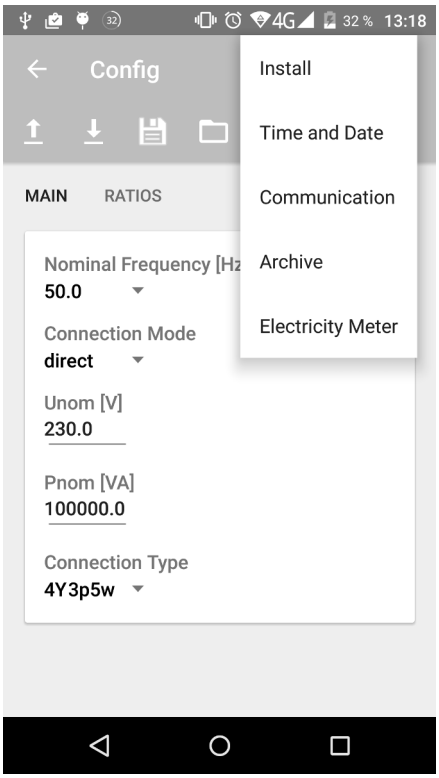
Ilustrace 5: UML diagram hlavní logiky aplikace

Na UML diagramu je zobrazena hlavní struktura viewmodel vrstvy aplikace. Hlavní aktivita implementuje rozhraní DeviceOptionsDialogListener. V hlavní aktivitě se nachází seznam zařízení, po vybrání jednoho ze zařízení se vykoná metoda OnItemClick. Tato metoda vytvoří dialog, kterému poskytne referenci na objekt implementující výše zmíněné rozhraní, které implementuje právě třída MainActivityViewModel. Dialog je možné vidět na ilustraci č. 19. Po vybrání jedné z položek se zavolá metoda OnOptionSelected na objektu mListner s argumentem indexu na vybranou položku. Hlavní aktivita podle získaného indexu spustí jednu ze 4 aktivit zobrazených v diagramu. Nově spuštěné aktivitě se předá proměnná selectedDivice třídy DeviceEntity, která poskytuje

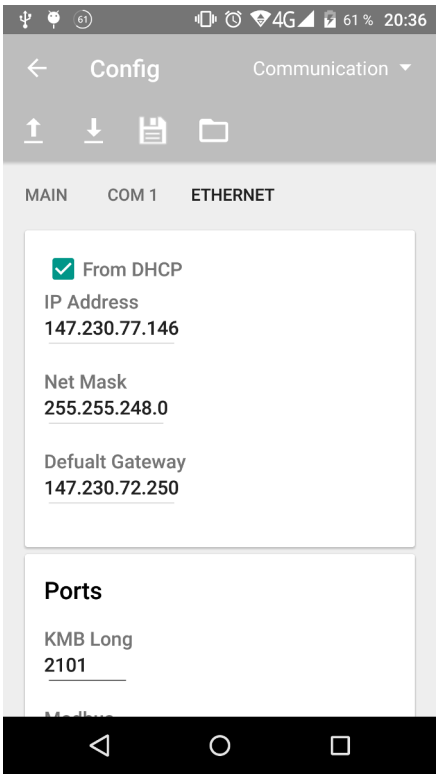
IP adresu a port zařízení, pomocí toho je v nové aktivitě vytvořen objekt třídy Device-Manager, který je v dalších aktivitách potřebný.

2.4 Konfigurace

Tato část aplikace slouží pro práci s konfigurací zařízení. Grafické rozhraní se skládá z dvou celků. Jedním celkem je panel nástrojů vytvořený pomocí AppBarLayout a Toolbar grafických komponent. Na levé straně je tlačítko s ikonou šipky pro navrácení do hlavní aktivity a za ní se nachází název aktivity. Na pravé straně panelu je umístěn Spinner, pomocí kterého si lze vybírat mezi jednotlivými kategoriemi nastavení. Na dolní části panelu jsou ikony, které popisují jednotlivé funkce, které může uživatel využívat. Jedná se o stahování konfigurace ze zařízení, nahrávání konfigurace do zařízení a ukládání konfigurace do souboru a nahrávání konfigurace ze souborů.



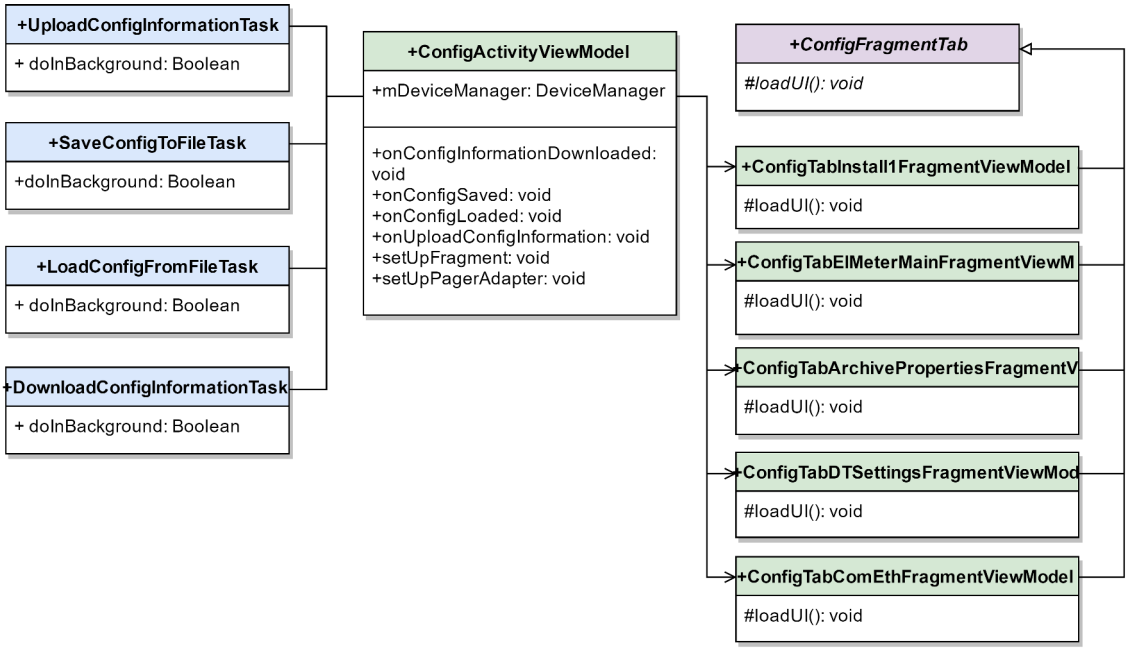
Ilustrace 6: Zobrazením konfigurace s nabídkou kategorií



Ilustrace 7: Zobrazením konfigurace – komunikační parametry

Druhá část je obsahová, ta se dynamicky mění podle vybrané kategorie nastavení spinneru v panelu nástrojů. Jednotlivé kategorie nastavení se ještě dělí na další podkategorie. Pro každou podkategorii existuje fragment. Jak lze vidět na obrázku, pro kategorii Install jsou dvě podkategorie Main a Ratios. Toho je docíleno pomocí nástroje Viewpager. Mezi podkategoriemi se lze přesouvat pomocí gest, přejetí prstem z jedné strany displeje na druhou a taky pomocí menu na horní straně obsahové části obrazovky. Jednotlivé skupiny parametrů jsou ohraničené pomocí grafické komponenty CardView. Parametry jsou zobrazeny pod sebou, aby nebyly problémy s délkou textů a vešly se na obrazovku u různých mobilních zařízení. Názvy parametrů jsou vytvořeny pomocí TextBoxu a jednotlivé hodnoty pak nejčastěji pomocí Spinnerů, EditTexů a CheckBoxů.

2.4.1 Logická struktura



Ilustrace 8: UML diagram konfigurace

Logická struktura konfigurace je zjednodušeně znázorněna pomocí UML diagramu na ilustraci č. 8. Hlavní třídou je ConfigActivityViewModel. Jedním z úkolů této třídy je reagovat na požadavky uživatele. Pokud je registrováno kliknutí na jednu z ikon umístěných na panelu nástrojů, spustí se příslušná operace v podobě úlohy. Úlohy jsou

s viewmodelem spojeny přes rozhraní, které nejsou kvůli úspornosti v diagramu zobrazeny. Úlohy jsou potomkem třídy AsyncTask a běží v jiném vlákně. Potom, co je úloha ukončena, se výsledek vrací pomocí rozhraní zpět do viewmodelu a aktualizuje se uživatelské grafické rozhraní.

Konkrétní obsah je zobrazen pomocí fragmentů. Každý fragment má svůj view neboli layout a viewmodel. To, jaký fragment je aktuálně zobrazený, je závislé na výběru v spinneru , tím je určena kategorie konfigurace. Každá kategorie má několik fragmentů, mezi kterými se může uživateli pohybovat. Každý Fragment představuje jednu podkategorii. V jednom okamžiku jsou nahrány maximálně dva fragmenty. To je nastaveno, kvůli tomu aby byl pohyb mezi fragmenty plynulý. Není jich načítáno více proto, aby nedošlo k vytížení prostředků.

Všechny fragmenty v diagramu rozšiřují abstraktní třídu ConfigFragmentTab a implementují její metodu loadUI(). Tato metoda je volána ve chvíli, kdy je fragment připojen k aktivitě. Pro zjednodušení práce při nastavováni grafických prvků je použita statická třída ConfigTabFragmentTools. Ta poskytuje metody pro vytvoření adaptérů a listenerů, které jsou ve fragmentech často používány. Některé konfigurační parametry je možné měnit a některé slouží pouze pro zobrazení. Pokud uživatel změní nějaký parametr, je provedena validace. Pokud je zadaný vstup validní, je zapsán do konfigurační struktury UniversalConfigs, na kterou má referenci instance třídy DeviceManager, ke které mají přístup všechny fragmenty, protože jsou všechny potomkem třídy ConfigFragmentTab.

2.4.2 Instalační parametry

Nastavení instalačních parametrů zajišťují dva fragmenty ConfigTabInstall1FragmentViewModel a ConfigTabInstall2FragmentViewModel. Grafické rozhraní je možné vidět na ilustraci č. 14 v příloze. První zmíněný fragment umožňuje změnu nominálního napětí, výkonu a frekvence a také typ a mód připojení. Tyto parametry jsou vytvořeny pomocí EditTextů a Spinnerů. Typ připojení je odlišný podle druhu měřícího zařízení, správný obsah Spinneru je vytvořen za běhu při vytvoření fragmentu. Druhý fragment

slouží pouze pro zobrazení poměrů a násobitelům, které jsou realizované pomocí TextView komponent odlišné stylem pro neměnné vlastnosti, které mají šedou barvou.

2.4.3 Čas a datum

Pro práci s měřícími zařízeními je důležité, aby měla správně nastavený čas a datum. První fragment je zobrazen na ilustraci č. 15, slouží pro zobrazení aktuálního času a data zařízení. První položkou je datum, pod ním čas a za ním následuje rozdíl mezi časem na měřícím a mobilním zařízením. K získání aktuálního času ze zařízení slouží speciální zpráva. Tato zpráva je poslána v jiném vlákně pomocí úlohy AsyncTask. Přijatý čas je ve formě milisekund od data 1. 1. 2000. Pokud je čas od zařízení získán, tak je převeden na objekt třídy Calendar, který je vrácen do fragmentu, ze kterého byla úloha spuštěna.

Po přijetí objektu třídy Calendar už je možné zobrazit datum, čas a rozdíl mezi časem na měřícím a mobilním zařízení. Pro aktualizaci času slouží Timer, který každou sekundu aktualizuje TextView s časem a datem. K aktualizaci těchto hodnot ještě slouží tlačítko s textem „Resfresh“.

Další fragment slouží k nastavení času a data. Pro nastavení data a času slouží další speciální zpráva, která je odesílána v jiném vlákně. Čas a datum lze nastavit buď manuálně nebo podle času na mobilním telefonu. Při kliknutí na nastavení data se zobrazí Android DatePickerDialog. Při kliknutí na nastavení času je pak zobrazen TimePickerWithSeconds, volně dostupný ke stáhnutí a použití, který umožňuje oproti klasickému Android TimePicker dialogu výběr vteřin [22].

Poslední, třetí fragment slouží pro nastavení synchronizace času, časové zóny a letního času. Nastavení letního času je řešeno jednoduše pomocí jednoho CheckBoxu, nastavení časové zóny je realizováno pomocí Spinneru. Zobrazen je také rozdíl mezi časovou zónou na mobilním zařízení a měřícím zařízeni. Synchronizace času je možné nastavit pomocí Spinneru, který nabízí několik možností. Při zvolení některých možností Spinneru se dynamicky zobrazí další grafické komponenty. Například při zvolení synchronizace přes NTP server, je zobrazen EditTextu pro zadání IP adresy serveru. Ve výchozím stavu je synchronizace vypnuta.

2.4.4 Komunikační parametry

Pro nastavení komunikace slouží tři fragmenty. V prvním fragmentu je nastavení pouze pro adresu zařízení. V dalším fragmentu se dají nastavit parametry pro komunikaci přes sériový port. Při výběru protokolu je zároveň nastavena parita, data bity a stop bity, které se odvíjejí podle zvoleného protokolu, pokud se jedná o KMB nebo M-Bus protokol. Poslední fragment se týká komunikace přes Ethernet. V tomto fragmentu je možné nastavit síťové parametry, jako IP adresa, maska sítě a výchozí brána. Také je možno zvolit přidělování síťových parametrů od DHCP serveru, pokud je tato možnost zvolena, není možné parametry nastavovat ručně pomocí EditTextu.

Toho je docíleno pomocí listeneru, který deaktivuje příslušné EditTexty, pokud je přidělování síťových parametrů od DHCP serveru zapnuto. Nastavování položek ve tvaru IP adresy je kontrolováno po každém znaku, a pokud není adresa validní, je uživatel informován zprávou. Dále je možné v tomto fragmentu měnit čísla portů pro KMB Long, Modbus protokoly a webový server.

2.4.5 Archivy

Nastavení archivů se dělí na dva fragmenty. První z nich slouží pro nastavení vlastností archivu. Například název záznamu, začátek archivace dat, zdali má zařízení při zaplnění paměti začít přemazávat staré záznamy. Pokud je změněna hodnota intervalu archivace, je dynamicky změněn i čas, který ukazuje, jak dlouho bude možné archivovat záznamy, dokud se nezaplní celá paměť zařízení. Toho je docíleno pomocí implementace nástroje TextWatcher, který při každé změně EditTextu, provede přepočítání zbývajícího času podle zadané validní hodnoty. Druhý fragment slouží pro výběr veličin, které se mají archivovat. Je složen z desítek CheckBoxů pro všechny veličin, které mohou být archivovány.

2.4.6 Elektroměr

Nastavení elektroměru je realizováno pouze pomocí jednoho fragmentu. Nastavovat se dají parametry jako interval ukládání, spotřeby elektrické energie, druh ovládání tarifu, používaná měna. Je možné si zvolit až 4 tarify, pokud je počet využívaných tarifů

změněn, tak se dynamicky pomocí listeneru aktualizuje tabulka s tarifním převodním poměrem. Potom lze k různým časovým intervalům dne přiřadit různý druh tarifu.

Vybrání časové informace je realizováno pomocí dialogu TimePicker. Lze vytvořit až 24 intervalů s přiřazenými tarify, pokud se jich uživatel pokusí vytvořit víc, bude zobrazena zpráva, že další interval už není možné přidat.

2.4.7 Stahování konfigurace

Stahování konfigurace probíhá hned při startu DeviceConfigActivity aktivity. Viewmodelu této aktivity je předána adresa a port zařízení. Pomocí těchto údajů je vytvořen objekt třídy DeviceManager. Pokud uživatel má přístup k internetu spustí se úloha DownloadConfigInformationTask. Úloha je potomkem třídy AsyncTask a běží v jiném vlákně, aby nedošlo k blokování UI vlákna.

Úloha provádí stahování ve třech krocích. Nejprve se pošle do zařízení zpráva s požadavkem o získání základní struktury konfigurace. Zařízení odešle odpověď se základní strukturou, která se zapíše do UniversalConfigs struktury. V dalším kroku se pošle zpráva s požadavkem o konkrétnější informace o struktuře dat. Například jaký datový typ mají konkrétní konfigurační parametry. Tyto informace se opět uloží do UniversalConfigs struktury. Potom už zbývají jen samotné hodnoty konfiguračních parametrů. K tomu slouží poslední krok, kdy se pošle zpráva s požadavkem o samotná data. Ty už lze načíst, protože je jasné, jaký mají parametry datový typ a jaká je jejich velikost v bajtech.

Pokud jsou všechny tyto kroky bez problému provedeny úloha pomocí listeneru, vrátí konfigurační data do viewmodelu. Viewmodel se postará o správnou reprezentaci dat v uživatelské rozhraní. Pokud by kroky nebyly úspěšně provedeny, úloha by jako výsledek předala hodnotu null a viewmodel by zobrazil chybovou hlášku, že konfigurační data se nepodařilo ze zařízení stáhnout.

Konfigurační data mohou být také stažena pomocí ikony z panelu nástrojů. Pokud už byla data předtím stažena, máme informaci o struktuře dat a není proto nutné opakovat

první dva kroky popsané výše. Provede se pouze poslední krok, tedy stáhnou se jen samotné hodnoty konfiguračních parametrů a aktualizuje se grafické rozhraní.

2.4.8 Nahrávání konfigurace do zařízení

Nahrát konfiguraci do zařízení je možné pomocí ikony v sadě nástrojů. Nahrávání je vyřešeno přes úlohu UploadConfigInformationTask, která je spuštěna jen pouze pokud má aplikace přístup k internetu. Úloze je předána reference na objekt třídy DeviceManager, kde jsou uložena konfigurační data v UniversalConfigs struktuře. Úloha na pozadí spustí metodu SetData, která ze struktury vytvoří pole bajtů. To je potom odesláno v těle zprávy do zařízení. Na zprávu zařízení odpoví, jestli operace proběhla v pořádku. Podle odpovědi je uživateli zobrazena informační zpráva, zdali se operace podařila.

2.4.9 Ukládání konfigurace do souborů

Konfiguraci lze uložit do souboru pomocí ikony v sadě nástrojů. K tomu v kódu aplikace složí třída SaveConfigToFileTask. Vytvoření souboru lze popsat ve třech krocích. Data se do souboru zapisují po bajtech. Struktura se rozděluje do několika skupin a každá skupina má několik konfiguračních parametrů. Nejprve se uloží velikost pole bajtů obsahující informace o konfiguračních skupinách a za ním následuje toto pole. Jedná se například o název a ID kategorie. Dále se uloží velikost pole bajtů, které obsahuje informace o konkrétních konfiguračních parametrech. To je název, ID, typ proměnné, počet hodnot proměnné a informace, zdali se má parametr ukládat. V dalším kroku už se uloží jen samotné hodnoty konfiguračních parametrů.

Takto vytvořený soubor se uloží do externí paměti mobilního zařízení. Přesně do složky „KMB/Configs“. Název souboru je tvořen typem zařízení a za ním současným datem a časem s koncovkou „kmbcfg“. Tento soubor je kompatibilní s aplikací ENVIS. To znamená, že soubory vytvořené v této aplikaci lze otevírat v ENVIS aplikaci a obráceně.

2.4.10 Nahrávání konfigurace ze souborů

Uložené konfigurační soubory jak z této nebo desktopové aplikace ENVIS lze načíst. Po kliknutí na ikonu v panelu nástrojů se spustí dialog. V dialogu se zobrazí list konfiguračních souborů ze složky „KMB/Configs“ v externí paměti mobilního zařízení. Po vybrání konkrétního souboru se spustí úloha LoadConfigFromFileTask. Ta v pozadí načte proud bajtů a načte je do UniversalConfigs struktury. Pokud je soubor validní, dojde k aktualizaci grafické rozhraní a data jsou zobrazena uživateli. Pokud dojde k chybě při nahrávání souboru, je uživatel informován zprávou.

2.5 Archivní data

Tato část aplikace slouží pro stahování archivních dat. Zařízení ukládají archivní data do flash paměti. Je rozlišeno několik druhů archivních dat a každé zařízení může mít těchto druhů různou množinu. Jsou to například záznamy o kvalitě energie, spotřebě elektřiny nebo událostech napětí.

Oproti původní aplikace byla tato část aplikace značně upravena. V původní aplikaci bylo pro stáhnutí dat v určitém intervalu nutné zadat index prvního záznamu a počet záznamu, které chce uživatel stáhnout. To nebylo uživatelsky přívětivé. Proto je nyní možné si zvolit jednu množnost z nabídky, kterou lze vidět na ilustraci č. 18 . V nabídce je stáhnout data ze současného nebo minulého dne, týdnu, měsíce nebo roku. Uživatel si může vybrat, které archivní data si přeje stáhnout pomocí checkboxů. Do aplikace byly přidány další typy archivních dat, které lze stáhnout. Jedná se o archivy s názvy PQ Main, Voltage Events a General Oscillograms.

2.5.1 Stahování

Každý archiv je reprezentován jako instance třídy ArchiveItemViewModel a má svůj grafický vzhled skládající se pouze z CheckBoxu a TextView s názvem archivu. Stahování archivních dat je komplexnější úloha, před samotným stahováním je nutné ověřit několik podmínek, aby bylo stahování uskutečnitelné. Proto je stahování popsáno pomocí tří fází.

První fáze probíhá hned po vytvoření aktivity. Pro každý archiv, který si aktivita drží v listu, se zavolá metoda stahující informace pro daný archiv. Pokud se podaří odeslat a přijmout informace o archivu, znamená to, že je archiv dostupný a je zobrazen v uživatelském prostředí. Pokud se informace o archivu nepodaří získat, archiv není zobrazen a není možné s ním dále manipulovat. Tím se určí, které archivy zařízení podporuje.

Druhé fáze už je závislá na vstupu od uživatele. Pokud uživatel nezaškrtne žádný z archivů nebo nezvolí žádný interval, tak se zobrazí zpráva, že nastavení není validní a data nemohou být stažena. Pokud uživatel zaškrtne aspoň jeden archiv a vybere interval, tak se pro každý vybraný archivu se pošlou dvě zprávy, pomocí kterých se zjistí, jaké záznamy se budou stahovat. Zpráva obsahuje časovou informaci, podle které zařízení nalezne záznam vytvořený v čas nejblíže k přijaté časové informaci a pošle zpátky index vybraného záznamu. První zpráva bude obsahovat spodní hranici vybraného intervalu a druhá zpráva horní hranici. Pomocí těchto indexů je možné určit počet záznamu, které boudou pro daný archiv stažené. Může nastat situace, že počet záznamů bude nula, protože některé archivy nejsou kontinuální a ukládají záznamy jen po určitých událostech.

Pokud je počet archivů roven nule, archiv nebude přiřazen do listu archivů pro stažení. Tím může dojít k tomu, že list archivů bude prázdný, pokud k takové situaci dojde, zobrazí se uživateli zpráva, že neexistují žádné záznamy ke stažení a stahování nebude spuštěno.

Když list s archivy není prázdný, začne třetí fáze, samotné stahování dat. Protože se jedná o dlouhotrvající proces je stahování realizováno pomocí služby [23]. Službě se předá list archivů s indexy záznamů, které mají být staženy. Na notifikační liště je zobrazen průběh stahování [24]. Původně se stahování nedalo ukončit jiným způsobem než vypnutím celé aplikace. Tato podstatná funkce byla do aplikace přidána, zrušení je možné provést gestem. Stačí prstem vybrat okno se stahováním a posunout jej mimo obrazovku. Zrušením stahování se odstraní všechny složky, co za svůj běh služba vytvořila.

Stahované data jsou ukládaný do CEA souboru. Nejprve se vytvoří struktura adresářů. Do adresářů se postupně ukládají jednotlivé soubory. Nejprve TreeList.xml obsahující informace o zařízení a VersionHistory.txt. Potom pro jednotlivé archivy XML soubor s koncovkou „uad“ obsahující popis struktury dat v XML, hlavičku s informace o záznamech a samotné data. Tato struktura složek je potom zkomprimována pomocí ZIP komprese do jednoho CEA souboru. Po vytvoření CEA souboru jsou původní složky už nepotřebné, ale v původní aplikaci nebyly vymazány, čím zbytečně zabíraly prostor, proto bylo do algoritmu přidáno odstranění složek.

2.5.2 Provedené změny

V původní aplikaci byla data stahována pouze po jednom záznamu. Původní algoritmus byl změněn a stahování je prováděno po 10 záznamech. Číslo 10 bylo zvoleno z toho důvodu, protože to je maximální počet záznamu, které zařízení podporuje. Při vyšším počtu záznamu začalo zařízení odesílat chybové zprávy. Přesto se podařilo redukovat dobu pro vytváření a posílání zprávy pro každý záznam a byla docílena vyšší rychlost stahování. Testování je popsáno v kapitole 2.7.

Výsledné stažené CEA soubory neumí tato aplikace zobrazovat, pro zobrazení těchto souborů slouží desktopová aplikace ENVIS. Soubory vytvořené pomocí původní aplikace nebyly v programu ENVIS čitelné. Pro kompatibilitu s aplikací ENVIS bylo nutné upravit několik chyb v předešlé aplikaci.

Pro názvy souborů s příponou „uad“ popisující formát archivních bylo použito 32 bitové CRC namísto původního 64 bitového CRC, aby souhlasili s názvy souborů vytvořené pomocí aplikace ENVIS. Při vytváření souboru TreeList.xml, bylo přidáno vytvoření položky SubDeviceTypes, které předtím nebyly vytvářeny . V tomto souboru byly taky opraveny estetické chyby vznikající při špatném převodu bajtů na řetězce, mezery potom nebyly správně reprezentovány. Při vytváření binárních souborů s příponou „arch“, bylo zamezeno, aby se ukládalo několik prvních bajtů, které se pro dodržení správného formátu ukládat neměly. Dále bylo zamezeno ukládání vadných záznamů. Vadné záznamy mají buď čas rovný nule, nebo maximální hodnotě. Některá archivní data nejsou

při stahování seřazeny podle času, proto nyní algoritmus data kontroluje a pokud nejsou uspořádána, tak je po stažení seřadí.

Na ilustraci č. 20 je zobrazeno otevření soubor CEA aplikací Envis. Otevřený soubor byl vytvořen pomocí této aplikace. Tento soubor je také pro další zkoumání přiložen na CD, které je součástí této práce.

2.6 Identifikace zařízení

Zobrazování identifikačních dat není specifikováno v zadání práce, ale je součástí desktopové aplikace Envis.Daq, proto bylo přidáno i do této aplikace. Tato část aplikace je tvořena jednou aktivitou a úlohou pro stažení identifikačních dat ze zařízení. Na stáhnutí těchto dat existuje speciální zpráva, není tedy nutné stahovat celou konfiguraci ze zařízení.

Tato část aplikace by ještě potřebovala upravit, protože, jak jde vidět na ilustraci č. 12, některé identifikační parametry o zařízení nejsou dekódované do čitelné formy. Pro porovnání jak by měla být data správně dekódována je poskytnuta ilustrace č. 11.

2.7 Profilování a testování aplikace

K profilování byl použit nástroj Android Monitor, který je součástí vývojového prostředí Android Studio. Profilování a testy probíhaly na zařízení ZOPO ZP951 vybavené osmijádrovým procesorem s frekvencí 1,5 GHz, pětipalcovým full HD displejem a systémem Android 5.1. Graf zobrazuje údaje o vytížení paměti RAM, procesoru, sítě a grafického procesoru. Při měření bylo využito měřící zařízení ARTIQ 144 U P005 E s ethernetovým rozhraním s rychlostí 10 Mbit [25]. Ping z mobilu na zařízení trval průměrně 35ms. Aplikace byla testována na lokální síti, která byla tvořená jedním Wi-Fi směrovačem fungující jako přístupový bod, měřícím zařízení a mobilním telefonem. Aplikace byla testována při procházení aktivity s konfigurací. Měření bylo zahájeno hned při startu aplikace. Po spuštění aplikace bylo vybráno zařízení, zvolena aktivita s konfigurací a byly rychle procházeny jednotlivé konfigurační kategorie. Z grafu lze vyčíst, že aplikace potřebovala ke svému chodu maximálně 27,8 megabajtů paměti.

Oproti aplikaci Davida Eislera řešící stejnou úlohu je aplikace podstatně méně náročná na paměť, ve své práci uvádí, že jeho aplikace využívá až 79,3 MB paměti [2]. Vytížení procesoru je nízké pohybující se kolem 10 %, pouze při spouštění bylo vytížení aplikace vyšší. Podle grafu zobrazující vytížení sítě, lze poznat, že vytížení bylo větší pouze jednou a to při stahování konfigurace.

Poslední graf zobrazuje čas, za jak dlouho grafický procesor vykreslí jeden rámec. V tomto ohledu by bylo možné aplikaci optimalizovat. Během rychlého přepínání jednotlivých fragmentů dochází k vykreslování mnoho grafických komponent naráz a přechod mezi fragmenty není zcela plynulý. Vykreslení jednoho rámce několikrát přeskočilo zelenou hranici, která značí snímkovou frekvenci 60 FPS a jednou červenou hranici, která značí 30 FPS.

Ilustrace 9: Sledování vytížení prostředků mobilního zařízení

Dále bylo měřeno trvání jednotlivých operací. Všechny měřené operace a statistické ukazatele jsou uvedeny v příloze v tabulce č. 1. Výsledky v tabulce jsou spíše orientační, protože byly prováděny pouze na jednom měřícím a mobilním zařízení. Každá operace v tabulce byla dvacetkrát provedena a při tom změřena, z těchto měření pak byly vypočítány statistické ukazatelé v tabulce. Měření probíhalo přímo v kódu aplikace pomocí metody System.currentTimeMillis() a výpisu do logu. Operace, které vyžadovali přenos dat přes sít, měly mezi sebou poměrně velké rozptyly. Doby trvání jsou tedy dost ovlivněné komunikací po sítí.

Stáhnutí konfigurace se zpracováním a nahráním do struktury trvalo 56 až 182 milisekund. David Eisler, který ve své práci měřil také stahování konfiguračních dat, uvedl, že stahování dat trvalo 165-185 ms [2]. Jako nejpomalejší se ukázalo, že je nahrávání konfigurace do zařízení, které trvalo až 314 ms. Všechny naměřené hodnoty byly v rámci několika stovek milisekund, což by pro uživatele měl být přijatelný čas.

Dále bylo testováno stahování archivních dat. Stahování bylo prováděno na zařízení ARTIQ 144 U S015 E (10). Nejprve byla porovnána rychlost stahování s předešlou aplikací, která data stahovala pouze po jednom záznamu. Bylo stahováno 17015 záznamů. Původní aplikace stahovala data 29 minut 5 sekund. Po vylepšení stejná úloha trvala 5 minut 47 sekund. Ukázalo se tak, že pomocí novější aplikace je stahování více jak 5krát rychlejší.

Dále bylo stahování archivních dat porovnáváno s aplikací Envis.Daq. Stahováno bylo 700236 záznamů. Stahování na aplikaci pro mobilní zařízení celkem trvalo 3 hodin 18 minut a 21 sekund. Výsledný soubor měl velikost 109 MB. Stejná úloha trvala aplikace Envis.Daq 2 hodiny 2 minuty a 51 vteřin. Z toho lze usoudit, že aplikace Envis.Daq je přibližně 1,6krát rychlejší než aplikace pro OS Android. To je stále znatelný rozdíl, ale je potřeba si uvědomit, že aplikace pro mobilní OS Android mají oproti PC dost omezené prostředky zejména v přidělování operační paměti. Zároveň se prokázalo, že pomocí výsledné aplikace je možné bez problémů stáhnout i větší množství data ze zařízení.

Závěr

Výsledkem této práce je aplikace pro OS Android kompatibilní pro zařízení s verzí Androidu 4.0.3 a vyšší, což činí 98 % současně používaných Android zařízení (duben 2017) [8]. Aplikace komunikuje s měřícími zařízeními pomocí TCP transportního a KMB Long aplikačního protokolu.

Největší část této práce se věnuje konfiguračním datům. Aplikace umožňuje stahování konfiguračních dat, které potom správně reprezentuje pomocí grafického rozhraní. Jednotlivé konfigurační parametry lze také do měřících zařízení nahrávat. Konfigurační data aplikace také dokáže uložit do konfiguračních souborů, které je potom možné zpětně ze souborů nahrávat a to, jak do aplikace popisující tato práce, tak do desktopové aplikace Envis.Daq.

Dalším přínosem práce je vylepšení funkce pro stahování archivních dat. Archivní data je možné stahovat podle časového intervalu, který si uživatel zvolí, což je uživatelsky přívětivější oproti původnímu řešení popsané v kapitole 2.5. Také byl upraven způsob, kterým byly archivní data stahovány ze zařízení, čímž bylo dosaženo vyšší rychlosti při stahování. Díky dalším změnám popsaných v kapitole 2.5.2 je výsledný soubor s archivními daty ve správném formátu a je tak možné jej načíst a používat v aplikaci Envis.

Aplikace byla podrobena několika testům se srovnáním s jinými aplikacemi poskytující stejné funkce. Bylo také popsáno, jaké má aplikace nároky na hardware. Aplikace byla vyzkoušena při komunikace se třemi různými zařízeními s různými konfiguracemi a fungovala správně bez závažných problémů. Bylo by však vhodné aplikaci podrobit širšímu testování, protože se jedná o komplexnější program.

Do aplikace byly také přidány funkce, které nebyly zahrnuty v zadání práce. Tím je lokátor, pomocí kterého lze nalézt dostupné zařízení na lokální síti a také aktivita zobrazující identifikačních informací o zařízení.

Aplikace byla předvedena na mezinárodním průmyslovém veletrhu Hannover Messe a má potenciál pro komerční využití. Aplikace by se dala dále rozšiřovat, například by bylo vhodné přidat průvodce nastavením, jen pro nejzákladnější nastavení pro neznalé uživatele. Dále možnost zobrazení archivních dat, které zatím dokáže zobrazovat pouze aplikace Envis. Také by bylo možné vylepšit práci se soubory, aby si mohl uživatel sám nastavit cestu pro ukládání. Některé fragmenty aktivity s konfigurací by bylo vhodné rozdělit na více malých pro lepší plynulost na mobilních zařízeních, které mají slabší hardware.

Seznam použité literatury

[1] Ing. Adam Smolík. *Aplikace pro zpracování měřených dat promobilní platformu Android*. 2016. Liberec. Fakulta mechatroniky, informatiky a mezioborových studií. Technická univerzita v Liberci

[2] Bc. David Eisler. *Aplikace pro konfiguraci a správu dat měřicích přístrojů v prostředí Android*. 2016. Liberec. Fakulta mechatroniky, informatiky a mezioborových studií. Technická univerzita v Liberci

[3] App Creation Software - Xamarin. *Mobile App Development* [online]. [vid. 9. 5. 2017]. Dostupné z: https://www.xamarin.com/

[4] Statista. *Mobile OS market share 2016* [online]. [vid. 9. 5. 2017]. Dostupné z: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[5] Oracle Technology Network . *Java SE - Downloads* [online]. [vid. 9. 5. 2017]. Dostupné z: http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html

[6] Android Developers. *Android 4.0.3 APIs* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/about/versions/android-4.0.3.html

[7] Android Developers. *Support Library* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/topic/libraries/support-library/index.html

[8] Android Developers. *Dashboards* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/about/dashboards/index.html

[9] Android Developers. *Android Studio* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/studio/index.html

[10] Android Developers. *App Manifest* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro.html

[11] Android Developers. *Activity* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/reference/android/app/Activity.html

[12] Android Developers. *Fragments* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/guide/components/fragments.html

[13] Android Developers. *AsyncTask* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/reference/android/os/AsyncTask.html

[14] MSDN Microsoft. *Implementing the Model-View-ViewModel Pattern* [online]. [vid. 9. 5. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff798384.aspx

[15] Jakub Kinšt. Github. *Android-ViewModelBinding* [online]. [vid. 9. 5. 2017]. Dostupné z: https://github.com/jakubkinst/Android-ViewModelBinding

[16] Wikipedia. *Data binding* [online]. [vid. 9. 5. 2017]. Dostupné z: https://en.wikipedia.org/wiki/Data_binding

[17] Android Developers. *ObservableField* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/reference/android/databinding/ObservableField.html

[18] KMB systems, s. r. o. *Měřící přístroje* [online]. [vid. 9. 5. 2017]. Dostupné z: http://www.kmb.cz/index.php/cs/merici-pristroje

[19] KMB systems, s. r. o. *PA 144, SMC 144, SMV, SMVQ, SMP, SMPQ Multifunctional Panel Meters & Power Quality Analyzers Communication Protocol Manual* [online]. [vid. 9. 5. 2017]. Dostupné z: http://www.kmb.cz/index.php/cs/component/phocadownload/category/14-dokumenty-komunikace?download=213:popis-komunikacnich-protokolu-modbus-kmblong

[19] Ing. Andrea Ronešová. *Přehled protokolu MODBUS* [online]. [vid. 9. 5. 2017]. Dostupné z: http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf

[20] Android Developers. *Preferences* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/reference/android/preference/Preference.html

[22] Ivan Kovac. Github. *TimePickerWithSeconds* [online]. [vid. 9. 5. 2017]. Dostupné z: https://github.com/IvanKovac/TimePickerWithSeconds

[23] Android Developers. *IntentService* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/reference/android/app/IntentService.html

[24] Android Developers. *Notifications* [online]. [vid. 9. 5. 2017]. Dostupné z: https://developer.android.com/guide/topics/ui/notifiers/notifications.html

[25] KMB systems, s. r. o. *Analyzátor sítě a podružný elektroměr ARTIQ 144* [online]. [vid. 9. 5. 2017]. Dostupné z: http://www.kmb.cz/index.php/cs/component/phocadownload/category/5-merici-pristroje?download=326:artiq-144-uzivatelsky-manual

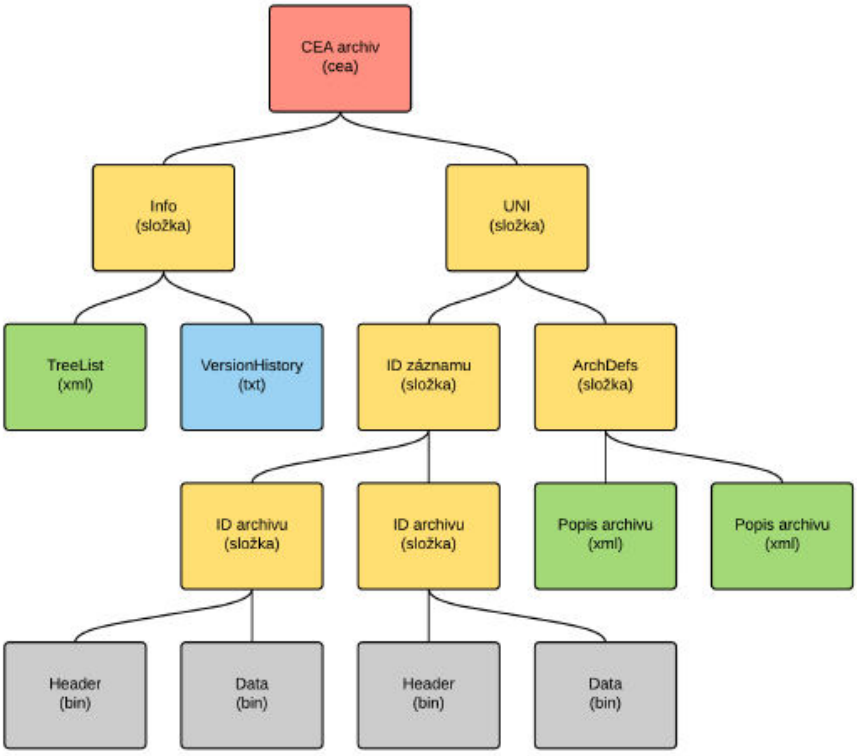
Přílohy

A Obsah přiloženého CD

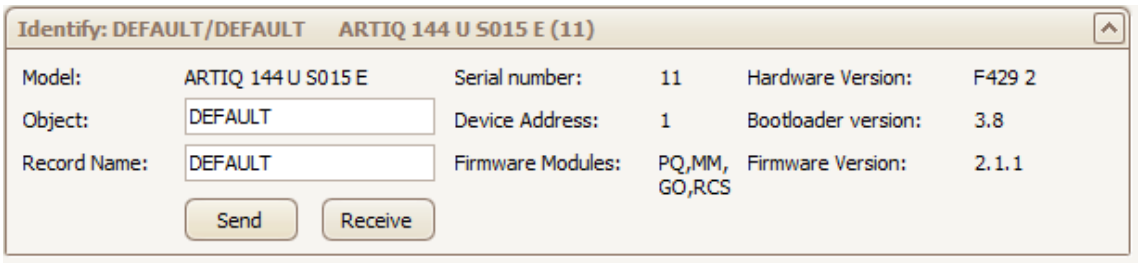
Na přiloženém CD jsou následující složky a soubory:

- archivní data
  - skola147_11-5-2017_10-42-04.cea – CEA soubor vytvořený pomocí této aplikace
- instalační balíček Android aplikace
  - aplikace_2017_Jan_Moravec.apk
- konfigurační data
  - ARTIQ_10-5-2017_15-36-29.kmbcfg – konfigrační soubor vytvořen pomocí této aplikace
- text bakalářské práce
  - bakalarska_prace_2017_Jan_Moravec.pdf

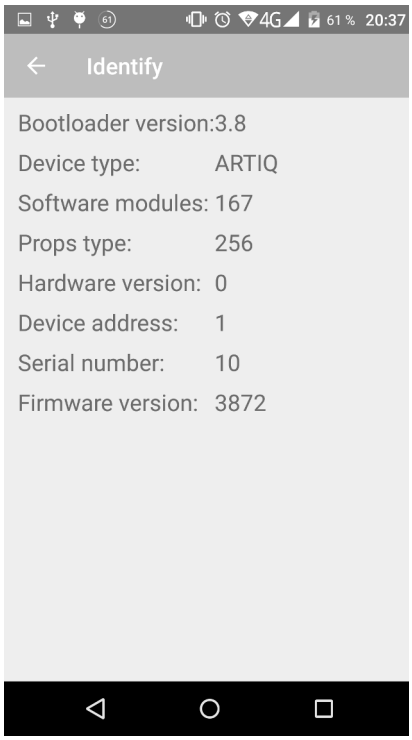
B Obrázky a snímky obrazovky



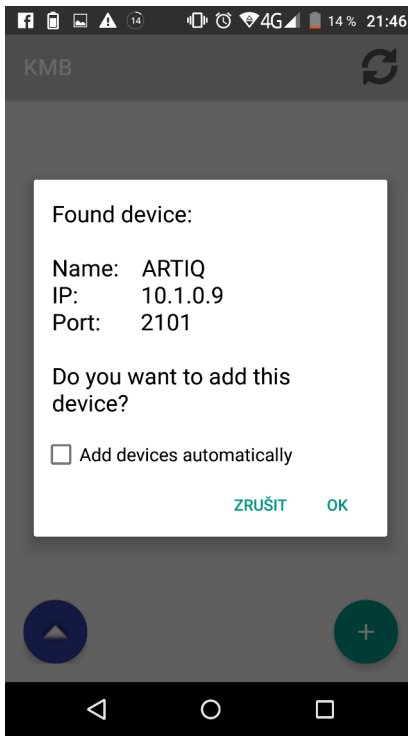
Ilustrace 10: Schéma struktury CEA souboru [1]



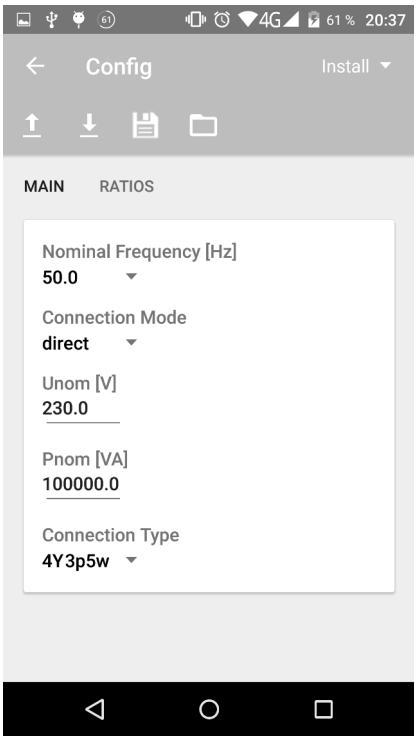
Ilustrace 11: Zobrazení identifikačních data v aplikaci Envis.Daq



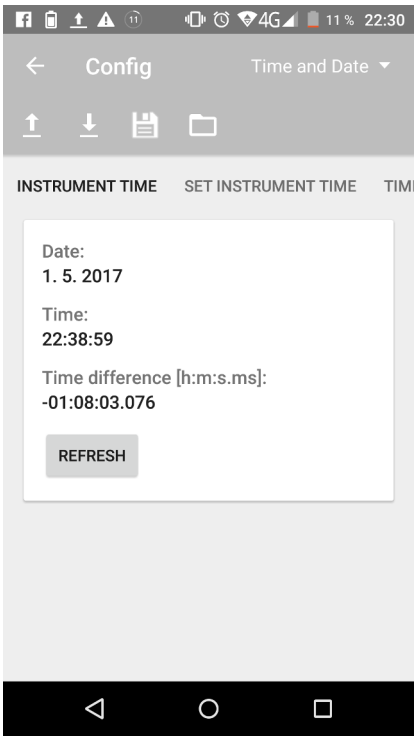
Ilustrace 12: Zobrazení identifikačních dat zařízení



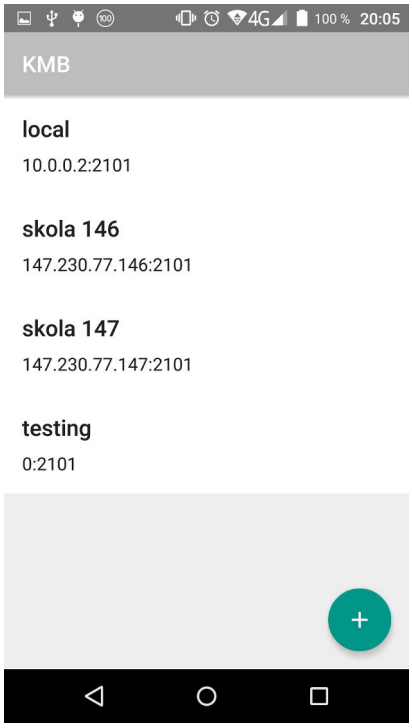
Ilustrace 13: Dialog při nalezení zařízení



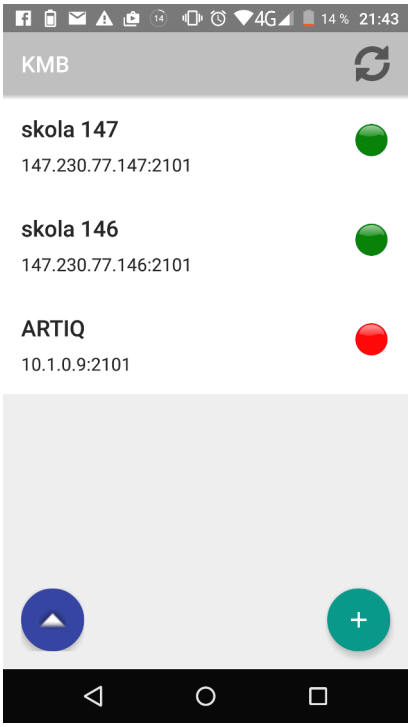
Ilustrace 14: Konfigurace – Instalace



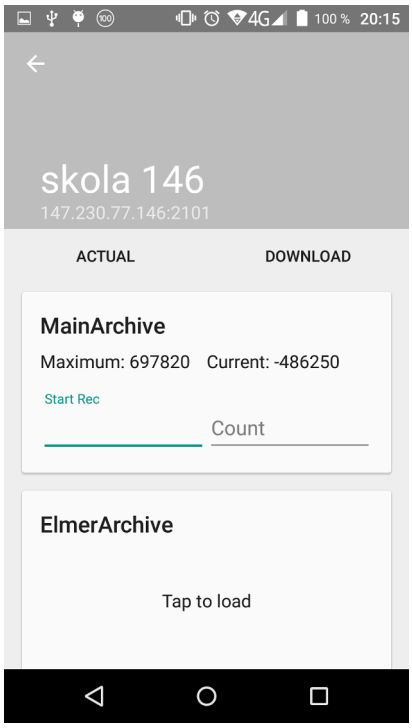
Ilustrace 15: Konfigurace – Datum a čas



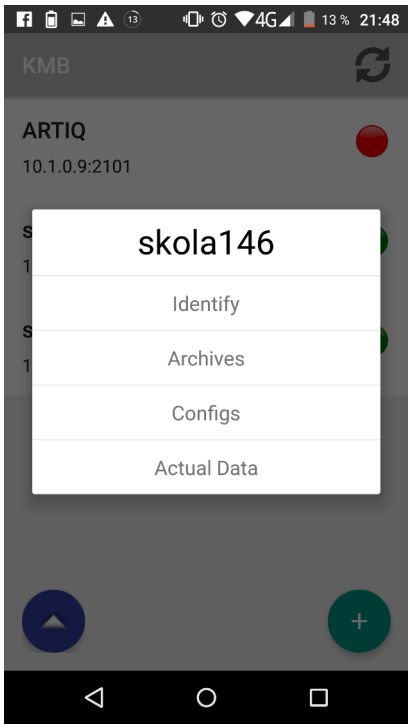
Ilustrace 16: Původní vzhled MainActivity



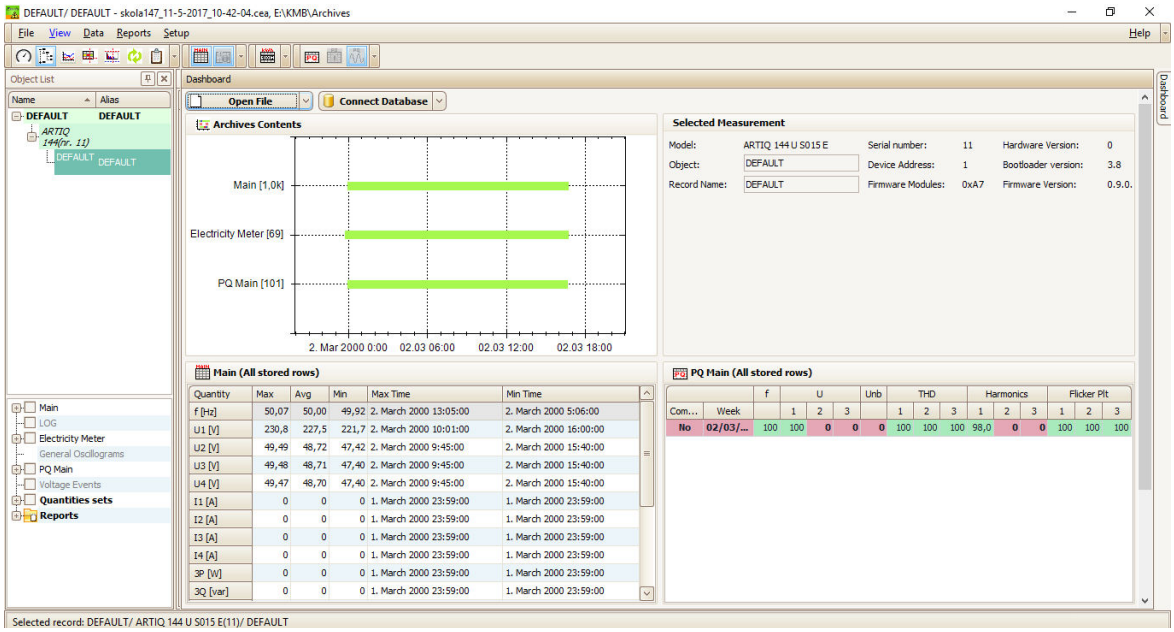
Ilustrace 17: Nový vzhled MainActivity



Ilustrace 18: Původní vzhled detailu zařízení



Ilustrace 19: Snímek obrazovky s navigací



Ilustrace 20: Otevření CEA souboru v aplikaci Envis

C Tabulky

Operace	min	max	arit. průměr	směr. odchylka
Stažení konfiguračních dat a uložení do struktury	56	182	114,4	45,0781777056611
Nahrání konfigurace do zařízení s příjmem odpovědi	70	314	139,35	58,008415904845
Aktualizace nahrané konfigurace a uložení do struktury	23	120	62,15	34,6459916417773
Uložení konfigurace do souboru	15	47	22,8	6,79473480370144
Nahrání konfigurace ze souboru	1	4	2,3	1,03109548284184

Tabulka 1: Orientační statistika doby trvání různých operací v milisekundách (z 20 vzorků)