TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

BAKALÁŘSKÁ PRÁCE

Liberec 2012 Pavel Novák

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

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

Vizualizace dat analyzátoru na přenosných zařízeních

Visualization of analyzer data on portable devices

Bakalářská práce

Autor: Pavel Novák
Vedoucí práce: Ing. Jan Kraus, Ph.D.
Konzultant: Ing. Tomáš Tobiška

V Liberci 16. 5. 2012

Zadání

Zásady pro vypracování:

1) Seznamte se s principy vývoje grafických aplikací pro zvolené mobilní zařízení.
2) Seznamte se se strukturou dat analyzátoru SMPQ a s použitým komunikačním protokolem tohoto přístroje.
3) Navrhněte a implementujte aplikaci pro čtení, záznam, lokální zobrazení dat a konfiguraci analyzátoru.
4) Demonstrujte funkčnost navrženého systému a diskutujte další možnosti rozvoje a omezení Vámi navržené aplikace.

Prohlášení

Byl(a) jsem seznámen(a) 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(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Tímto krátkým odstavcem bych rád poděkoval vedoucímu této práce Ing. Janu Krausovi, Ph.D. za poskytnuté rady, připomínky a celkové vedení práce. Další velké poděkování bych směřoval konzultantovi této práce Ing. Tomášovi Tobiškovi za jeho rady a sdělené informace. Na závěr bych rád poděkoval celé své rodině za trpělivost, kterou se mnou měla při řešení a vypracovávání této práce.

Abstrakt

V rámci této bakalářské práce byla vytvořena aplikace pro mobilní zařízení, která umožňuje spojení a bezdrátovou komunikaci s analyzátory kvality elektrické energie společnosti KMB systems, s.r.o.. Mezi hlavní úlohy v komunikaci patří nastavování analyzátorů, čtení základních identifikačních informací, sledování aktuálně měřených dat a stahování archivů s uchovanými záznamy.

Mezi jednotlivé úkoly patřila mimo jiné volba mobilní platformy, pro kterou bude aplikace vyvíjena. Zvolena byla platforma Android, konkrétně ve verzi 2.1, která pokrývá téměř všechny mobilní zařízení se systémem Android u koncových zákazníků. S vývojem výsledné aplikace byla také postupně vyvíjena komunikační knihovna, která je v aplikaci používána. Tato knihovna byla programována čistě za pomoci prostředků jazyka Java. Tento text se také zabývá komunikačním protokolem KMB Long, pomocí kterého je realizována komunikace s použitými analyzátory.

Dílo má strukturu, která se v první části věnuje výběru, rešerši a seznámení s použitými prostředky. V této části je odůvodněn výběr mobilní platformy, popsán komunikační model, seznámení s protokolem KMB Long a použitými prostředky při vývoji. Druhou částí je návrh, v němž jsou popsána rozhodnutí, jednotlivé kroky ve vývoji a návrh jednotlivých komponent aplikace. Dále je popsána implementace, která popisuje přímo některé postupy či problémy při vývoji a je zakončena úvahou nad dalšími možnými rozšířeními aplikace.

Přínosem této bakalářské práce je výsledná aplikace určená pro mobilní zařízení, která dokáže komunikovat se zmíněnými analyzátory. Výsledný produkt je možné používat jako zjednodušenou a přenositelnou náhradu za aplikaci ENVIS.Daq. Komunikační knihovna byla vyvíjena odděleně od samotného programu a je možné ji využít v dalších projektech či aplikacích a dále rozšiřovat a zdokonalovat.

Abstract

The application for mobile devices which enables connection and wireless communication with analysers of power quality, designed by company KMB systems, s.r.o., was created in this bachelor work. Main functions of this application are settings of analysers, reading basic identification information, monitoring the measured data and downloading archives.

Including other tasks belonged the choice of mobile platform which the application will be developed for. The platform Android was selected in version 2.1, because it covers almost all mobile devices with Android operating system by end-users. While the application was in develop, the communication library which is used in the application was developed too. This library was developed only with programming language Java. This text describes communication protocol KMB Long which realizes the communication with used analysers.

The structure of this work is following. First part is about choice, research and get knowing with used tools. In this part there is mentioned the choice of mobile platform, described the communication model, meeting the protocol KMB Long and used tools by developing. Second part is about designing. There are described solutions, each steps in development and design each components of the application. Finally there is described the implementation which discusses some of steps or problems during developing and it ends with consideration of other extensions of program.

The contribution of this work is developed application for mobile devices. This program can communicate with named analysers. Final product can be used as simplified and portable replacement for program ENVIS.Daq. The communication library was developed separately from the program and it can be used in other projects or applications and could be extended and improved.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tématem této bakalářské práce je vizualizace dat analyzátoru na přenosných zařízeních. Téma bylo zvoleno z důvodu vzrůstajícího zájmu a perspektivního výhledu na vývoj aplikací pro mobilní zařízení. Komunikace může být realizována pomocí spojení wi-fi nebo za pomoci internetu poskytovaného například od mobilního operátora. Hlavním cílem této práce je naprogramování aplikace pro mobilní zařízení, která bude umožňovat konfiguraci analyzátorů a vizualizaci aktuálně měřených dat pomocí tabulek či grafů. Dílo se konkrétně zabývá analyzátory řady SMPQ od společnosti KMB systems. Tyto analyzátory umožňují dosti přesně měřit a vyhodnocovat kvalitu elektrické energie a mnoho dalších veličin. Hlavním smyslem a přínosem této práce je možnost následně využívat vytvořenou aplikaci ke snadné správě analyzátorů z mobilních zařízení, typicky třeba z klasického mobilního telefonu. To přináší pohodlné možnosti konfigurace a sledování měřených dat, které analyzátor vyhodnocuje. Společnost KMB systems nově vydala SMC 144, který nevlastní žádný displej. Dále se očekává vydání celé nové série analyzátorů s absencí displeje. V tomto případě lze aplikaci vyvíjenou v rámci této práce jako náhradu za displej těchto zařízení.

Cílová vyvíjená aplikace by měla umožňovat vizualizace aktuálně měřených dat, konkrétně veličin spadajících pod napětí, proud a výkon a měla by obsahovat fázorový diagram. Další funkcí je konfigurace analyzátoru. Pomocí programu by mělo být možné provádět konfiguraci instalace analyzátoru, názvu měření a objektu a konfiguraci komunikačních parametrů, tedy IP adres a portů. Dále by aplikace měla umožňovat čtení a zobrazování identifikačních informací, jako jsou například typ analyzátoru, sériové číslo, verze firmwaru a další. Mezi cílové funkce patří stahování archivů z analyzátoru a jejich následné ukládání do CEA souborů. Ukládání do CEA souborů by mělo být realizováno pomocí specializované knihovny od Adama Smolíka, který tuto problematiku řešil ve své bakalářské práci. Pro vývoj aplikace nebyla definována žádná omezení, přičemž hlavním omezením byl čas. Aplikaci by bylo možné rozvíjet a přidávat nové funkce v podstatě neustále.

Nejprve byla provedena rešerše použitelných prostředků a lehké srovnání s konkurencí. Jako cílová mobilní platforma byl zvolen momentálně velice populární operační systém Android. Tvorba aplikace tedy probíhala podle pravidel a předpokladů pro vývoj aplikací určených pro Android. Pro komunikaci mezi analyzátorem a aplikací byla navržena a implementována samostatná knihovna, která byla naprogramována v jazyce Java. Z tohoto důvodu je tato knihovna dále použitelná i v dalších aplikacích.

1 Použité prostředky

1.1 Analyzátory SMPQ

Analyzátory SMPQ (Power Quality) [15] patří do samostatné rodiny analyzátorů měřících a vyhodnocujících kvalitu elektrické energie. Tyto analyzátory vyvíjí firma KMB systems, s.r.o. [14], která sídlí ve městě Liberec. Firma vznikla v roce 1991, přičemž v roce 1992 byla registrována do obchodního rejstříku jako společnost s ručením omezením. KMB systems, s.r.o. se zabývá vývojem zařízení orientovaných na měřící a regulační techniku, dále výrobou těchto zařízení a jejich prodejem.

Skupina analyzátorů SMPQ má mnoho společného s analyzátory z rodiny SM, konkrétně pak se SMV a SMP, přičemž disponuje všemi jejich možnostmi a obsahuje další funkce. Jedná se o nejvíce vybavené analyzátory z kategorie kompaktních přístrojů firmy KMB systems. Oproti analyzátorům SMV a SMP dokáží většinu veličin měřit s větší přesností a umožňují vyhodnocovat flicker. Třída analyzátorů SMPQ vlastní 2 modely, konkrétně SMPQ 33, který dokáže vyhodnocovat měřené veličiny na 3 fázích a SMPQ 44, který navíc umožňuje měření na vodiči N. Tyto analyzátory zastávají mimo jiné funkce jako měření aktuálních hodnot veličin jako napětí, proudu, jalového, deformačního a činného výkonu, vyhodnocování harmonických až do 50. řádu, tvorba záznamů měření, včetně logů přístroje. Dále umí měřit krátkodobé a dlouhodobé míry vjemu flickeru a mnoho dalších funkcí.

Analyzátory rodiny SMPQ komunikují s ostatními klientskými zařízeními pomocí více komunikačních rozhraní. Každý model SMPQ analyzátoru může být vybaven různými rozhraními. Standardně mají podporu USB, volitelně sériové linky RS-232, RS-485 nebo ethernetu. Tato práce se konkrétně věnuje komunikačním rozhraním ethernet IEEE 802.3.

Komunikace po ethernetovém komunikačním rozhraní umožňuje připojit analyzátor přímo do počítačové sítě typu LAN, případně je možné pomocí veřejné IP analyzátor zpřístupnit do sítě internet. Z pohledu komunikace s mobilním zařízením je zajímavější možnost bezdrátového spojení pomocí wi-fi, protože většina dnešních

mobilních telefonů disponuje integrovaným wi-fi modulem. Firma KMB systems vlastní také přenosné analyzátory, které mají integrovaný wi-fi modul. Mezi ně patří například řada SIMON PQ, která se svými vlastnostmi velice podobá analyzátorům SMPQ, obsahuje většinu jejich funkcí, ale má absenci vestavěného displeje.

1.1.1 Dodávaný software

Společnost KMB systems také vyvíjí a distribuuje software pro vyhodnocování kvality elektrické energie, který umožňuje komunikaci mezi analyzátory a počítačem, na kterém je software nainstalován. Tento software je, v případě analyzátorů SMPQ, pojmenován ENVIS. Dále existuje program RETIS, který je primárně určen pro regulátory jalového výkonu NOVAR. ENVIS umožňuje práci se záznamy analyzátorů, zobrazování dat ze záznamů pomocí grafů či tabulek, jejich celkovou analýzu, export dat a další funkce. Software obsahuje nástroj, pomocí kterého lze analyzátory konfigurovat, sledovat aktuální průběhy dat a stahovat záznamy z analyzátoru do souborů či databáze. Zmíněný nástroj se nazývá ENVIS.Daq. Ten lze považovat za abstraktní předlohu výsledné aplikace zadané bakalářské práce. Výsledný program by měl pokrývat vybranou funkcionalitu tohoto nástroje. Aplikace ENVIS obsahuje dále nástroj ENVIS.Online, který umožňuje pravidelné stahování dat z analyzátorů. Software ENVIS je určen na stanice s operačním systémem Microsoft Windows, konkrétně Windows 7, Windows Vista a Windows XP. Verze tohoto softwaru pro nasazení na mobilní telefony neexistuje.

1.1.2 Konkurence

Firma KMB systems není ve světě, a ani v České republice, jedinou firmou svého druhu. Z Českých firem, zabývajících se výrobou a vývojem analyzátorů k měření a vyhodnocování kvality elektrické energie, lze jmenovat společnost MegA - Měřící Energetické Aparáty, a.s. [18], která úzce spolupracuje s univerzitou VUT v Brně, se kterou vlastní společnou laboratoř. Společnost MEgA také vyvíjí vlastní software pro své přístroje, ale též pouze pro nasazení na počítačích. Vývoj mobilního softwaru nebyl zveřejněn.

Světově známou firmou v této oblasti je firma Fluke [22] sídlící ve státě Washington v USA. Tato společnost se zabývá vývojem elektronických měřících přístrojů již od roku 1948. Korporace disponuje velkým počtem analyzátorů, ale v textu je zmíněna především z důvodu, že některé její výrobky používají vizuální rozhraní pro PDA. Jedná se například o třífázový záznamník kvality výkonu Fluke 1750. Tato měřící jednotka je typicky přímo dodávána včetně PDA s operačním systémem Android a aplikací Fluke power view [8]. Aplikace umožňuje spojení s analyzátorem pomocí bezdrátové technologie wi-fi a následné zobrazení důležitých dat a vizualizací, jako je například fázorový diagram.

1.2 Komunikační model

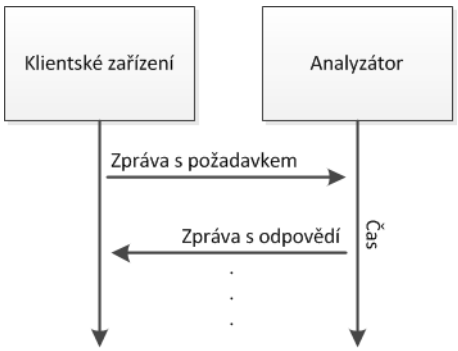
Kapitola popisuje použitý komunikační model, systém výměny zpráv mezi analyzátorem a klientem, navazování a ukončování spojení. Komunikační proces je v kapitole podrobně okomentován a vysvětlen na doprovodných obrázcích. Samotná struktura zpráv je popsána v samostatné kapitole 1.3.

Při navazování komunikace se vytvoří klasické TCP/IP socketové spojení, což umožňuje relativně snadnou komunikaci a výměnu zpráv. Zprávy se následně odesílají a přijímají pomocí výstupních a vstupních proudů na daném navázaném socketu. Díky volbě TCP/IP se nemusí řešit další režijní vlastnosti komunikace, protože TCP je stavový protokol, který sám o sobě kontroluje doručená data a jejich správnost. Pro navázání spojení s analyzátorem pro rozhraní ethernet či wi-fi je nutné znát jeho IP adresu a cílový port zařízení. Cílový **port** na analyzátorech SMPQ je typicky nastaven na číslo **2101**, ale je možné ho manuálně změnit například prostřednictvím aplikace ENVIS.Daq, nebo pomocí výsledné aplikace ENVIS mobile.

Obrázek 1: Komunikace z pohledu analyzátoru

Komunikace z pohledu analyzátoru je znázorněna na obrázku 1. Po navázaní spojení analyzátor čeká na dotaz, který následně zpracuje a odešle jeho výsledek ve zprávě s odpovědí. Ve stavu, kdy analyzátor čeká na dotaz a nevykonává v daném spojení žádnou další komunikaci, je vhodné v případě potřeby žádat o ukončení spojení. Takto probíhá komunikace v ideálním případě. V reálném provozu se může stát, že je přijata žádost o ukončení spojení například při vyřizování dotazu, nebo může nastat situace, že k hladkému ukončení spojení nedojde vůbec.

Jako model výměny zpráv je u analyzátorů SMPQ zvolen model typu **požadavek-odpověď**. Požadavek-odpověď [23] je mimo jiné znám jako *request-reply* nebo *request-response*. Jedná se o jeden z nejjednodušších vzorů pro výměnu zpráv, který je preferován především ve spojeních založených na architektuře klient-server. Tento model z pohledu analyzátoru funguje na principu, kdy serverová část (v tomto případě analyzátor) vyčkává na obdržení nějakého požadavku od části klientské (počítač, mobilní zařízení, ...). Po přijetí požadavku je zpracována odpověď, která je následně odeslána klientovi. Takto se v navázaném spojení může klient dotazovat na mnoho zpráv a přijímat od analyzátoru odpovědi bez nutnosti řešení další režie v komunikace. Model je znázorněn na obrázku číslo 2.



Obrázek 2: Model požadavek-odpověď

1.3 Protokol KMB Long

Analyzátory SMPQ typicky používají pro komunikaci vlastní protokol zvaný KMB Long (lze komunikovat také pomocí protokolu Modbus). Ten je navržen firmou KMB systems. Funguje prostřednictvím výměny zpráv. Zprávy lze rozdělit do dvou skupin – odchozí a příchozí. Jak odchozí, tak příchozí zprávy mají pevně daný formát, který musí být striktně dodržován. Na základní strukturu/formát zprávy nemá vliv, jakého je typu. V případě nedodržení analyzátor odešle zprávu nesoucí kód chyby, případně neodpoví vůbec.

Všechny hodnoty v těchto zprávách jsou zaznamenávány pomocí endianity [5] typu **big-endian**. To znamená, že se na paměťové místo s nejnižší adresou ukládá nejvíce významný byte dané hodnoty.

Adresa zařízení

Adresa zařízení definuje adresu analyzátoru. Je pro ni vyhrazen blok o velikosti jednoho bytu, tudíž může obsahovat hodnoty 0 až 255. V USB komunikaci je adresa zařízení ignorována. V základní konfiguraci mají analyzátory SMPQ nastavenu adresu na hodnotu 1.

Délka těla zprávy

Blok nesoucí délku těla zprávy musí dodržovat velikost 2 byty. Z toho vyplývá, že tělo zprávy nemůže obsahovat více dat, než se lze uchovat pomocí 8 kilobytů. Analyzátory SMPQ ovšem vracejí zprávu s chybou *0xfe*, pokud by tělo jejich odpovědi obsahovalo více než 7 kilobytů. Délka těla je v tomto případě synonymem pro počet bytů obsažených v těle zprávy.

Kód zprávy

Kód zprávy, nebo také *Command code* je její identifikátor, který definuje, jaká data jsou přenášena, případně z něj analyzátor určí, jaká data požadujeme. Jelikož je identifikátor délky jednoho bytu, může analyzátor poskytovat obsluhu až pro 256 typů zpráv. V následující tabulce je ukázka několika nejpoužívanějších kódů zpráv, které byly v rámci této práce nejhojněji používány.

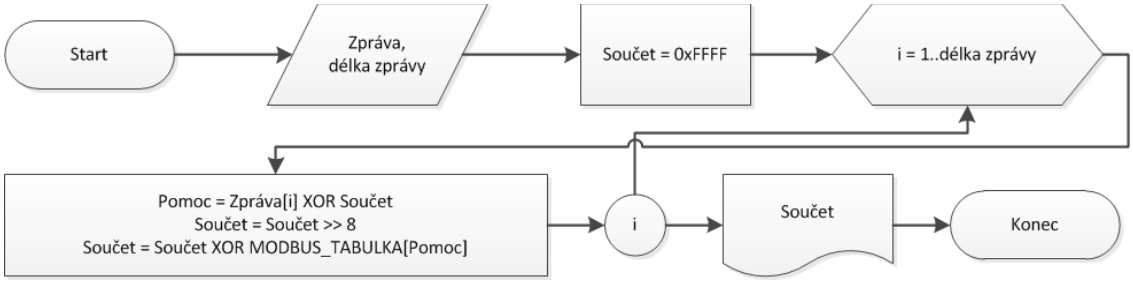
Tělo

Tento blok obsahuje samotná přenášená data, ve kterých jsou například uloženy hodnoty různých veličin, nastavení nebo informací o analyzátoru. Tělo nemá pevně určenou velikost a u každé zprávy se může lišit. Jeho velikost je uložena v bloku, který uchovává velikost těla zprávy. V případě dotazu může být u některých typů zpráv toto tělo nulové. Například v případě *GET_IDENTIFY* o sobě analyzátor odešle základní informace, ale klientské zařízení mu odešle dotaz s nulovou délkou těla, protože

požadované informace, které chce získat jsou definovány samotným kódem. Délka těla je analyzátory SMPQ omezena na velikost 7 kilobytů, což je dostatečná velikost pro přenášená data.

CRC

Jedná se o závěrečný kontrolní součet celé zprávy. Tento kontrolní součet je počítán ze všech předchozích bloků, tedy z adresy zařízení, délky těla zprávy, kódu zprávy a těla. Kontrolní součet je počítán pomocí algoritmu **CRC16 Modbus**.



Obrázek 3: Vývojový diagram výpočtu kontrolního součtu metodou CRC16 Modbus

Tento algoritmus vrací součet jako dvoubytové celé číslo. Metoda výpočtu tohoto algoritmu využívá pole předdefinovaných dvoubytových konstant. CRC16 Modbus je počítán ze zprávy postupem, jež je znázorněn v ukázkovém vývojovém diagramu na obrázku 3.

V tabulce 1 je dodatečně shrnuta základní struktura zprávy. Druhý řádek tabulky informuje o definované velikosti daného bloku. Struktura zachovává požadované pořadí bloků.

Tabulka 1: Základní struktura zprávy

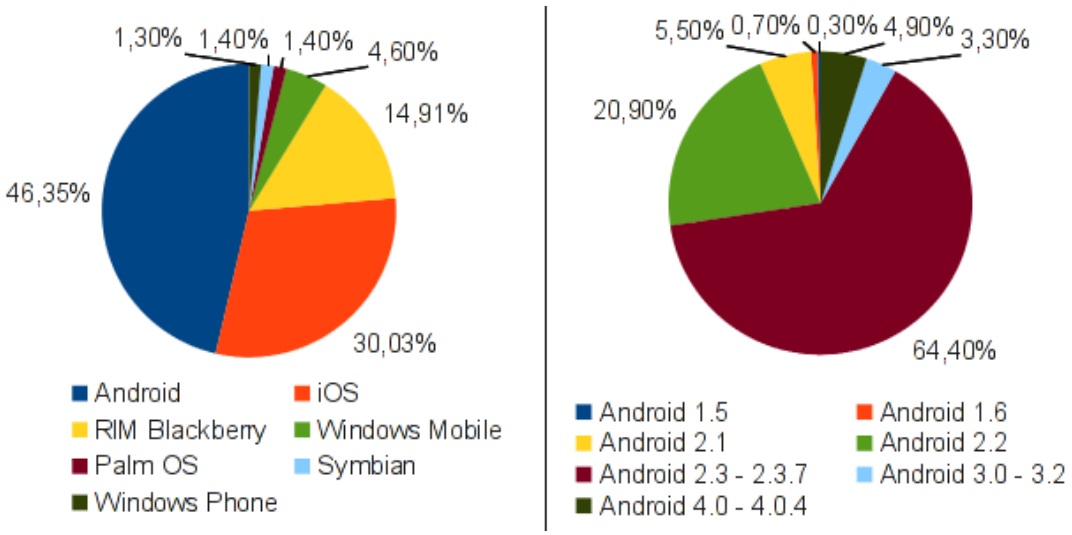
Adresa zařízení	**Délka těla zprávy**	**Kód zprávy**	**Tělo**	**CRC**
1B	2B	1B	??B	2B

Analyzátory SMPQ ovšem v komunikaci přes protokol TCP kontrolní součet CRC zprávy ignorují. To se na první pohled jako problém jevit nemusí, protože komunikace probíhá po TCP/IP spojení, kde každý datagram ve své hlavičce obsahuje

vlastní součet přenášených dat. Data by v případě ethernetu či wi-fi měla dorazit do cíle vždy správně, případně by se měl mechanismus protokolu TCP postarat o to, aby byla data přeposlána. Na druhou stranu může nastat problém, pokud je špatně implementován nějaký mechanismus tvorby zpráv v klientském zařízení. Klientské zařízení může poté poslat do analyzátoru chybnou zprávu, která může v nejhorším případě ztrátu archivů, případně způsobit zaseknutí analyzátoru. Popis jednotlivých zpráv je uveden v příloze Příloha B.

1.4 Volba mobilní platformy

Před samotným počátkem vývoje komunikační knihovny a aplikace bylo třeba vybrat mobilní platformu, na které by aplikace mohla fungovat. Mezi uvažované a aktuálně používané mobilní operační systémy patří systém Symbian [24], Android [10] vyvíjený společností Google, iOS [12] společnosti Apple a Windows Phone 7 [4] od firmy Microsoft. Volba operačního systému závisela na několika faktorech. Prvním faktorem je rozšířenost mobilních telefonů, případně tzv. chytrých zařízeních na trhu a mezi koncovými uživateli. V tomto bodě vítězí jednoznačně mobilní zařízení s operačním systémem Android. Například podle statistik americké společnosti Gartner [9], která prováděla výzkum celosvětového prodeje mobilních zařízení koncovým uživatelům podle obsaženého operačního systému za třetí kvadrant roku 2011, měl operační systém Android zastoupení 52,5 procent. Druhý v pořadí byl systém Symbian, za kterým následoval iOS. Společnost Nielsen [19] provedla podobný výzkum, ve kterém se snažila porovnat celkové zastoupení operačních systémů u dosavadních koncových zákazníků. V těchto statistikách si opět nejlépe vedl Android, který vlastnil údajně celkové zastoupení 46,3 procent. Kompletní graf vystihující statistické měření je zobrazen na obrázku číslo 4 v levé části.



Obrázek 4: Levá část: statistika zastoupení mobilních OS u koncových uživatelů od společnosti Nielsen [19] na začátku roku 2012, pravá část: zastoupení verzí OS Android u koncových zákazníků od společnosti Google [10] - duben 2012

Dalším důležitým faktorem výběru cílového operačního systému je programovací jazyk, pomocí kterého lze psát aplikace pro daný OS. V tabulce 2 je uveden seznam uvažovaných mobilních OS a jim přidružených programovacích jazyků. V tomto případě byl opět vybrán Android. Hlavním důvodem je možnost programování aplikací a knihoven v jazyce Java [21]. Programovaní v jazyce Java přináší totiž možnost přenositelnosti aplikací a knihoven do dalších prostředí, jako je například desktopová Java SE.

Tabulka 2: Přehled mobilních platforem a jejich hlavních prog. jazyků

Mobilní operační systém	**Programovací jazyk**
Android	Java
iOS	Objective-C
Symbian	Qt, Symbian C++
Windows Phone 7	Silverlight, XNA framework

Zvolena byla platforma Android. Jádro tohoto systému je založeno na Linuxu, konkrétně na verzi 2.6, verze Android 4 na linuxovém jádře ve verzi 3. Běhové prostředí zajišťující chod aplikací se nazývá Dalvik [6]. Dalvik je vyvinut přímo pro operační systém Android, přičemž se při jeho vývoji dbalo na bezproblémový provoz

na mobilních zařízeních a na nízkou spotřebu baterie. Toto prostředí implementuje většinu standardních knihoven jazyka Java. Jazyk Java je včetně jeho knihoven volně šiřitelný, proto je mohla společnost Google při vývoji běhového prostředí Dalvik použít. Dalvik neimplementuje knihovny pro vývoj GUI jako AWT a Swing. Operační systém Android [20] totiž disponuje vlastními knihovnami pro grafické uživatelské prostředí. Při programovaní grafických aplikací pro systém Android je možné grafické prvky definovat přímo ve zdrojovém kódu, nebo v externích XML souborech.

1.5 Vývojové prostředky

Jedním z hlavních požadavků pro vývoj aplikací pro operační systém Android je JDK Java Development Kit ve verzi 6, což je balíček standardních nástrojů pro vývoj Java SE aplikací. Pomocí JDK jsou zdrojové kódy aplikací pro Android překládány. Překlad a sestavení je řízen pomocí nástroje Apache Ant [25]. Tento nástroj umožňuje řídit kompilaci a sestavit výsledný balík pro distribuci aplikace (v případě vývoje pro Android se jedná o APK soubor). Všechny konfigurace pro nástroj Apache Ant se píší ve značkovacím jazyce XML. Dalším důležitým nástrojem pro vývoj je Android SDK, který obsahuje vývojové nástroje a emulátor Android zařízení. Pomocí Android emulátoru je možné testovat vyvíjené aplikace na různých verzích Androidu, odlišných velikostech displejů a různých emulovaných hardwarových modulech. Emulované stroje se nazývají AVD (Android Virtual Device).

Pro vývoj aplikací pro Android je předně doporučované vývojové prostředí **Eclipse**. Toto prostředí nabízí podporu pro vývoj Android aplikací po instalaci ADT rozšíření. V tomto rozšíření je také integrován modul pro snadnou tvorbu grafického uživatelského rozhraní. Eclipse IDE je distribuováno zdarma. Mnoho informací o vývoji v prostředí Eclipse je uvedeno na stránkách určených pro vývojáře aplikací pro Android [10] od společnosti Google. Alternativou, jak vyvíjet aplikace pro systém Android, je vývojové prostředí **Netbeans**. Netbeans podobně jako Eclipse nemá přímou podporu správy Android projektů a pomocné nástroje pro vývoj. Tato podpora je možná přidat pomocí instalace rozšíření NBAndroid [2]. V tomto rozšíření není doposud obsažen modul pro tvorbu grafického uživatelského rozhraní. Aplikace pro Android je

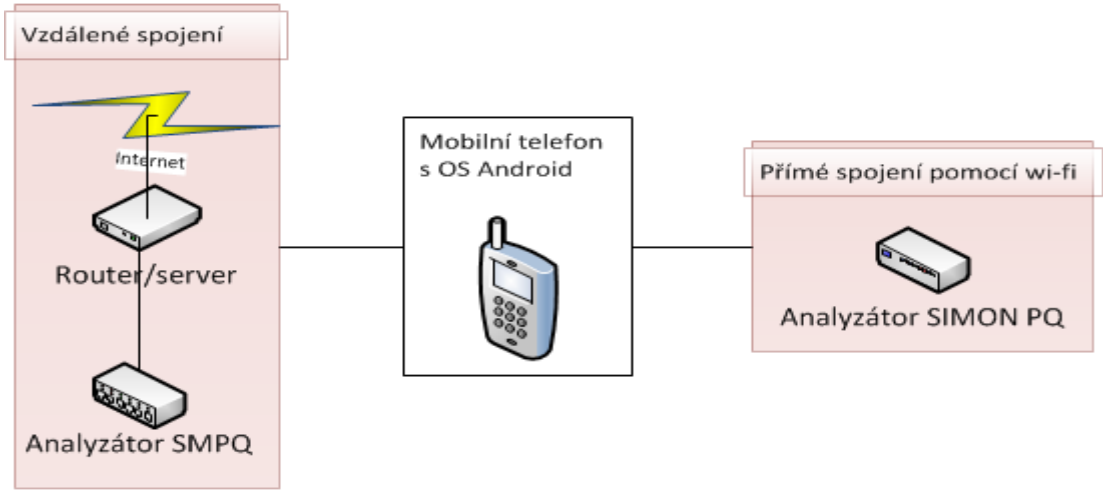
také možné programovat v jakémkoliv textovém editoru, ale tím vývojář přichází o výhody vývojových prostředí jako například zvýrazňování a kontrola syntaxe, automatizované kompilování projektu, možnosti ladění aplikace a tak dále.

2 Návrh řešení

Mezi hlavní cíle této bakalářské práce patří návrh aplikace pro mobilní zařízení a její následná implementace. Tato kapitola se bude věnovat pouze návrhu výsledné aplikace, přičemž její implementace bude popsána v kapitole 3 . Výsledná aplikace by měla umožňovat připojení se k analyzátoru, načtení základních informací, nastavení instalace a komunikačních informací jako IP adresu, výchozí bránu atd., zobrazování měřených aktuálních dat, stahování hlavních archivů, archivů elektroměru, archivů logů přístroje a jejich následná komprese do CEA souborů. Řešení tvorby CEA souborů není zadáním této práce a k této funkcionalitě slouží knihovna od Adama Smolíka, který ji vyvíjel ve své bakalářské práci.

2.1 Možnosti spojení aplikace s analyzátory

Při návrhu aplikace se brala v úvahu dvě hlavní možnosti spojení. Ta jsou znázorněna na obrázku číslo 5. První možnost spojení je založena na připojení analyzátoru SMPQ pomocí ethernetu (lze podobně realizovat s analyzátorem SIMON PQ) k routeru či serveru, který přiřadí analyzátoru veřejnou IP adresu. Připojit se k takto zapojenému analyzátoru je možné téměř všude, kde je dostupné internetové připojení. Druhá možnost, která je založena na přímém spojení mezi analyzátorem a mobilním zařízením, je vhodná především pro přenosné analyzátory SIMON PQ, které mají stejný komunikační protokol jako analyzátory SMPQ, tudíž by aplikace měla umět

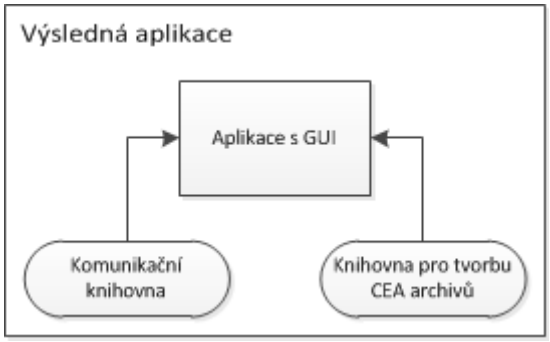


Obrázek 5: Možnosti spojení s analyzátory

komunikovat i s těmito analyzátory. V tomto případě lze aplikaci plnohodnotně (v rámci implementovaných funkcí) použít jako vzdálený displej analyzátoru a usnadňovat tak manipulaci při jeho nastavování a vizualizaci dat.

2.2 Rozdělení funkčních bloků

Aplikace, která by měla být výsledkem práce, je relativně rozsáhlá, proto ji bylo při jejím návrhu nutné rozdělit na menší funkční bloky. První hlavní částí je komunikační knihovna, která umožňuje samotnou komunikaci s analyzátory. Druhá část je samotná aplikace umožňující koncovému uživateli snadnou manipulaci při konfiguraci přístroje, sledování aktuálních dat a informací o analyzátoru. Za třetí část lze považovat použitou knihovnu pro tvorbu CEA archivů. Model funkčních bloků je znázorněn na obrázku 6. Samotná aplikace s grafickým uživatelským rozhraním bude využívat dvou hlavních modulů (knihoven), které ji budou zajišťovat komunikaci, stahování archivů a jejich ukládání do CEA souborů.



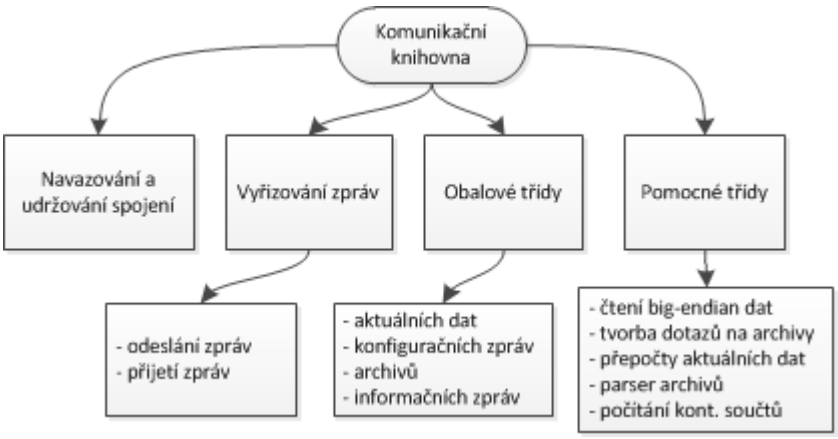
Obrázek 6: Model funkčních bloků výsledné aplikace

Takovéto rozdělení je vhodné především ze dvou důvodů. Prvním je celkové zjednodušení pro implementaci a orientaci v programu. Dalším důvodem je snadnější testování, hodnocení a optimalizace jednotlivých komponent daného komplexního systému. Například testování funkčnosti komunikační knihovny by bylo zbytečně komplikované, pokud by muselo probíhat v rámci kompletní aplikace.

2.3 Komunikační knihovna

2.3.1 Základní vlastnosti

Při návrhu komunikační knihovny bylo rozhodnuto, že tato knihovna bude naprogramována kompletně pomocí tříd ze standardního balíku jazyku Java (konkrétně ve verzi 1.6), které jsou podporované v systémech Android. Toto rozhodnutí přináší



Obrázek 7: Zjednodušený návrh komunikační knihovny

lepší možnosti testování, ale především přenositelnost knihovny, tudíž je možné její následné použití například v desktopové aplikaci. Zjednodušený návrh komunikační knihovny je znázorněn na obrázku číslo 7. Přeložené třídy knihovny budou ukládány do balíku JAR [17], což je Java archiv. JAR umožňuje mít třídy z celého projektu v jednom souboru, který poté slouží jako klasická knihovna. Komunikační knihovna byla nazvána **KmbLongCommunicator**. Pokrývá komunikační vrstvu mezi výslednou aplikací a analyzátory. Pokrývat by měla konkrétně následující funkcionalitu.

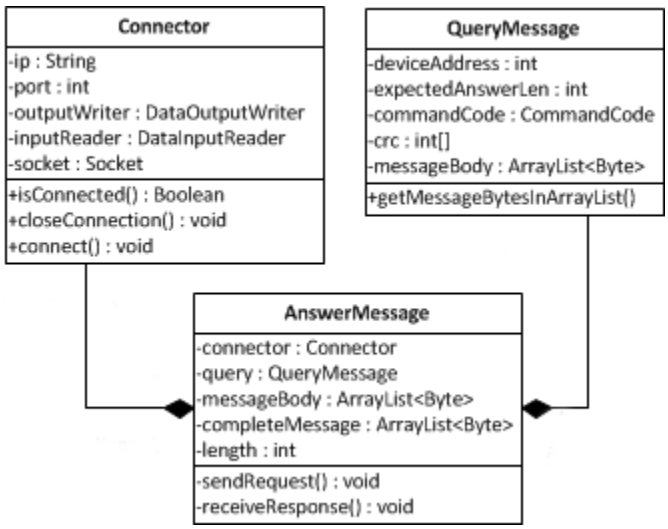
- připojení k analyzátoru
- odesílání a přijímání zpráv
  - identifikační zprávy
  - konfigurační zprávy
  - zprávy s aktuálně měřenými daty
  - zprávy umožňující stahování archivů
- vlastnit obalové a pomocné třídy pro jednotlivé zprávy
- odpojení od analyzátoru

Při snaze o zjednodušení knihovny byly navrženy pomocné třídy, které umožňují čtení vybraných datových typů z posloupnosti bytů ve formátu big-endian. Pro práci s archivy byla navržena třída, která umožňuje snadné vytvoření těla zprávy dotazující se

na archivy. Dále také parser archivů, který ze zprávy s odpovědí obsahující více archivů dokáže jednotlivé záznamy rozdělit a vracet je rovnou v objektech nebo jako posloupnost bytů. Mezi další pomocné třídy patří třída, která obsahuje metody pro převody získaných měřených hodnot na hodnoty reálné. Výčet těchto převodů je znázorněn v tabulce 9.

2.3.2 Návrh výměny zpráv

Při návrhu komunikační architektury se dbalo na následné vhodné použití ve vyšší logice výsledného programu. Na obrázku 8 jsou pomocí jazyka UML znázorněny hlavní třídy umožňující výměnu zpráv. UML schéma je zjednodušeno, neobsahuje návratové a nastavovací metody privátních atributů. Idea byla taková, že se vytvoří objekt (instance třídy *Connector*), který bude umožňovat navázání TCP/IP spojení se zařízením, následně bude spojení udržovat a v případě potřeby ukončovat. Tento objekt by byl také využíván v případě odesílání a přijímání zpráv od analyzátoru. Knihovna byla navržena, aby umožňovala snadnou tvorbu zpráv a jejich následné odeslání bez nutnosti řešení vlastností protokolu KMB Long. Každá vytvářená zpráva (instance třídy *QueryMessage*) musí obsahovat tři základní informace. Adresu zařízení, identifikátor typu zprávy a tělo, pokud daná zpráva tělo obsahuje. Třída obstarávající odpověď (*AnswerMessage*) byla navržena tak, že odešle zprávu s dotazem, vyčká na odpověď analyzátoru a přijatá data uchová a částečně rozdělí. To se děje přímo při vytváření objektu a dále není možné metody pro odeslání požadavku a příjem odpovědi

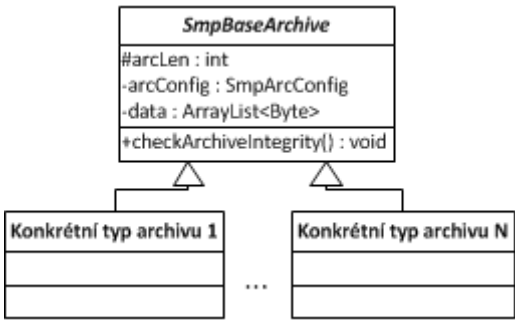


Obrázek 8: Zjednodušené schéma zpráv v knihovně

možné volat. Třída *AnswerMessage* oddělí samotné tělo přijaté zprávy, její délku, typ a kontrolní součet. Těla zpráv jsou realizována pomocí obalových tříd, ke kterým je plně objektový přístup. Všechny obalové třídy mají dvě společné vlastnosti. Musí umět získat potřebná data z přijaté posloupnosti bytů, a také nastavená data převést do posloupnosti bytů, které jsou definovány protokolem KMB Long. Z tohoto důvodu bylo navrženo rozhraní *KmbStructurable*, které přikazuje implementaci dvou zmíněných metod. Toto rozhraní musí implementovat všechny obalové třídy zpráv.

2.3.3 Archivy

Komunikační knihovna byla navržena tak, aby měla podporu pro hlavní archiv, archiv elektroměru, archiv s provozními záznamy přístroje a aby bylo možné přidávat další typy archivů. Byla navržena abstraktní třída pro jednotlivé archivy, která je znázorněna na obrázku 9. Konkrétní typy také implementují *KmbStructurable*, jako všechny ostatní třídy obalující zprávy. Pro přehledné rozlišení jednotlivých archivů byl vytvořen výčtový typ *SmpArchiveType*, který pokrývá všechny existující typy archivů.



Obrázek 9: Předek jednotlivých archivů

Pro vytváření dotazové zprávy na archivy byla navržena třída, která umožňuje snadné generování těla zprávy. Tato třída byla pojmenována *SmpArchiveQuery*. Její důležité atributy jsou definovány podle tabulky 10. Atributy je možné předat pomocí parametrů konstruktoru nebo pomocí obslužných metod. Vygenerovat tělo samotné zprávy pro archiv je možné metodou *getMessageBody*.

Dotaz na archivy může požadovat více archivů najednou. To bylo důvodem pro vytvoření třídy *SmpArchiveParser*, která umožňuje rozdělení přijaté posloupnosti bytů do jednotlivých archivů. Jednotlivé archivy lze vracet ve formě seznamu objektů (konkrétně instancí *SmpBaseArchive*) nebo jako seznam posloupností bytů. Posloupnost bytů nachází vhodné použití při tvorbě CEA souborů, kdy není potřeba přistupovat k jednotlivým atributům archivů.

2.4 Výsledná aplikace

Druhou hlavní etapou práce byla tvorba samotné výsledné aplikace. Ta byla pojmenována **ENVIS mobile**. Bylo rozhodnuto, že tato aplikace bude naprogramována pro mobilní zařízení vybavené operačním systémem Android. Při návrhu bylo potřeba zvolit, pro jakou konkrétní verzi zvoleného OS bude aplikace vyvíjena. Verze systému Android se z vývojářského hlediska označuje pomocí celočíselné identifikace zvané API Level. Přičemž platí, že čím novější verze OS, tím vyšší hodnota API Level. Aplikace určená například pro API Level 7 bude funkční i na všech vyšších platformách, ale na nižších fungovat nebude. Společnost Google pravidelně vydává statistiku, ve které zveřejňuje zastoupení jednotlivých verzí systému Android mezi koncovými uživateli. Pomocí těchto statistik lze vhodně zvolit cílovou platformu. Na obrázku 4 se na pravé straně nachází graf se zmíněnou statistikou. Pomocí těchto statistik bylo rozhodnuto, že výsledná aplikace bude vyvíjena pro verzi Android 2.1 (API Level 7). To znamená, že aplikace bude spustitelná přibližně na 99 procentech všech zařízení s OS Android, které jsou momentálně zastoupeny u koncových uživatelů.

2.4.1 Základní vlastnosti aplikace

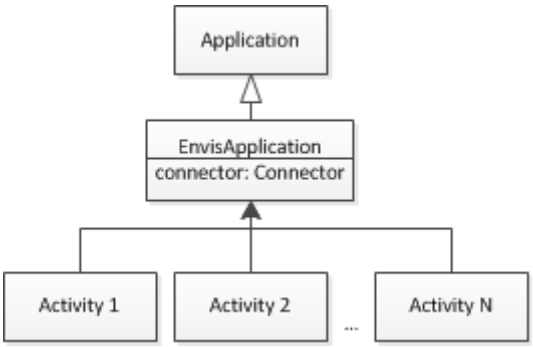
Aplikace by měla umožňovat konfiguraci instalace zařízení, konfiguraci komunikačního rozhraní, zobrazování identifikačních hodnot o analyzátoru, stahování záznamů z analyzátoru, jejich následné ukládání do CEA souborů a sledování aktuálně měřených dat a jejich vizualizaci. Vizualizace bude realizována pomocí tabulek a grafů. Aplikace byla navržena s ohledem na **multijazyčnost**, takže je možné snadno přidávat další jazyky. Cílem vývoje bylo navrhnout přehledné a intuitivní rozhraní, které bude tvořeno základními grafickými prvky.

Navržená aplikace se může nacházet ve dvou základních stavech. Prvním je situace, kdy není aplikace připojena k analyzátoru. V tomto stavu je možné přidávat a mazat IP adresy analyzátorů, jejich výběr a volit port zařízení. V druhém stavu se aplikace nachází v případě, kdy má navázané spojení s daným analyzátorem. V tomto případě je možné provádět všechny podporované operace a je možné se od zařízení odpojit.

V případě sledování aktuálně měřených dat by mělo být možné přepínat mezi tabulkou s hodnotami týkajících se proudů a napětí, tabulkou nesoucí informace o výkonech, grafem znázorňující průběh napětí a proudů na všech třech fázích a fázorovým diagramem. Ve všech těchto případech byla navržena volba změny intervalu obnovování aktuálních dat, přičemž po spuštění aplikace je tato hodnota nastavena na dvě sekundy. Všechny zmíněné grafy a tabulky jsou sjednocené pod jednou položkou v hlavním menu. Změna jednotlivých komponent je realizována pomocí ikonek v horní části aplikace. Pomocí totožného mechanismu je realizováno také nastavení, konkrétně změna mezi konfigurací instalace a konfigurací komunikace.

2.4.2 Udržení spojení v aplikaci

V momentě, kdy aplikace naváže spojení s analyzátorem, je třeba objekt udržující komunikaci předávat mezi jednotlivé aktivity aplikace. Zde existuje několik způsobů, jak takové předávání objektu realizovat. Jedním ze způsobů je předávání objektu pomocí instance *Intent*, která je předávána vytvářeným aktivitám. V případě, že se nejedná pouze o předání základních datových typů, musí instance předávané třídy implementovat rozhraní *Serializable*. Nevýhodou je nutné předávání objektu v kódu na všech místech, kde vytváříme nové aktivity. Další možností jak předávat objekty mezi aktivitami je vytvoření třídy se statickými členskými metodami. Pro toto použití by bylo vhodné použít návrhový vzor Singleton. Při návrhu aplikace byla zvolena metoda, která je založena na přetížení třídy *Application*. Instance této třídy je nadřazena všem aktivitám v aplikaci a je z nich dostupná. Toto řešení je výhodné z důvodu rychlosti, protože přístup ke statickým metodám je obecně pomalejší, než přístup k instancím tříd. Schéma realizace je znázorněno na obrázku číslo 10.



Obrázek 10: Sdílení objektu s navázaným spojením

2.4.3 Vizualizace měřených dat

Vizualizace aktuálně měřených dat je realizována pomocí několika tabulek a grafů. Při jejich navrhování bylo dbáno na zajištění přehlednosti a použitelnosti i na relativně malých displejích přenosných zařízení. První tabulka v aktuálně měřených datech zobrazuje hodnoty týkající se napětí a proudů na 3 fázích, případně na vodiči N. Tabulka byla svou strukturou zobrazovaných dat navržena podle přehledu vyskytujícího se v aplikaci ENVIS.Daq. Zobrazuje data uvedená v tabulce 3. Všechna data z této tabulky jsou přijímána ve zprávě s aktuálními daty. Pod touto tabulkou jsou také zobrazeny aktuální informace o měření jako frekvence, teplota uvnitř analyzátoru, aktivní vstupy a výstupy, podtečení, přetečení a chyby.

Tabulka 3: Přehled zobrazovaných hodnot ve výsledné aplikaci týkajících se napětí a proudů

Zkratka	**Význam**	**Jednotka**
uLL	sdružené napětí	V
uLN	fázové napětí	V
I	proud	A
uTHD	harmonické zkreslení napětí	%
iTHD	harmonické zkreslení proudu	%

Další vizualizací aktuálně měřených dat je tabulka obsahující měřená data týkající se výkonů. Většina těchto dat je také posílána ve zprávě aktuálních dat, ale některé hodnoty jsou dopočítávány. Dopočítávané hodnoty včetně vzorců jsou uvedeny v tabulce 5. Tabulka 4 popisuje všechna data zobrazovaná v tabulce výkonů.

Tabulka 4: Přehled zobrazovaných hodnot ve výsledné aplikaci týkajících se výkonů

Zkratka	**Význam**	**Jednotka**
P	činný výkon	W
Q	jalový výkon	var
S	zdánlivý výkon	VA
Pfh	činný výkon základní harmonické složky	W
Qfh	jalový výkon základní harmonické složky	var

D	deformační výkon	var
PF	skutečný účiník	--
Cos Fi	účiník základní harmonické složky	--
Pst	krátkodobá hodnota flickeru	--
Plt	dlouhodobá hodnota flickeru	--

Tabulka 5: Přepočty hodnot zobrazovaných v tabulce výkonů

Veličina	**Vzorec**
zdánlivý výkon	$S_1 = U_1 \times I_1$
deformační výkon	$D_1 = \sqrt{S_1^2 - P_1^2 - Q_1^2}$
skutečný účiník	$PF_1 = \lvert P_1 \rvert / S_1$
účiník základní harmonické složky	$\cos(\varphi)_1 = Pfh_1 / \sqrt{Pfh_1^2 - Qfh_1^2}$

Další možnost sledování aktuálních dat je realizována spojnicovým grafem, který zobrazuje průběh proudů a napětí na 3 fázích v průběhu v čase. Tento graf byl navržen tak, že existují dvě osy Y, jedna pro napětí a druhá pro proud. Proudy a napětí na jednotlivých fázích jsou barevně odlišeny. Pro tvorbu grafu bylo využito knihovny AchartEngine [1]. Ta umožňuje pro Android zařízení generovat různé druhy grafů, které disponují možnostmi přibližování dat pomocí gest na displeji a dalšími vlastnostmi. AChartEngine je šířen pod licencí Apache 2.0 License [3].

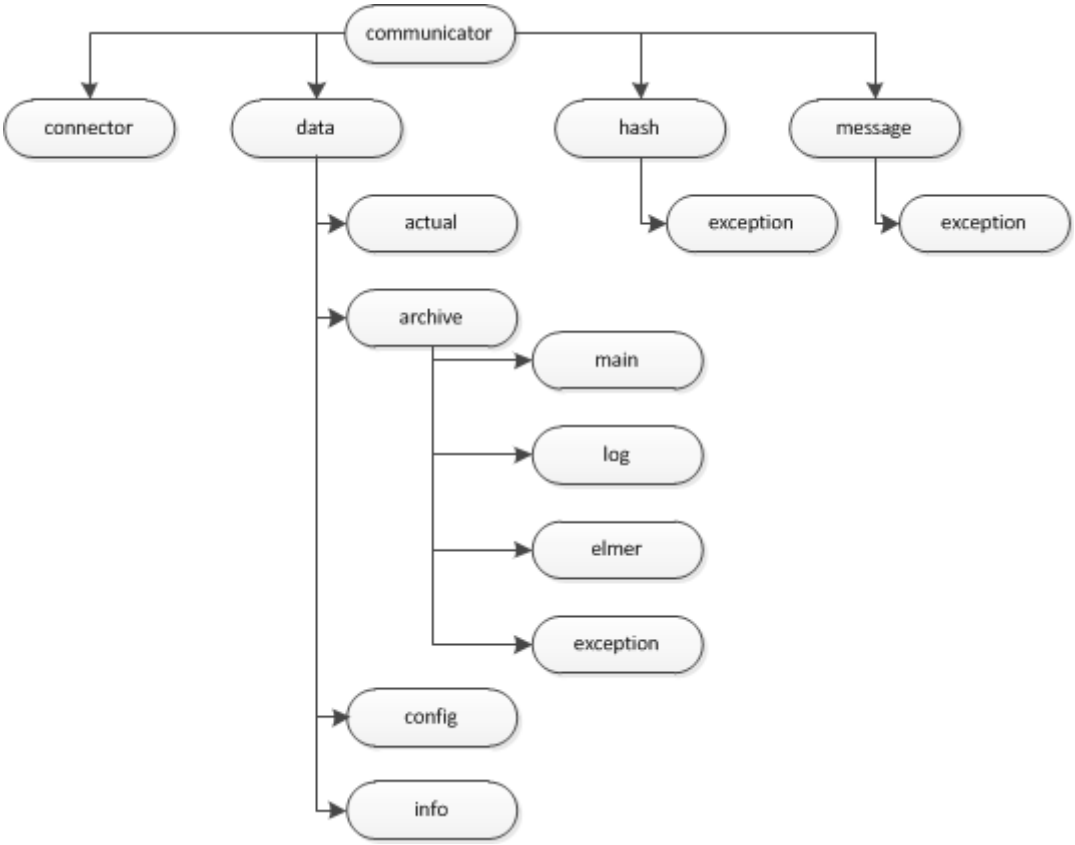
Poslední vizualizací je fázorový diagram. Komponenta fázorového diagramu byla navržena a následně implementována pomocí základních prostředků jazyka Java a Android. Navržená komponenta umožňuje zobrazovat úhly na jednotlivých fázích pro proud, napětí, zobrazuje legendu a je konfigurovatelná co se týče velikosti, barev a dalších vizuálních možností.

3 Implementace

Tato kapitola se zabývá konkrétními důležitými implementačními praktikami částí výsledné komunikační knihovny a aplikace. Dále jsou v této kapitole zveřejněny výsledky, řešení určitých problémů a celková demonstrace výsledného systému.

3.1 Komunikační knihovna

Komunikační knihovna KmbLongCommunicator byla implementována podle předem vytvořeného návrhu. Celkem tento samostatný modul obsahuje přibližně 40 veřejných Java tříd včetně rozhraní, které jsou rozděleny do čtrnácti balíků. Struktura balíků je zobrazena na obrázku číslo 11. Ke knihovně byla vytvořena kompletní dokumentace popisující chování tříd a metod, jejich návratové hodnoty a vstupní argumenty pomocí nástroje **Javadoc** [13]. Dokumentace je dostupná na dodaném CD k této práci.



Obrázek 11: Struktura balíků komunikační knihovny

3.1.1 Realizace komunikace a datových proudů

Jazyk Java obsahuje pro spojově orientovanou komunikaci, která je v komunikaci s analyzátorem požadována z důvodu TCP/IP spojení, třídu *Socket*. Třída *Socket* poskytuje možnost navázat TCP/IP spojení se serverem či serverovou částí aplikace. Komunikace následně probíhá pomocí vstupně výstupních proudů totožně, jako se pracuje v jazyce Java například se soubory. Jako třídy obstarávající vstupní a výstupní streamy byly původně zvoleny *BufferedReader* pro čtení a *PrintWriter* pro zápis. Třída *PrintWriter* ovšem nebyla vyhovující z důvodu, který je uveden a popsán v následujícím odstavci.

V počátcích vývoje komunikační knihovny KmbLongCommunicator se vyskytl problém, který byl způsoben absencí neznaménkových datových typů v jazyce Java. Protože jsou zprávy posílány jako **posloupnost bytů**, tak tento problém nastal v případě, kdy se zapisovala data na výstupní stream a zapisovaná hodnota byla větší než 127, tedy maximální hodnota datového typu *byte* v jazyce Java. Řešením tohoto problému bylo použití kombinace tříd *DataOutputStream* a *ByteArrayOutputStream*. Obě tyto třídy se nacházejí v balíku *java.io* jazyka Java. Následuje ukázka řešení.

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
DataOutputStream in = new DataOutputStream(baos);
for (int i : bytes) {
      in.write(i & 0xFF);
}
byte[] msgArray = baos.toByteArray();
outputWriter.write(msgArray);
```

3.1.2 Připojení k zařízení – třída Connector

Třída *Connector*, což je třída realizující připojení k analyzátoru, potřebuje k navázání komunikace dvě informace. To je cílová IP adresa a port analyzátoru, na kterém běží služba KMB Long. Adresa IP a port lze předat jako parametry konstruktoru při vytváření instance třídy *Connector*, nebo je lze později dodefinovat pomocí příslušných metod. Poté je potřeba zavolat metodu *connect*, která vyvolá samotné navázání spojení s analyzátorem. Po skončení komunikace je nutné zavolat metodu *closeConnection*, která uzavře vstupní a výstupní proudy a ukončí komunikaci. Následující ukázka zdrojového kódu demonstruje, jak se spojit s analyzátorem na IP adrese 192.168.0.155 a portu 2101.

```
Connector conn = new Connector("192.168.0.155", 2101);
conn.connect();
```

3.1.3 Tvorba dotazů – třída QueryMessage

Tato třída poskytuje metody nastavení jednotlivých vlastností, jako adresu zařízení, kód zprávy a tělo zprávy. Tyto vlastnosti je možné také předat v konstruktoru třídy při vytváření její instance. Délka těla zprávy a kontrolní CRC součet jsou automaticky dopočítávány. Výsledné sestavení zprávy je možné získat zavoláním metody *getMessageInBytes*, které vrací pole datového typu byte. Další možnost získání sestavené zprávy lze uskutečnit metodou *getMessageInByteArrayList*, která vrátí instanci třídy *ArrayList* obsahující jednotlivé byty zprávy.

Kód a tělo zprávy není třeba definovat ručně byte po bytu. Pro kód se v balíku *communicator.message* nachází třída *CommandCode*, která je definována jako výčtový typ a obsahuje všechny kódy zpráv, pro které je knihovna určena. *CommandCode* obsahuje navíc metody, které umožňují vrátit číselný kód zprávy podle jejího názvu. Tělo lze generovat vlastními obslužnými třídami, které jsou určené pro daný typ. Následuje ukázka vytvoření dotazu pro získání identifikačních informací.

```
QueryMessage query = new QueryMessage(1, CommandCode.GET_IDENTIFY);
```

3.1.4 Zpracování dotazu – třída AnswerMessage

Po navázání spojení s analyzátorem a vytvoření jedné či více zpráv je potřeba zprávu odeslat analyzátoru. Ten ji následně zpracuje a odešle odpověď. O odeslání zprávy, následné obdržení a základní zpracování odpovědi se stará třída *AnswerMessage*. Při přijímání dat z analyzátoru se může vyvolat výjimka *AnswerTimeoutException* a to za podmínek, že se zpráva nestáhla do časového intervalu dvou sekund a ze vstupního proudu nelze načíst další byte. Časový interval dvě sekundy se při testování knihovny projevil jako dostatečný a v případě potřeby lze ve zdrojovém kódu knihovny změnit na jinou požadovanou hodnotu. Využití výjimky *AnswerTimeoutException* je takové, že ve vyšší vrstvě aplikace bude odchycena a zpracována náležitým způsobem. Například bude uživateli zobrazen dialog s oznámením o nezdaření přijetí dat s možností vyžádání opakování operace.

3.1.5 Čtení datových typů ze zpráv protokolu KMB Long

Z důvodu neexistující podpory neznaménkových datových typů v jazyce Java je vhodné uchovávat hodnoty v o jednu úroveň vyšším datovém typu. V případě výsledné knihovny se všechna nepřevedená data uchovávají jako znaménkové byty. Z tohoto důvodu je nutné při provádění matematických a logických operací nejprve na tyto hodnoty použít operaci logického součinu s hodnotou *0xFF*. Tato podkapitola se snaží seznámit se skládáním bytových hodnot do větších datových typů, do typu float a obsluhou neznaménkového typu long (uint64) v jazyce Java.

V případě, že je přijatá hodnota znaménkového typu, tak ji stačí uložit do identického typu. To znamená, že pokud přijmeme 4 byty, které mají vcelku vyjadřovat sint32, tedy znaménkový typ integer, tak je možné ji uložit do datového typu int v jazyce Java. V případě neznaménkové veličiny je také možné ukládat hodnoty do stejně rozsáhlého datového typu, ale bylo by třeba při jakékoli matematické operaci provádět navíc logický součin, aby byla hodnota převedena na neznaménkovou. Zde nastává problém u typu long. Z tohoto důvodu je vhodnější používat o řád větší datové typy. Konverze bytů do požadované hodnoty lze realizovat pomocí operací sčítání, bitového součinu a bitového posunu. Následující příklad demonstruje zmíněné složení hodnoty znaménkového typu int z pole čtyř bytů, za použití endianity big-endian.

```
int result, i = 0;
for (int j = 24; j >= 0; j -= 8) {
    result += (bytes[i++] & 0xFF) << (j));
}
```

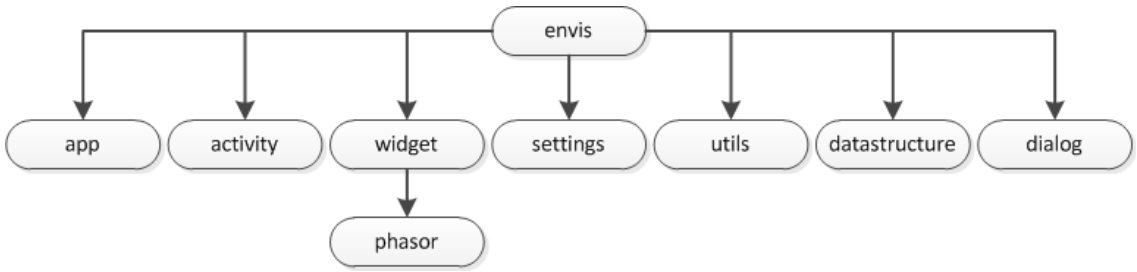
Dalším problémem byla potřeba vyřešit složení hodnot s plovoucí desetinou čárkou. Analyzátory SMPQ používají pro tyto hodnoty datový typ float, který je čtyřbytový a splňuje standard IEEE 754 [11]. Float je v jazyce Java 32 bitový, jeden bit je pro určení znamínka, dalších 8 bitů definuje exponent a zbylých 23 bitů mantisu. Jazyk Java obsahuje obalovou třídu datového typu float s názvem *Float*, která se nachází ve standardním balíku jazyka *java.lang*. Tato třída vlastní statickou metodu *intBitsToFloat*, jejímž vstupním parametrem je hodnota typu int a návratovou hodnotou je float. Převedení 4 bytových hodnot na float pomocí této metody není problém.

Nejprve je nutná konverze do datového typu int. Tato konverze je znázorněna a popsána nad tímto odstavcem. Složená hodnota int se poté použije jako vstupní parametr zmíněné metody *intBitsToFloat* a návratem metody je získána hodnota typu float.

Největší překážkou v ukládání a skládání hodnot posílaných analyzátorem je neznaménkový long tedy uint64. Pro operace s takto velkými čísly nemá Java přímou podporu na úrovni datových typů. V balíku *java.math* se nachází třída *BigInteger*, která se doporučuje používat pro uchovávání větších čísel, než je schopný pokrýt datový typ long. Tato třída obsahuje množství metod pro matematické a logické operace s uchovávanými čísly. Třída je v knihovně používána pro uchovávání časů záznamů, což je jediná hodnota (ze zatím podporovaných), která je kódována do neznaménkového typu long.

3.2 Výsledná aplikace ENVIS mobile

Aplikace ENVIS mobile byla implementována pro systém Android 2.1. Tato verze systému Android nedisponuje tolika widgety a implementovanými třídami jako novější verze operačního systému Android. Z tohoto důvodu musela být pro některé funkce zavedena vlastní řešení. Celkem je v aplikaci implementováno přes třicet tříd, které se nacházejí ve struktuře osmi balíků, které jsou znázorněny na obrázku číslo 12. Ukázkové snímky obrazovky jsou znázorněny v příloze Příloha A.



Obrázek 12: Struktura balíků aplikace ENVIS mobile

3.2.1 Oddělení zdrojů od zdrojového kódu

Systém Android se ve své filozofii snaží oddělovat samotný zdrojový kód od dalších zdrojů, jako jednotlivé definice vzhledů nebo použitých řetězců. Tyto definice jsou ukládány ve formátu XML. Všechny soubory s definicemi

a konfiguracemi se nacházejí v základním adresáři projektu ve složce *res*. V případě výsledné aplikace byla tato filozofie dodržena a všechny definice vzhledů, dialogů a řetězců jsou psány ve značkovacím jazyce XML a dle těchto standardů.

3.2.2 Multijazyčnost aplikace

Možností, jakým způsobem implementovat podporu více jazyků do jedné aplikace, existuje více. Mezi možná řešení lze považovat implementace vlastního řešení. Android ovšem obsahuje vlastní lokalizační mechanismus, který umožňuje přidávání lokalizací a jejich následnou automatickou volbu podle aktuálního nastavení systému. Vždy je jeden jazyk považován za primární. Ten je vždy zvolen, pokud není v aplikaci dostupná lokalizace pro aktuální jazyk systému. Primární jazyk je uložen v souboru *strings.xml*, který se nachází ve složce *values* ze složky zdrojů. Další lokalizace se ukládají do téhož souboru, ale do patřičné složky. Ta je ve složce zdrojů, konkrétně s názvem *values-xx*, kde *xx* vyjadřuje jazyk podle dvouznakové jmenné reprezentace jazyků podle ISO 639-1 [16]. Do aplikace ENVIS mobile byly implementovány v aktuální verzi tři jazykové lokalizace. Primárním jazykem je **angličtina**, další podporované jsou **čeština** a **chorvatština**.

3.2.3 Ukládání IP adres v aplikaci

Výsledná aplikace umožňuje přidávání, volbu a mazání jednotlivých IP adres analyzátorů. Tato volba byla implementována z důvodu jednodušší správy analyzátorů, respektive byla touto vlastností odebrána nutnost zadávat IP adresu analyzátoru vždy při pokusu o spojení. V prvních fázích vývoje aplikace bylo ukládání implementováno pomocí textového souboru, ve kterém se jeden řádek rovnal jedné adrese analyzátoru. Později byla tato možnost nahrazena XML souborem. XML forma uložení byla zvolena, protože je flexibilnější a v dalších verzích aplikace se mohou přidávat další atributy, které by umožňovaly například uchovávat pojmenování jednotlivých analyzátorů. XML soubor s uloženými adresami dodržuje následující formát.

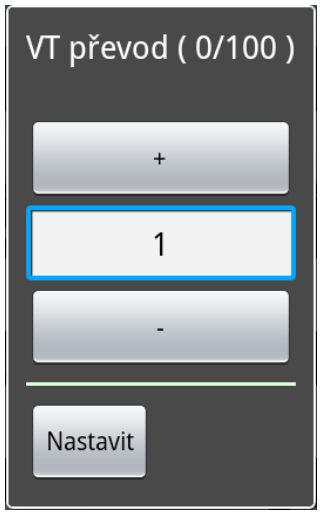
```
<?xml version="1.0" encoding="utf-8"?>
<addresses>
      <address>
            <ip>XXX.XXX.XXX.XXX</ip>
      </address>
</addresses>
```

Pro práci s XML dokumenty je ve výsledné aplikaci využíváno reprezentace DOM [7]. Jako programovací platforma byla zvolena Android API 7, která neobsahuje potřebné obslužné knihovny pro textový zápis XML dokumentu. Z tohoto důvodu bylo potřeba vytvořit metodu, která tuto práci bude automatizovat. Metoda byla pojmenována *getStringFromNode* a funguje na postupném procházení uzlů celého dokumentu a ty předává do textového výstupního řetězce. Od platformy API Level 8 je již dostupná třída *Transformer* v balíku *javax.xml.transform*, která dokumenty DOM umí převádět do formátovaného textového řetězce.

3.2.4 Vytvořené komponenty

Při vývoji aplikací pro platformu Android existuje možnost používat velký počet různých grafických a interaktivních komponent, takzvaných widgetů. Widgety jsou typicky potomci třídy *View*. Tyto komponenty lze tvořit vlastní nebo používat ty, jež jsou obsaženy v rozhraní OS Android. Pro aplikaci bylo nutné vytvořit dva vlastní widgety. Jeden pro snadnou a bezpečnou konfiguraci číselných hodnot a druhý pro vizualizaci fázorového diagramu.

Pro číselnou konfiguraci hodnot byl vytvořen widget, který byl nazván *NumberPicker*. Filozofií se podobá komponentně pro nastavování času a data, který je obsažen v systému Android. Widget *NumberPicker* je dostupný i přímo v systému Android, ale až od platformy API Level 11. Z tohoto důvodu byla uskutečněna vlastní implementace. Tento widget umožňuje nastavovat číselnou hodnotu v určitém intervalu. Nastavení lze provádět pomocí dvou tlačítek, nebo pomocí zápisu na klávesnici konkrétního zařízení. Zadávat lze pouze celá čísla. Komponenta vznikla pomocí odvození třídy *LinearLayout* a byly jí doprogramovány další potřebné vlastnosti. Vizuální vzhled komponenty je znázorněn na obrázku 13.



Obrázek 13: Komponenta pro číselnou konfiguraci hodnot - NumberPicker

Druhým vlastním navrženým a implementovaným widgetem je komponenta fázorového diagramu. Tato komponenta umožňuje grafický přehled o měřených proudech a napětí v jejich závislosti na úhlu. Komponenta je potomkem třídy *View* a byla nazvána *PhasorDiagramView*. Data jí lze předávat pomocí obslužných metod, jejichž vstupním parametrem je buď instance třídy *VoltagePointer* nebo *CurrentPointer*. To jsou třídy, které umožňují uchovávat pár dat hodnoty dané veličiny a úhlu. Tyto třídy jsou potomkem třídy *AbstractPointer*. Fázorový diagram byl implementován tak, že funguje bez problému na jakkoli velkém displeji. Pomocí obslužných metod lze definovat barvy všech jednotlivých částí komponenty. Celá komponenta funguje na principu vykreslování na plátno (Canvas). Rozsah hodnot se mění dynamicky podle nejvyšší hodnoty v dané skupině. Jednotlivé polohy bodů ve fázorovém diagramu jsou počítány pomocí následujících vztahů.

$$x = stredGrafuX + delka * \sin(uhelHodnoty)$$
$$y = stredGrafuY - delka * \cos(uhelHodnoty)$$

Středy grafů jsou souřadnice středu, délky jsou normalizované hodnoty délky dané zobrazované veličiny a úhlem hodnoty je nastavený úhel. Ukázka vizualizace výsledné komponenty fázorového diagramu je znázorněna v příloze Příloha A.

3.2.5 Realizace struktury aplikace

Aplikace byla implementována způsobem, ve kterém lze rozlišit dva hlavní stavy. Za první lze považovat aktivity, které slouží informačně nebo k připojení se k analyzátoru. V tomto stavu není nikdy navázána komunikace s analyzátorem. V druhém stavu se aplikace nachází, pokud došlo k navázání komunikace. Zde je v nejnižší vrstvě (menu s možnostmi analyzátoru) ignorováno tlačítko zpět, tudíž nelze přejít do předchozího stavu. Přejít lze pouze po odpojení, pro které je v menu přidružená položka. Tlačítko zpět je ignorováno pomocí přetížení metody naslouchající stisku tlačítek a kláves. Konkrétní implementace je znázorněna v následující ukázce kódu.

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        return false;
    }
    return super.onKeyDown(keyCode, event);
}
```

3.2.6 Reakce aplikace na chyby komunikace

Reakce na chyby komunikace je velice rozsáhlou problematikou. Tyto chyby mohou vznikat na mnoha místech a jejich kompletní ošetření není snadnou záležitostí. V případě aplikace mohou nastat chyby, například pokud se ztratí signál od wi-fi přijímače. Jinak zařízení reaguje na ztrátu či přerušení signálu při komunikaci přes mobilní připojení. Také se může stát, že z nějakého důvodu přestane komunikovat analyzátor, nebo bude komunikovat nesprávně. Scénářů je velké množství.

Ideálním řešením by byla neustálá kontrola socketového spojení. Zde nastává problém v tom, že v jazyce Java se instance třídy *Socket* chová po připojení jako připojená i v případě, kdy dojde k selhání signálu. Tato instance se považuje za uzavřenou, respektive že obaluje uzavřenou komunikaci jen v případě, kdy dojde ke korektnímu ukončení spojení pomocí zpráv FIN či RES.

Výsledná aplikace po výskytu chyby v komunikaci zůstává aktivní a stále se pokouší komunikovat. Samotný systém s upozorněními o chybách komunikace pro uživatele aplikace nebyl zatím implementován. V tomto směru by bylo možné aplikace dále rozšiřovat a vylepšovat, protože s lepším odchytávání chyb by bylo také třeba řešit reakci na chyby přenosu a tak dále.

3.2.7 Implementované stahování archivů

Stahování archivů bylo implementováno podle předem vytvořeného návrhu. K samotnému zabalování jednotlivých záznamů a archivů do CEA souboru je využíváno zmíněné knihovny od Adama Smolíka. Samotná aktivita pro volbu stahovaných archivů a omezení časů byla vytvořena vlastní. Její náhled je znázorněn v příloze Příloha A. V příloze Příloha C je tabulka časových srovnání stahování archivů pomocí výsledné aplikace ENVIS mobile a aplikace ENVIS.Daq. ENVIS.Daq dosahuje lepších časových výsledků. To je způsobeno rychlejším CPU v počítači a lepším algoritmickým řešením stahování archivů. Toto je část aplikace, kterou lze dále

vylepšovat. Měření časů v aplikaci ENVIS mobile probíhalo na telefonu Samsung Galaxy S s OS Android 2.3.3. Všechna jednotlivá stahování a zabalování byla provedena pět krát a následně z nich byla vypočtena střední hodnota, která je uvedena v tabulce.

3.2.8 Paměťová náročnost a odezva aplikace

Výsledná aplikace je určena pro mobilní zařízení a z tohoto důvodu byl při její implementaci kladen důraz na nízkou paměťovou náročnost. Proto bylo grafické uživatelské rozhraní implementováno pomocí základních grafických a ovládacích prvků systému Android. V tabulce 6 je znázorněna průměrná paměťová náročnost aplikace v jednotlivých aktivitách. Průměry byly počítány jako střední hodnota z pěti měření v čase jedné minuty spuštění aktivity. Měření probíhalo na mobilním telefonu Samsung Galaxy S s operačním systémem Android ve verzi 2.3.3. Spotřeba paměti bude pokaždé trochu odlišná, protože správu paměti obstarává garbage collector.

Tabulka 6: Průměrná paměťová náročnost vybraných aktivit aplikace

Aktivita	**Průměrná paměťová náročnost [MB]**
Výběr IP a portu k připojení	5,82
Tabulka aktuálních dat – U, I	7,74
Graf aktuálních dat – U, I	8,87
Graf fázorového diagramu	8,66
Stahování archivů – měřeno při stahování	8,63
Nastavení instalace	7,28

Důraz byl kladen také na odezvu aplikace. Odezva je v aplikaci téměř vždy okamžitá, protože všechny komunikační a výpočetní úlohy se konají v samostatných vláknech, které nezatěžují vlákno obstarávající grafické uživatelské rozhraní.

4 Závěr

S výslednou aplikací, která byla v rámci této bakalářské práce navržena a implementována pro mobilní platformu Android může její uživatel zobrazovat identifikační informace, provádět implementované konfigurace analyzátoru, sledovat aktuálně měřená data a stahovat archivy ukládané do CEA formátu. Veškerá komunikace s analyzátory je realizována bezdrátově. Vyvinutá aplikace může jejím uživatelům poskytnout pohodlnou, ale omezenou variantu aplikace ENVIS.Daq od společnosti KMB systems. Hlavní výhodou je, že uživatel s sebou, například při zapojování, konfiguraci či kontrole analyzátoru, nemusí mít osobní počítač či notebook, ale vystačí si s jakýmkoliv mobilním zařízením s operačním systémem Android. Může jít buď o tablet, PDA, mobilní telefon či netbook.

Aplikaci je možné (a bylo by vhodné) dále rozšiřovat. Její návrh a implementace byly realizovány tak, aby byla zajištěna další rozšířenost. Aplikace zdaleka nepokrývá všechny své možnosti a v budoucnu ji bude možné vylepšovat například v rámci dalšího projektu či diplomové práce. S postupem času by mohla být vylepšena až na úroveň blížící se aplikaci ENVIS.Daq.

Možnost stahování archivů implementována je, ale pouze pro hlavní archiv, archiv s logy přístroje a pro archiv elektroměru. V dalších verzích aplikace by bylo vhodné zavést rozšíření jak do aplikace, tak do komunikační knihovny pro stahování dalších typů archivů. Možným vhodným rozšířením aplikace a komunikační knihovny by byly další možnosti nastavování analyzátoru. V aktuální verzi lze nastavovat analyzátor pomocí zpráv obsluhující nastavení instalace, jména záznamu a měření a nastavení komunikace, což je část zprávy TConfig. Tuto zprávu by bylo vhodné v budoucnu doimplementovat celou. Tím by se získaly další možnosti konfigurace, konkrétně například typ zobrazení na display, jazyk analyzátoru a časování nastavení. Co se týká aktuálně měřených dat, je možné přidat do aplikace další grafy a tabulky, ve kterých by bylo možné zobrazovat hodnoty napětí a proudů harmonických. Případně implementovat grafy, které budou vizualizovat tvary vln průběhů proudů a napětí.

Hlavními omezeními aplikace je obsluha chyb komunikace na nízké úrovni. V tomto směru by bylo vhodné výslednou aplikaci také upravit a doprogramovat řešení obstarávající odchytávání chyb a reakci na tyto chyby.

Samotné řešení této práce a vývoj výsledné aplikace přineslo jejímu řešiteli mnoho zkušeností s vývojem graficky zaměřených uživatelských aplikací pro mobilní platformy, konkrétně pro operační systém Android. Za další získané zkušenosti lze považovat seznámení s nestandardním komunikačním protokolem využívaném v průmyslově používaném zařízení. Další dobrou zkušeností byl vývoj okrajově založený na práci v týmu, přesněji na komunikaci s vedoucím práce a konzultantem, případně s kolegou, který ve své práci řešil knihovnu pro tvorbu CEA souborů.

Seznam použité literatury

[1] Achartengine: Charting library for Android. *Google Code* [online]. [2011] [cit. 2012-05-11]. Dostupné z: http://code.google.com/p/achartengine

[2] *Android plugin for Netbeans* [online]. © 2010 [cit. 2012-05-12]. Dostupné z: http://kenai.com/projects/nbandroid

[3] Apache License, Version 2.0. THE APACHE SOFTWARE FOUNDATION. *The Apache Software Foundation* [online]. 2004 [cit. 2012-05-11]. Dostupné z: http://www.apache.org/licenses/LICENSE-2.0

[4] Application Platform Overview for Windows Phone. *MSDN | Microsoft Development, Subscriptions, Resources, and More* [online]. 22.3.2012 [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67a6rSVMp

[5] Big and Little Endian. *Department of Computer Science* [online]. 2003 [cit. 2012-05-11]. Dostupné z: http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Data/endian.html

[6] *Dalvik Virtual Machine* [online]. © 2008 [cit. 2012-05-11]. Dostupné z: http://www.dalvikvm.com

[7] Document Object Model (DOM). W3C. *World Wide Web Consortium (W3C)* [online]. © 1997-2005 [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67aLLslUX

[8] FLUKE. *Android Application software installation guide*. [2010]. Dostupné z: http://assets.fluke.com/software/PowerQuality/Install_FPV_Archos.pdf

[9] Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent. GARTNER. *Technology Research | Gartner Inc.* [online]. 2011 [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67a5dctxE

[10] GOOGLE. *Android Developers* [online]. [2008] [cit. 2012-05-11]. Dostupné z: http://developer.android.com

[11] IEEE 754: Standard for Binary Floating-Point Arithmetic. *IEEE-SA - Working Group* [online]. [2009] [cit. 2012-05-15]. Dostupné z: http://grouper.ieee.org/groups/754

[12] IOS Developer Library. *Apple Developer* [online]. © 2012, 2012-01-24 [cit. 2012-05-11]. Dostupné z: https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/Introduction/Introduction.html

[13] Javadoc Tool. ORACLE. *Oracle* [online]. [2000] [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67aKYdktG

[14] KMB SYSTEMS. *KMB Systems s.r.o. - KMB systems* [online]. © 2011 [cit. 2012-05-11]. Dostupné z: http://kmb.cz

[15] KMB SYSTEMS. *Manuál SMV, SMP a SMPQ* [online]. Liberec, 2011 [cit. 2012-05-11], 67 s. Dostupné z: http://www.webcitation.org/query?id=1336741551790744

[16] Language Codes according to ISO 639-1. *MathGuide* [online]. © 1997-2000 [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67aL1H9kj

[17] Lesson: Packaging Programs in JAR Files. *Oracle Documentation* [online]. © 1995 [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67a8PAXuH

[18] MEGA - MĚŘICÍ ENERGETICKÉ APARÁTY. *Měřicí Energetické Aparáty, a.s.* [online]. © 2007 [cit. 2012-05-12]. Dostupné z: http://www.e-mega.cz

[19] More US Consumers Choosing Smartphones as Apple Closes the Gap on Android. In: *Nielsen Wire* [online]. 18.1.2012 [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67a6NqR0F

[20] MURPHY, Mark L. *Android 2 : Průvodce programováním mobilních aplikací*. 2011. Brno : Computer Press, a. s., 2011. 375 s. ISBN 978-80-251-3194-7.

[21] ORACLE. *Java SE Documentation at a Glance* [online]. [2009] [cit. 2012-05-11]. Dostupné z: http://www.oracle.com/technetwork/java/javase/documentation/index.html

[22] Profil korporace Fluke. FLUKE. *Fluke* [online]. © 1995 - 2012 [cit. 2012-05-11]. Dostupné z: http://www.fluke.com/Fluke/czcs/about/korporace/default.htm

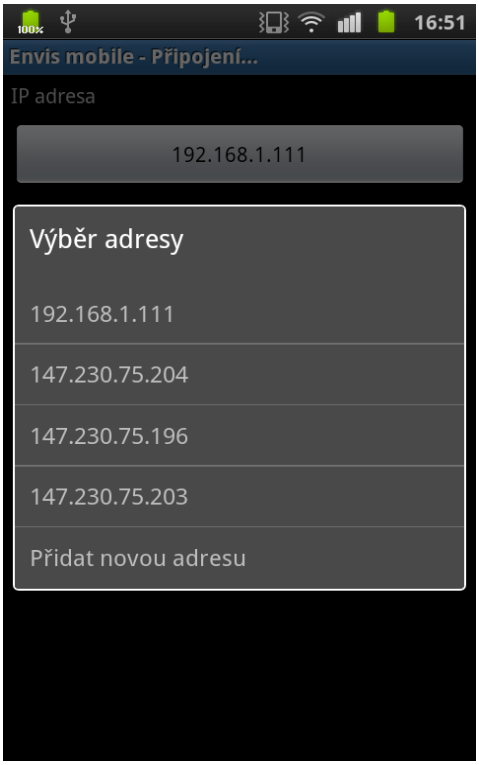
[23] Request-response. *Wikipedia* [online]. 3.10.2011 [cit. 2012-05-11]. Dostupné z: http://en.wikipedia.org/wiki/Request-response

[24] Symbian platform. NOKIA. *Nokia Developer* [online]. © 2012 [cit. 2012-05-11]. Dostupné z: http://www.webcitation.org/67a7ibQMl

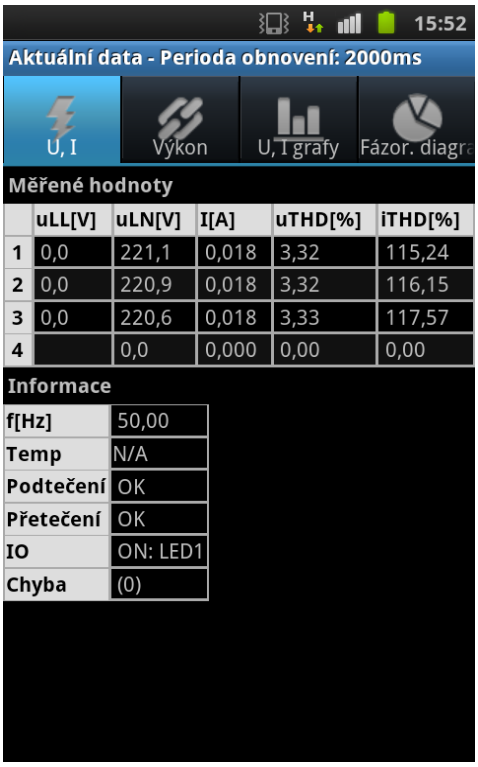
[25] THE APACHE SOFTWARE FOUNDATION. *Apache Ant* [online]. [2000], 2012-04-20 [cit. 2012-05-11]. Dostupné z: http://ant.apache.org

Příloha A

- Výběr IP adresy analyzátoru pro připojí

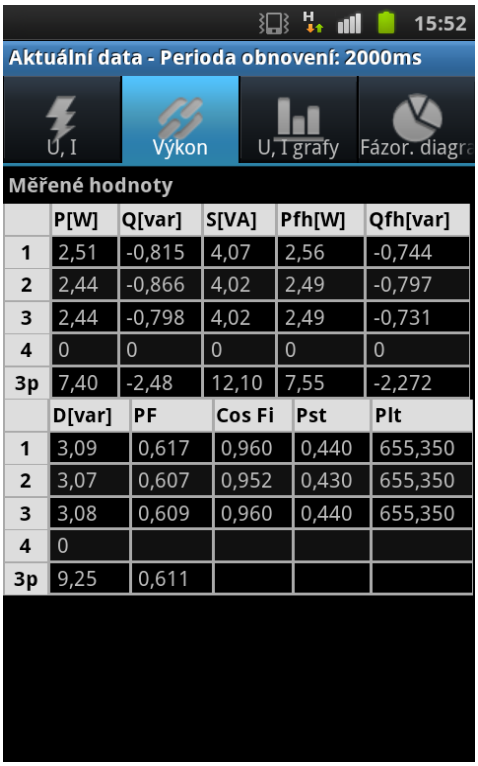


- Tabulka s aktuálními daty – U, I

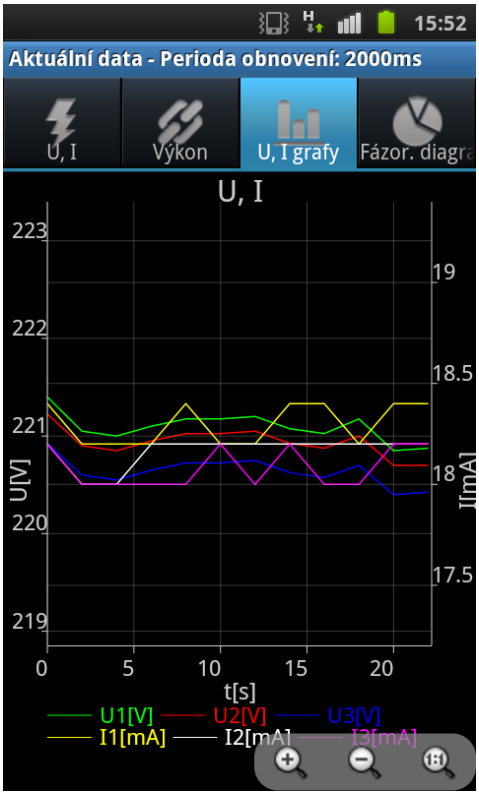


- Identifikační informace analyzátoru

- Tabulka s aktuálními daty – výkony

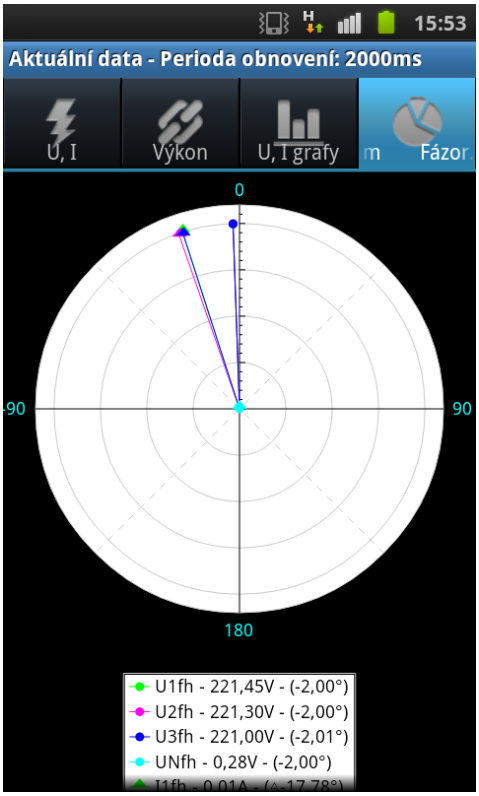


- Graf aktuálních dat proudů a napětí

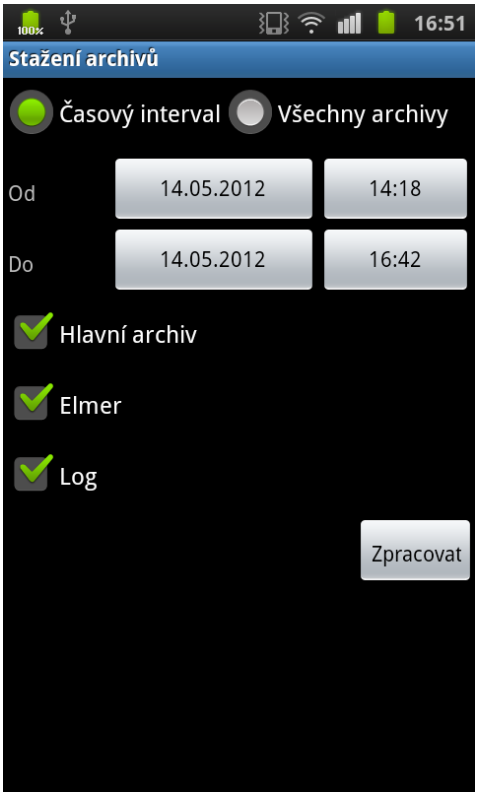


- Část okna nastavení instalace

- Fázorový diagram



- Volba stahování archivů



Příloha B

1. Zpráva Identify

Kód této zprávy je *0x01*. Jedná se o jednu z nejjednodušších zpráv protokolu KMB Long. Tato zpráva přenáší data obsahující základní informace a identifikační vlastnosti daného analyzátoru . Mezi přenášená data patří číselné označení analyzátoru, jeho typ, rodina, do které patří, verze softwaru, hardwaru a bootloaderu, odemknutelné vlastnosti od výroby a adresa zařízení.

2. Zpráva Notebook

Zprávy Identify a Notebook se podobají, protože obě nesou pouze několik málo informací o analyzátoru, přičemž se tyto informace netýkají měřených hodnot. Hlavním rozdílem mezi těmito zprávami je možnost změny informací v analyzátoru. Zpráva Indentify je pouze informativní, tedy nám může pouze poskytovat definovaná data, zatímco zpráva Notebook lze poslat analyzátoru a ten si podle ní dané atributy přenastaví. Kód zprávy pro stažení informací z analyzátoru je 0x1F a kód pro odeslání nastavení dat do analyzátoru je 0x20. Notebook umožňuje získat a nastavit vlastnosti objekt a jméno záznamu. Objekt je v tomto případě synonymem k názvu zařízení a jméno záznamu k názvu daného měření. Oba tyto atributy jsou textové řetězce, které mohou mít maximální délku 32 znaků a jsou zakončeny nulou. Znaky jsou kódovány pomocí ASCII tabulky. V případě zresetování analyzátoru do počátečního nastavení jsou všechny byty nastaveny na 255, tedy na 0xFF. V tomto případě jsou řetězce v aplikaci ENVIS či ENVIS mobile převedeny na textový řetězec „Default“.

3. Zprávy InstallConfig a TConfig

Zpráva InstallConfig umožňuje nastavovat důležité vlastnosti týkající se měřené energie. Zpráva typu TConfig umožňuje číst a definovat mimo jiné nastavení ohledně komunikace analyzátoru, zobrazení, jazyk, zámek přístroje, časová nastavení či nastavení flickeru.

Pro správné měření analyzátorem je nejprve nutné provést nastavení definice měření. Jedná se například o definici frekvence, nominálního napětí, výkonu a dalších vlastností. Některé tyto vlastnosti je také nutné využívat u výpočtu reálných měřených hodnot například v aktuálních datech nebo datech z archivů. Kódy této zprávy jsou *0x26* a *0x27*. Konkrétně *0x26* pro přečtení nastavení instalace a *0x27* pro nastavení vlastností nastavení instalace. Následující seznam demonstruje všechna nastavení analyzátoru, které jsou posílány v této zprávě.

- Napěťové převody transformátoru (zvlášť pro fáze a pro vodič N)
- Proudové převody transformátoru (zvlášť pro fáze a pro vodič N)
- Frekvence
- Metoda měření
- Nominální napětí
- Nominální výkon

V případě, že vstupní hodnoty napětí nejsou měněny pomocí napěťového transformátoru, jsou příslušné proměnné nastaveny na hodnotu *0xFFFF*. Hodnoty nastavení proudových převodů transformátorů mají také vyhrazenu posloupnost dvou bytů. Ta se ovšem dělí na 2 části. První částí je samotná primární hodnota převodu a druhou částí, která může nabývat pouze 0 a 1 (reálně představující hodnoty 1 a 5), která definuje sekundární hodnotu převodu. Sekundární hodnota převodu je definována pomocí nejvýznamnějšího bitu dané proměnné. Zbylých 15 bitů určuje poměr primární hodnoty převodu transformátoru. Při dalších výpočtech je hodnota primární dělena hodnotou sekundární. Frekvence může nabývat typicky hodnot 50 a 60. To jsou charakteristické hodnoty frekvence v hertzích používané ve světě. Přestože frekvence může nabývat pouze dvou hodnot, je uchovávána v dvoubytové proměnné. Tato hodnota by přitom mohla být bez problému kódována pouze jedním bytem. Metodě měření je ve zprávě InstallConfig vyhrazen jeden byte. Může nabývat hodnot 2, 3 a 5, přičemž

2 definuje metodu měření jako hvězdici, 3 jako trojúhelník a 5 značí zapojení včetně vodiče N. Hodnoty nominálního napětí a výkonu jsou posílány jako hodnota float, tedy pomocí 4 bytů.

4. Zpráva aktuálních dat

Jednou z hlavních možností SMPQ analyzátorů je sledování aktuálních dat, tedy hodnot právě měřených veličin. Ve zprávě aktuálních dat se ale nepřenášejí jen hodnoty měřených veličin, ale také, mimo jiné, informace o vzniklých chybách při měření, přetečení, podtečení, teplotě zařízení a informace o právě aktivních vstupech a výstupech analyzátoru. Kód zprávy pro vyžádání aktuálních dat je *0x3A*. Do požadavku této zprávy je třeba vkládat masku, které definuje, jaké měřené veličiny se mají od analyzátoru přijmout. To je velice vhodná vlastnost, především z pohledu na úsporu síťového provozu a komunikace. Často je totiž situace taková, že je zbytečné posílat všechna aktuální data a je požádáno jen o chtěná cílová data. Tato maska je definována pomocí čtyř bytů. Každý bit z této hodnoty má svůj význam. V tabulce 7 je znázorněna definice masky aktuálních dat.

Tabulka 7: Význam bitů masky požadavku aktuálních dat

Bit(y)	Význam
0-3	fáze (1, 2, 3, N)
4	data napětí, proudu, výkonů, THDU, THDI
5	harmonické od napětí
6	harmonické od proudu
7	úhly harmonických od proudu
8	meziharmonické od napětí
9	meziharmonické od proudu
10	sudé harmonických
11-16	počet harmonických
17	vyhodnocování kvality – Plt, Pst
18	pole fázorů napětí a proudu
24-31	vlny (U1, U2, U3, U4, I1, I2, I3, I4)

V samotné odpovědi zprávy aktuálních dat se nenachází pouze vyžádaná data. Vyžádaným datům předchází několik hodnot, které informují o stavu analyzátoru, jeho chybách a dalších vlastnostech. Tato data mají konstantní délku a pořadí. Na začátku odpovědi aktuálních dat se vždy nachází maska, pro kterou analyzátor data vyhodnotil. Vždy se jedná o masku, která je analyzátoru odeslána ve zprávě požadavku. Po masce následuje hodnota o velikosti jednoho bytu nesoucí informaci o počtu změn instalačního nastavení (zprávou InstallConfig – kapitola 2.). Tato hodnota je velice významná při zobrazování aktuálních dat. Je třeba, aby byla kontrolována, a v případě její změny, což je typicky její inkrementace, požádat analyzátor zprávou GET_INSTALL_CONFIG o odeslání aktuálního nastavení instalace. Data převodů napěťových a proudových transformátorů nacházejících se ve zprávě InstallConfig jsou důležitým faktorem při přepočtu aktuálně poslaných dat na hodnoty reálně měřené.

Po předchozím bytu následuje v posloupnosti zprávy aktuálních dat dvoubytová hodnota informující o chybách analyzátoru, která má v kódu označení errCode. Mezi možné vyhodnocované chyby patří například chyba či změna frekvence, adjustace času nebo výpadek napájení přístroje v intervalu měření. Následuje další dvoubytová hodnota, která informuje o dalších možných chybách či rozsynchronizování. Dále se ve zprávě nacházejí dvě hodnoty o délce jednoho bytu, které informují o přetečení a podtečení rozsahů převodníků. Po těchto informacích následují data oznamující jaké LED a relé jsou v analyzátoru aktivní, aktuálně měřená frekvence, teplota uvnitř analyzátoru, informace o sledu fází. Po těchto informacích se ve zprávě nacházejí měřené hodnoty. V případě, kdy by byla vyžádána k odeslání všechna měřená aktuální data, by byla odeslána v pořadí, které se nachází v tabulce 8.

Tabulka 8: Odesílané veličiny v aktuálních datech včetně jejich datových typů

Skupina veličin	Veličiny	Datový typ
Fázové veličiny	fázové napětí, sdružené napětí, proudy	uint16
	činný výkon, činný výkon první harmonické, jalový výkon, jalový výkon první harmonické	float
	krátkodobý a dlouhodobý flicker (Pst, Plt)	uint16
	harmonická zkreslení napětí a proudů	uint16
Harmonické veličiny	skupiny harmonických napětí a proudů	uint16
	úhly harmonických	sint16
	skupiny meziharmonických napětí a proudu	uint16
Pole fázorů	hodnoty prvních harmonických napětí a proudů	uint16
	hodnoty uhlů prvních harmonických napětí a proudů	sint16
Vlny	kalibrace napětí a proudu	float
	tvary průběhů napětí a proudů	sint16

Měřené hodnoty jsou v případě vyžádání hodnot z více fází odesílány ve formě pole. Tato pole mají v reálném provozu většinou velikost 4 (vyhodnocení fází 1, 2, 3 a vodiče N). Přenos vln je realizován pomocí dvourozměrného pole.

Data nejsou posílána v jejich reálné hodnotě. K reálné hodnotě se je nutno dopočítat pomocí vzorců z tabulky 9. Proměnná A vyjadřuje přijatou hodnotu v aktuálních datech. Hodnoty *MTN* a *MTP* je nutné získat ze zprávy konfigurace instalace.

Tabulka 9: Vzorce pro dopočítání reálných hodnot z přijímaných dat

Výpočet napětí	Výpočet proudu	Výpočet výkonu	Výpočet frekvence, THD a flickeru
$\frac{A \cdot MTN}{40}$	$\frac{A \cdot MTP}{5000}$	$A \cdot MTN \cdot MTP$	$\frac{A}{100}$

5. Zprávy týkající se archivů

Jedna z hlavních možností analyzátorů SMPQ je archivace záznamů měřených veličin a logů. Záznamy jsou ukládány do paměti analyzátoru. Maximální počet záznamů a typ archivovaných veličin se vztahuje na konkrétní model analyzátoru a aktuální konfiguraci archivace. Archivovaná data analyzátorů je poté možné uchovávat v databázi, konkrétně v Microsoft SQL Server, nebo v CEA souborech, případně je možný export do CSV či XLS. CEA soubory jsou komprimované binární soubory, které obsahují archivované veličiny včetně nastavení a vlastností analyzátoru v době zaznamenávání. Jednotlivé položky CEA archivů jsou svou strukturou podobné se strukturou zpráv sloužících pro komunikaci protokolu KMB Long. Cílem této kapitoly je seznámení s přenosem jednotlivých použitých archivů, jejich složením, požadavkem na vyžádání daných archivů a ukázka navržených a implementovaných pomocných tříd, které se snaží práci se záznamy zjednodušit. Jedná se především o třídy určené pro snadnou tvorbu dotazu nebo pro jednoduché rozdělení přijatých dat do objektově orientované formy.

Analyzátory SMPQ umožňují archivovat záznamy do 10 typů archivů. V každém typu archivu jsou ukládána různá data, ať se jedná o samotné logy přístroje, záznamy měřených dat v určitém čase, záznamy elektroměru nebo další vlastnosti. Každý typ archivu je určen jednoznačným identifikačním číslem. Tyto identifikátory jsou důležitým faktorem při tvorbě požadavku zprávy pro získání archivu.

Důležité informace o nastavení archivace záznamů lze přečíst ze zprávy **Status**, která v sobě mimo jiné přenáší také konfigurační data hlavního archivu, tzv. arcConfig. Zpráva Status lze vyžádat od analyzátoru pomocí požadavku s kódem *0x14*. Odpověď na tuto zprávu obsahuje informace o jednotlivých typech archivů, a to konkrétně maximální počet záznamů, pozice posledního záznamu a časy prvního a posledního záznamu daného typu archivu. Na začátku odpovědi se také nachází dvoubytová hodnota obsahující informace o chybách archivace či jejího nastavení. Každý bit této hodnoty má svůj význam. První bit značí chybu v uloženém nastavení přístroje, druhy bit špatnou kalibraci analyzátoru, čtvrtý bit chybu času a sedmý bit chybu CRC v archivu. Ostatní bity jsou aktuálně neobsazeny. Posledních 33 bytů odpovědi drží

informace o hlavním archivu (arcConfig), přesněji jaké veličiny jsou momentálně v hlavním archivu zaznamenávány. Pomocí těchto dat lze také dopočítat struktura, ve které jsou poté přijata data na požadavek hlavního archivu. To je velice důležitá vlastnost, protože data hlavního archivu nemají pevnou délku a složení, tudíž by nebylo možné jednotlivé hodnoty z přijaté zprávy se záznamy z hlavního archivu přečíst. Přesná struktura této zprávy lze vyčíst ze zdrojového kódu knihovny nebo z interní dokumentace firmy KMB systems.

Zpráva pro získání archivů se nijak neliší od ostatních zpráv protokolu KMB Long. Kód zprávy pro načtení archivu/archivů je 22 v soustavě desítkové, respektive 16 v případě hexadecimálního kódování hodnot. Důležitým aspektem je tělo zprávy, které je v případě zprávy požadavku na archivy definováno pevnou strukturou, ve které se musí povinně předat čtyři argumenty. Tyto argumenty, jejich posloupnost a velikost jsou znázorněny v tabulce 10.

Tabulka 10: Struktura těla dotazu na archiv

Typ archivu	**Počáteční adresa**	**Počet archivů**	**Počet opakování**
uint8	uint32	uint16	uint8

Příloha C

Počet stahovaných archivů			**Čas stažení a zabalení do CEA pomocí ENVIS.Daq [s]**	**Čas stažení a zabalení do CEA pomocí ENVIS mobile [s]**
Main	***Elmer***	***Log***		
15	1	0	2,34	10,64
100	6	35	3,22	13,26
200	12	60	4,52	19,22
375	25	100	7,03	28,84