TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: 2612R011 Elektronické informační a řídící systémy

Klient pro zobrazování dat virtuálního měřicího přístroje v prostředí GNU/Linux

Client for displaying data from virtual measure device on GNU/Linux

Bakalářksá práce

Autor:	**Tomáš Bedrník**
Vedoucí práce:	ing. Jan Kraus
Konzultant:	ing. Tomáš Tobiška

V Liberci 21.5.2010

Zadání

1. seznamte se s možnostmi vývoje grafických aplikací v prostředí Linux pro desktopové a embedded systémy
2. navrhněte komunikační rozhraní mezi virtuálním přístrojem a navrhovaným modulem pro uživatelskou interakci
3. realizujte navržený systém demonstrujte jeho správnou funkci
4. diskutujte výhody a nevýhody zvoleného řešeni, uveďte možnosti dalšího rozvoje aplikace

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou 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 diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou 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.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum 21.5.2010

Podpis

Poděkování

Chtěl bych poděkovat především vedoucímu bakalářské práce ing. Janu Krausovi za pomoc a trpělivost při řešení problémů. Také bych chtěl poděkovat ing. Tomáši Tobiškovi za pomoc s protokolem Modbus.

Samozřejmě také rodině, za podporu ve studiu na vysoké škole.

Abstrakt

Hlavním cílem práce bylo vytvořit grafického klienta k virtuálnímu měřícímu přístroji, který simuluje Power Monitoring Device (PMD). Klient musí umožňovat běh na společném hardware s PMD i na oddělené stanici. Při vývoji bylo nutné zohlednit omezené možnosti embedded platformy, připojitelných vstupních periferií i výstupního zobrazovacího zařízení. Hlavní funkcí klienta je přehledně a srozumitelně zobrazovat měřené veličiny a konfigurační parametry, které musí umožňovat také jednoduše nastavovat.

Při práci s virtuálním přístrojem se vyskytly problémy, proto vývoj probíhal přímo s reálným PMD a ne na jeho virtuální náhradě.

Podmínkou pro tvorbu klienta byl běh na operačním systému GNU/Linux a použití některé z volně dostupných grafických knihoven. Součástí práce byl i výběr vhodného způsobu komunikace mezi PMD a grafickým klientem a jeho realizace. Cílem nebylo vytvoření kompletního klienta se zobrazováním všech dat a konfiguračních parametrů dostupných z PMD, protože to přesahuje rozsah bakalářské práce, ale pouze návrh řešení a zpracování základních dat a parametrů.

První část práce je věnovaná teoretickému úvodu do problematiky vývoje grafických klientů v operačním systému GNU/Linux a také problematice komunikace v průmyslovém prostředí. Jsou zde představeny jednotlivé stupně grafického systému v operačním systému GNU/Linux, nejběžnější grafické knihovny a základní způsoby komunikace, které se hodí pro toto použití.

V druhé části jsou popsány postupy při vývoji grafického klienta a navíc je zde představen i textový klient, který v rámci práce vznikl. Textový klient slouží pro potřeby budoucího vývoje, protože umožňuje přímý přístup ke konfiguračním proměnným připojeného PMD.

Výsledkem práce jsou dva klienti, kteří sdílí podstatnou část zdrojového kódu a s PMD dokážou komunikovat přímou výměnou XML souborů nebo pomocí protokolu Modbus. Klienti jsou navrženi objektovým přístupem, což zajišťuje jejich jednoduché rozšiřování o další funkce a vlastnosti.

Klíčová slova: klient, komunikace, grafika, Linux, embedded

Abstract

Main goal of thesis was to design a graphic client for Virtual Measure Device (VMD), vhich simulate Power Monitoring Device (PMD). Client has to be able to run on single hardware with PMD or on standalone station. It was necesary to consult limited abilities of embedded platform, connectable input devices and output display device. Main function of client is to digestedly display measured data and configuration parameters which he has to be able to edit.

During work on VMD appeared some difficulties, so the development was proceed on real PMD instead of its virtual simulation.

A requirement for the client was to run on operating system GNU/Linux and to use one of free available graphic libraries. Part of the work was also to choose suitable communication method between PMD and graphic client, and its realization. Goal wasn't to make a complete client which is able to display all measured data and configuration parameters available in PMD, as it would exceed range of bachelor thesis, but only to design a solution for processing main data and configuration parameters.

First part is theoretical introduction to problem of making graphical clients on GNU's Not Unix! (GNU)/Linux and communication in industry environment. Here are described main levels of graphical subsystem in GNU/Linux, main graphical libraries and and basic communication methods, which suits for this usage.

In second part is described the process of developing graphical client and also a text client, which was also developed, is introduced here. The text client was intended for further development, because it can simply display measured data and directly access the configuration parameters on connected PMD.

Main product of thesis are the two clients which can communicate with PMD via Modbus protocol or share XML files. Both clients share a lot of common source code and are projected with object approach to ensure simple adding of new functions and features.

Obsah

Seznam obrázků

Seznam zdrojových kódů

Seznam použitých zkratek

ARM Advanced RISC Machine

API Application Programming Interface

ASCII American Standard Code for Information Interchange

BSD Berkeley Software Distribution

CDE Common Desktop Environment

CLI Command Line Interface

DOS Disk Operating System

FSF Freee Software Foundation

FTP File Transfer Protocol

GIMP GNU Image Manipulation Program

GNOME GNU Network Object Model Environment

GNU GNU's Not Unix!

GPL General Public License

GTK+ GIMP Toolkit

GUI Graphic User Interface

HTTP Hypertext Transfer Protocol

IBM International Business Machines

IDE Integrated Development Environment

KDE K Desktop Environment

LGPL Lesser General Public License

LXDE Lightweight X11 Desktop Environment

OS Operační systém

OSI Open Source Iniciative

PC Personal Computer

PMD Power Monitoring Device

RISC Reduced instruction set computer

stdio Standard Input Output

SVG Scalable Vector Graphics

twm Timeless Windows Manager

USB Universal Serial Bus

VMD Virtual Measure Device

VMP Virtuální měřící přístroj

Xaw X Window System Athena

XML Extensible Markup Language

Xt X Toolkit Intrinsics

1 Úvod

Zadání bakalářské práce vzniklo během vývoje nové generace zařízení na sledování kvality elektrické energie označované anglickým názvem Power Monitoring Device (PMD), kterou ve spolupráci se společností KMB zajišťuje ing. Jan Kraus. Starší generace měla pouze malý monochromatický displej a ovládání pomocí několika tlačítek. Připravované přístroje budou mít výkonnější hardware a plně grafický dotykový displej.

Bylo rozhodnuto, že dojde k oddělení zobrazovací části do samostatného programu, který nejčastěji poběží na stejném zařízení, ale bude možno ho spouštět i jako vzdáleného klienta na jiném zařízení nebo počítači. Návrh takového klienta byl právě předmětem této práce. Již minulý rok, v rámci projektu [Bedrník2009], vznikla knihovna *libxmlrw* pro čtení a zápis XML souborů do struktur, kterou tento klient využívá.

Cílem práce bylo tedy vytvořit klienta, který bude sloužit jako uživatelské rozhraní PMD. Bude umět měřené veličiny zobrazit do přehledných tabulek a diagramů a bude umožňovat konfiguraci zařízení pomocí srozumitelných nastavovacích formulářů. Součástí práce je i návrh komunikačního rozhraní mezi klientem a PMD.

Protože prototyp zařízení nebyl během práce k dispozici a při práci s virtuálním měřícím přístrojem (VMP) se vyskytly problémy, probíhal vývoj v simulovaném prostředí s daty ze současné generace přístrojů.

Velká pozornost byla věnována výběru knihovny, která zásadním způsobem ovlivňuje vlastnosti výsledné aplikace i způsob programování. Bylo vyzkoušeno několik knihoven pro tvorbu grafického rozhraní a byla vybrána nevhodnější pro použití v projektu, byl vybrán vhodný způsob komunikace a celá aplikace byla realizována.

2 Teoretická rešerše

2.1 Vývoj aplikací v Linux

V operačním systému GNU/Linux je velmi oblíbený programovací jazyk C, je v něm napsáno mnoho programů a především jádro systému. Je to jazyk kompilovaný, proto je velice rychlý. Díky jeho rozšířenosti existuje velké množství knihoven pro nejrůznější použití.

Jazyk C++ vychází z C a za určitých podmínek je s ním zpětně kompatibilní. Knihovny naprogramované v C je možné použít i v C++. Přidává především podporu objektového programování, díky čemuž je možné psát rychleji a efektivněji. S C++ sdílí jeho výhody i nevýhody.

Další možností pro vývoj aplikací je použití programovacího jazyka Java, který patří mezi jazyky interpretované – překládá se do mezikódu a spouští se ve virtuálním stroji Javy. To zajišťuje jeho platformní nezávislost, je možné ho použít v operačních systémech GNU/Linux, BSD i dalších Unixech, Mac OSX, Windows a mobilních platformách. Je to jazyk silně objektově orientovaný, jednoduchý a mezi programátory oblíbený na všech podporovaných platformách.

Jazyk C# byl vytvořen firmou Microsoft pro operační systém Windows a je používán s knihovnou .NET. Vychází z jazyka Java a C++, se kterým má i podobnou syntaxi. Díky projektu Mono je nyní možné ho i s knihovnou .NET využít pod GNU/Linuxem. Jednou z největších výhod použití Mona je možnost sdílet kód z již hotových aplikací pro Windows. Nicméně jeho vývoj zaostává za vývojem .NET, proto není možné použít nejnovější verze a ani překlad podporovaných verzí se často neobejde bez problémů. Dalším problémem jsou licenční a především patentové nejasnosti.

Dalším z interpretovaných jazyků, který se ve větší míře používá pro tvorbu programů, je Python. Někdy bývá zařazován mezi skriptovací jazyky, ale je možné ho využít i pro tvorbu rozsáhlých aplikací. Množství knihoven, původně určených pro jiné jazyky, má wrappery, umožňující jejich použití v Pythonu. Výhodou tohoto jazyka je krátký a snadno čitelný kód, možnost použít objektově orientovaný, procedurální nebo funkcionální přístup k programování.

V práci jsou použity specifické knihovny, které nejsou dostupné pro jiné jazyky než C a C++ pro GNU/Linux, rozhodovalo se pouze mezi těmito dvěma jazyky. Byl vybrán C++, protože pro tvorbu Graphic User Interface (GUI) je objektový přístup vhodnější.

2.2 Grafický systém v operačním systému GNU/Linux

Základním způsobem pro zobrazování jakýchkoli dat na obrazovce je použití Framebufferu, který vykresluje data z paměti (bufferu) přímo na obrazovku.

V paměti jsou uložena data pro jeden rámec v matici po jednotlivých pixelech. Počet pixelů určuje rozlišení, počet bitů na jeden pixel určuje barevnou hloubku zobrazení. Používají se hodnoty 1 bit pro monochromatické a 8, 16 a 24 bitů pro barevné zobrazení.

Další, složitější vrstvy pro zobrazování používají na nejnižší úrovni framebuffer.

2.2.1 Textový režim

Základním režimem v GNU/Linuxu je textový režim. Především se zde používají textové programy pro příkazovou řádku a programy pro textové uživatelské rozhraní. S různými omezeními je zde možné zobrazit obrázky nebo video přímo přes framebuffer.

Příkazová řádka je uživatelské rozhraní, komunikující s uživatelem pomocí textových příkazů. Výstup programů je pouze textový a po řádcích se zobrazuje na obrazovce. Uživatel může pracovat pouze s posledním řádkem, kde je zobrazena výzva pro zadání příkazu – anglicky Prompt. Není možné používat myš, protože by zde ani nebyla k ničemu použitelná. Programovací jazyky obsahují standardní knihovny pro práci s tímto rozhraním, v C je to knihovna Standard Input Output (stdio). Často se pro označení příkazové řádky a programů, určených pro ni, používá anglická zkratka Command Line Interface (CLI).

Textové uživatelské rozhraní rozšiřuje textový režim o prvky grafického rozhraní. Používá celou obrazovku, jednoduchá menu, okna se záhlavím a okraji, vyskakovací hlášky, posuvníky a další věci známé z GUI. Je zde možné používat myš, ale programy jsou udělány tak, aby je bylo možné plnohodnotně požívat i bez ní. Obrazovka je rozdělena na matici, kde každá buňka může zobrazovat jeden znak z pevně dané množiny. Znakem se zde rozumí například American Standard Code for Information Interchange (ASCII) nebo části pozadí, rámečků, tlačítek a další části uživatelského rozhraní. Celé rozhraní je sestaveno z těchto znaků. V Linuxu ho využívají například programy Midnight Commander, Linx nebo Vim, v MS-DOS Norton Commander nebo M602. Pro programování takovýchto aplikací se používá nejčastěji knihovna *ncurses*.

2.2.2 Grafický režim

Dnešní desktopy standardně používají grafický režim. Jednotlivé programy se otevírají ve vlastních oknech, které se ovládají myší a klávesnicí. Obsahuje emulátor textového terminálu, díky čemuž je možné spouštět i programy pro příkazovou řádku nebo textový režim.

Pro programování se používají knihovny, využívající služeb X Window systému, nejčastěji GTK+ a Qt.

Grafické rozhraní v Linuxu poskytuje softwarový systém, síťový protokol a toolkit souhrnně označovaný jako X Window systém. Základním prvkem systému je X server, který má na starosti zobrazování na obrazovce a hardware – grafickou kartu, klávesnici, myš, touchpad, případně další polohovací zařízení. K X serveru se pomocí X protokolu připojují klienti – aplikace. Systém poskytuje základní framework pro tvorbu uživatelského rozhraní.

Komunikační protokol mezi X serverem a klientem – aplikací je navržený jako síťový, díky čemuž je možné spustit aplikaci na počítači bez monitoru a zobrazovat ji na jiném počítači s běžícím serverem. Komunikace probíhá po počítačové síti.

X server se stará pouze o vykreslení okna a jeho obsahu. Titulek, okraje, pohyb a změnu rozměrů okna má na starosti window manager.

Existuje více implementací X window systému. V GNU/Linux se nyní nejčastěji používá X.Org. Dříve se používal také XFree86, po změně licence se od jeho používání upustilo a dnes ho najdeme na ostatních Unix systémech, například Berkeley Software Distribution (BSD).

Window manager obstarává vykreslování okrajů okna, titulku, ovládacích tlačítek a změny jeho umístění či velikosti. České pojmenování pro tento program je okenní správce, ale používá se spíš anglický výraz. Standardním managerem je Timeless Windows Manager (twm). Existuje velké množství alternativ, nejznámější jsou kWin, používaný v Grafickém prostředí KDE, Metacity, používaný v GNOME, Xfwm z Xfce, FluxBox, Window Maker, BlackBox, které se používají samostatně, bez grafického prostředí a mnoho dalších.

Přidáváním dalších programů k window manageru vznikla desktopová prostředí, někdy také označovaná jako grafická prostředí. Přidané programy zajišťují jednotný vzhled celého prostředí – správu ikon, tapet na pozadí, písem, sady vzhledů a barev oken a ovládacích prvků. V rozsáhlejších grafických prostředí jsou i programy pro základní správu systému – tiskáren, periferií, síťových připojení tiskáren a mnoho dalších.

Dvě nejpoužívanější desktopová prostředí jsou GNU Network Object Model Environment (GNOME) a K Desktop Environment (KDE). Každé z nich využívá jinou grafickou knihovnu – Qt pro KDE a GIMP Toolkit (GTK+) pro GNOME. Obě dnes obsahují opravdu

velké množství programů a tvoří jakousi softwarovou platformu pro vývoj programů, proto už KDE od verze 4.4 používá název KDE Software Compilation. Některé programy se snaží zůstávat nezávislé na prostředí, ale už výběrem základní grafické knihovny se k některému blíží. Stále více programů proto volí cestu užší integrace s prostředím, z čehož samozřejmě plynou značné výhody jak pro programátory, tak pro uživatele. Sice je možné spouštět programy napsané pro jedno prostředí i v druhém, ale vyžaduje to mít nainstalované minimálně základní knihovny z obou, proto se tomu uživatelé snaží pokud možno vyhnout a linuxový svět se tak čím dál víc dělí na dva tábory.

Dalším, menším, prostředím je Xfce, které využívá grafickou knihovnu GTK+, ale není tak komplexní jako GNOME. Pro svoji jednoduchost a rychlost bývá nasazováno na starší počítače, ale i uživateli, kterým nevyhovuje integrační filosofie velkých prostředí a samostatný window manager jim nestačí.

Existují ještě další prostředí, která se dnes již příliš nepoužívají, jmenujme alespoň Common Desktop Environment (CDE) využívající grafickou knihovnu Motif.

2.2.3 Embedded systémy

Embedded systém je široký a značně flexibilní pojem, do češtiny se překládá jako vestavný systém, ale používá se převážně anglické pojmenování. Zpravidla se jím označují systémy, které slouží k jednomu specifickému úkolu. Tomuto úkolu bývá přizpůsoben i hardware, který může být oproti klasické PC architektuře značně zjednodušený. Běží na něm pouze nejnutnější aplikace a běžný uživatel zpravidla nemá možnost instalovat jiné aplikace ani systém nastavovat.

Typickými příklady embedded systému jsou mp3 přehrávač, bankomat nebo mobilní telefon. Jako embedded systémy se dnes označují i PDA nebo telefony s operačním systémem, které už výše zmíněnou definici příliš nesplňují.

Grafická prostředí nepoužívají ani textový režim a většinou ani klasický grafický režim. Prostředí často vypadá podobně jako klasické desktopové, ale obsahuje jen omezený počet aplikací nebo dokonce jen jednu, ovládá se pomocí dotykové obrazovky, trackballu nebo jiného polohovacího zařízení a může používat knihovny pracující s X Window systémem nebo přímo s Framebufferem.

2.3 Vývojová platforma

Při zjišťování možností vývoje budoucího grafického klienta musela být brána v úvahu především platforma, na které výsledné zařízení poběží.

Původně zvolen vývojový kit BSQUARE DevkitIDP od firmy LynuxWorks. Jako operační systém je v kitu použita distribuce BlueCat od stejného výrobce, která je určena speciálně pro real-time embedded systémy. Operační systém, označený jako real-time, je určen pro ta nasazení, kde je nejdůležitější rychlost a plynulost provozu. Systém je navržen tak, aby umožňoval přerušení probíhajícího procesu v případě, že je vyslán požadavek na proces s vyšší prioritou.

Vývojový kit obsahuje Intel PXA255 Xscale procesor z rodiny Advanced RISC Machine (ARM), který běží na 400 MHz, LCD dotykový displej, slot pro PCMCIA nebo Compact Flash kartu, maticovou klávesnici, integrovanou zvukovou kartu a další běžné vstupy a výstupy,včetně ethernetu. [LynuxWorks]

Od použití tohoto kitu se nakonec upustilo. Ještě předtím, než začal vývoj grafického klienta, byl vybrán jiný přípravek – i.MX31 od firmy Freescale. V Kitu je požuita upravená verze GNU/Linuxu, která obsahuje upravenou verzi knihovny Qt rozšířenou o ovladače dotykového displeje a dalšího hardwaru na kitu. [Freescale]

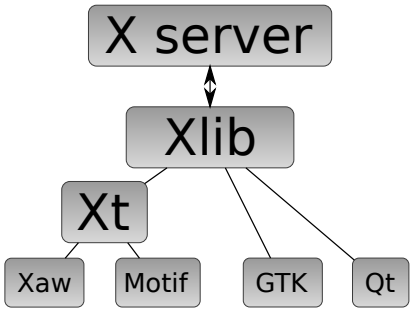
Již během vývoje grafického klienta se rozhodlo o změně vývojového přípravku na Beagle Board. Je to malá základní deska s procesorem z rodiny ARM běžící na 600 MHz s integrovanou grafickou kartou zvládající HD video. Deska obsahuje zvukovou kartu se stereo vstupem a výstupem, síťovou kartu s ethernet konektorem, DVI-D výstup na monitor, S-Video výstup, jeden USB port a slot pro MMC+, SD a SDIO karty. Napájena je z USB. Výkon tohoto kitu je dostatečný pro běh současné distribuce pro procesory ARM, například Ubuntu a přehrávání HD Videa. [BBoard]

3 Grafické knihovny

Existuje množství knihoven, určených pro tvorbu uživatelského rozhraní. Vzhledem k zadání vynecháme ty pro textový režim a budeme se zabývat pouze plně grafickými knihovnami.

Základním stavebním kamenem grafických aplikací je widget. Tímto pojmem se označuje prvek uživatelského rozhraní, se kterým přichází uživatel přímo do styku – tlačítko, posuvník, checkbox, spinbox, pole pro psaní textu a další. Jednotlivé knihovny se liší množstvím a propracovaností widgetů, ty nejjednodušší je neobsahují vůbec.

Dál se budeme zabývat pouze knihovnami pro jazyky C a C++, vynechány budou ty pro Javu a Mono, protože bylo jako programovací jazyk vybráno C++.



Obrázek 1: Schéma závislostí grafických knihoven

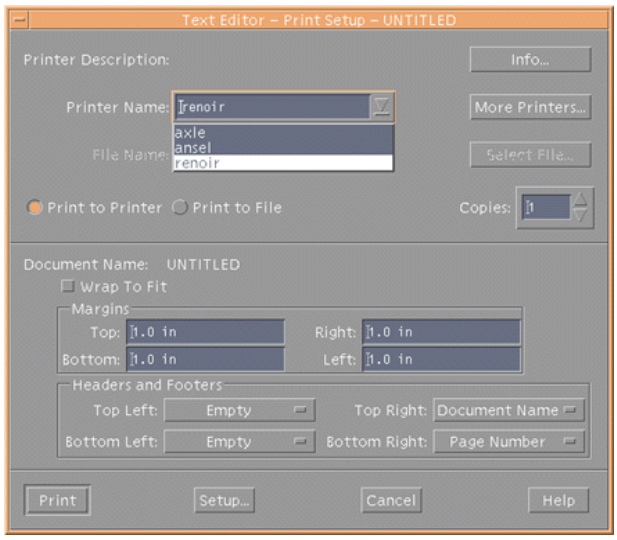
Na obrázku 1 jsou vidět vztahy mezi nejběžnějšími grafickými knihovnami, používanými v GNU/Linux.

Základní knihovnou, zprostředkovávající komunikaci pomocí X Window protokolu, je Xlib. Je napsaná v programovacím jazyce C a umí pouze to, co X window systém – vykreslovat okna, psát texty, kreslit jednoduché grafické útvary a pracovat s událostmi. Neumí používat žádné widgety. Používá se především jako základ vyšších knihoven a pro tvorbu vlastních widgetů od začátku, ale je možné vytvořit grafickou aplikaci pouze s ní.

X Toolkit Intrinsics (Xt) je knihovna zabalující funkce Xlib do lépe použitelného Application Programming Interface (API). Je napsaná v jazyce C. Oproti Xlib přidává podporu pro tvorbu a používání widgetů, ale sama žádné neobsahuje. Tato knihovna je opět určena především pro tvorbu knihoven a kolekcí widgetů. Oproti Xlib přináší jistá zjednodušení při práci s widgety.

Knihovna Motif je postavená na Xt a používá se pro tvoření GUI. Obsahuje set widgetů a definuje vlastní Motif API, které je standardizováno jako IEEE 1295. Byla navržena tak, aby vypadala podobně jako GUI operačního systému OS/2, vytvořeného společnostmi Microsoft a International Business Machines (IBM). Tvoří základ grafického prostředí CDE. Je to jednoduchá, rychlá a nenáročná knihovna, ale dnes se již příliš nepoužívá, protože je zastaralá.

Na obrázku 2 je ukázka aplikace z grafického prostředí CDE, které využívá knihovny Motif. Je z něho patrné, že poskytuje pouze základní, strohou grafiku, což je dáno její velikostí a nenáročností.

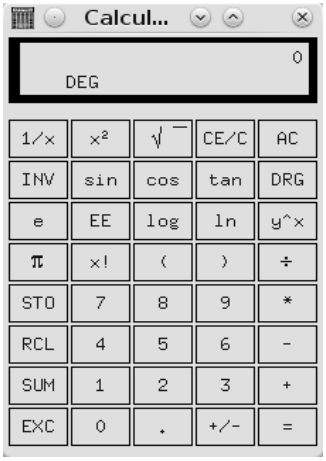


Obrázek 2: Ukázka aplikace naprogramované s použitím knihovny Motif [OpenMotif]

3.1 Xaw

Stejně jako Motif je i grafická knihovna X Window System Athena (Xaw) postavená nad Xt. Obsahuje vlastní sadu widgetů a distribuuje se společně s X Window systém, proto je dostupná na velkém množství počítačů. Oproti předchozím knihovnám se s touto již můžeme setkat v některých grafických aplikacích, především v embedded systémech. Oproti použití velkých knihoven je programování s ní komplikovanější a zdlouhavější. Naopak její výhodou je rychlost a nenáročnost na výpočetní výkon.

Na Obrázku 3 je ukázková aplikace, vytvořená za pomocí knihovny Xaw. Je to jednoduchá kalkulačka, která je součástí standardní instalace X Window systému. Z obrázku je vidět, že knihovna poskytuje pouze strohou grafiku, což pro zamýšlené použití není velký problém, protože vzhled není prioritou. Okraje a titulek okna na obrázku jsou z prostředí KDE 4.



Obrázek 3: Ukázková aplikace s použitím knihovny Xaw

Výhody :

- široce rozšířená,
- rychlá,
- nenáročná na výpočetní výkon.

Nevýhody :

- méně funkcí oproti velkým knihovnám,
- složitější použití ve větších projektech.

3.2 Gimp Toolkit

GIMP Toolkit (GTK+) je multiplatformní toolkit pro tvorbu GUI. Je napsaný v C, ale existují wrappery pro použití ve velkém množství dalších jazyků, například C++, C#, Python, Perl, Ruby nebo PHP. API bylo původně navrženo jako neobjektové, což se projevilo i při tvorbě wrapperu do C++, který není pro použití v objektově orientovaném kódu tak jednoduchý a intuitivní jako knihovny vytvářené jako objektové od začátku.

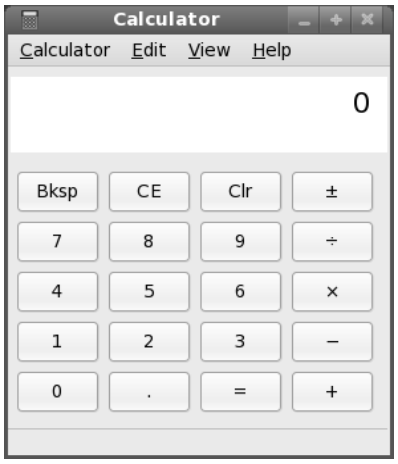
Je dostupný pro mnoho platforem – Windows, OSX, GNU/Linux a další Unix systémy. Existuje projekt GTK on DirectFB, který umožňuje použití GTK+ v embedded systémech – bez X Window systému přímo nad framebufferem. Tvoří základ grafického prostředí GNOME, Xfce a LXDE.

Oproti předchozím jednoduchým knihovnám je GTK+ komplexní toolkit s daleko většími možnostmi použití. Podporuje různá témata a přepínání mezi nimi, je vícevláknové,

má zabudované mechanismy pro jednoduchý překlad GUI do cizích jazyků a je uživatelsky přívětivější, v neposlední řadě i lépe vypadá.

O vývoj se stará open source komunita, ale podílí se na něm mnoho vývojářů, placených společnostmi, využívajícími GTK+ ve svých produktech. [GTK]

Na obrázku 4 je ukázková aplikace z prostředí GNOME , která využívá GTK+. Oproti předchozím knihovnám je vidět použití moderního vzhledu ovládacích prvků. Pomocí velkého množství veřejně dostupných témat je možné upravit vzhled widgetů globálně v celém systému.



Obrázek 4: Ukázková aplikace s použitím knihovny GTK+

Výhody :

- komplexní,
- jednoduché použití,
- vhodná i pro velké projekty.

Nevýhody :

- větší náročnost na systémové prostředky.

3.3 Framework Qt

Druhou velkou knihovnou, používanou pro tvorbu GUI, je Qt. Na rozdíl od předchozích knihoven a toolkitů je Qt komplexní framework, obsahující knihovny pro mnoho dalších věcí – například jádro internetového prohlížeče WebKit, SQL server, knihovnu pro přístup k vzdáleným serverům nebo tvorbu Scalable Vector Graphics (SVG) a OpenGL grafiky.

Qt umožňuje vyvíjet aplikace pro systémy Windows, OSX, GNU/Linux a ostatní Unix systémy s X Window systémem, Windows CE a Mobile, Symbian, Maemo a GNU/Linux bez X Window Systému. Aplikaci se stejným zdrojovým kódem je možné zkompilovat pro použití s X Window Systémem nebo bez něj, aniž bychom prováděli změny v kódu – pouhou změnou kompilátoru. Při používání multiplatformních knihoven je možné psát plně multiplatformní aplikace.

Je naprogramované v C++, ale obsahuje vlastní preprocesor, umožňující použití speciálních konstrukcí, které čisté C++ nezná. API bylo primárně navrženo jako objektové a díky tomu je jeho použití v objektovém kódu velice snadné. Existují i neoficiální wrappery pro použití v jazycích C#/.NET, Java, Python, Pascal, Perl, PHP, Ada a Ruby.

Společně s frameworkem je dostupné i kompletní vývojové prostředí, vytvořené speciálně pro Qt. Jeho hlavní částí je Qt Creator, editor, obsahující funkce pokročilého IDE – správu projektů, debugger, inteligentní doplňování kódu, kontextovou nápovědu a GUI designer. Ten umožňuje návrh uživatelského rozhraní stylem drag & drop. Především GUI designer je věc, kterou IDE pro ostatní knihovny neposkytují. Díky soustředění pouze na jeden programovací jazyk a jednu knihovnu je vše podřízeno jejich potřebám – jednoduché a intuitivní, což se o universálních Integrated Development Environment (IDE) často říci nedá.

Za vývojem frameworku stojí společnost Trolltech, kterou nedávno kopila Nokia. Ta ho používá ve svých projektech, což přináší záruku dlouhodobého a cíleného vývoje celé platformy. [QT]

Z obrázku 5 je patrné, že vzhled je podobný aplikacím založeným na platformě GTK+. Pomocí témat je opět možné vzhled jednotlivých widgetů změnit.

Obrázek 5: Ukázková aplikace s použitím knihovny Qt

Výhody :

- komplexní,
- jednoduché tvorba GUI i metodou drag & drop,
- možné použití s X Window systémem i bez něho,
- vhodná i pro velké projekty.

Nevýhody :

- větší náročnost na systémové prostředky.

Jako knihovna pro vývoj grafického klienta byla vybrána knihovna Qt pro běh bez X Window serveru. Hlavními důvody pro výběr byla jednoduchost vývoje a dostupnost simulačního prostředí framebufferu přímo v Qt IDE. Svoji roli sehrál i fakt, že v době výběru se počítalo s použitím kitu i.MX31, který ve svém systému obsahuje upravenou verzi Qt pro toto zařízení s proprietárními ovladači například pro dotykový displej. Z důvodů omezení kitu bylo vybráno jako základní rozlišení 640×480.

4 Komunikační rozhraní

Grafický klient sám o sobě žádná data neměří – nějakým způsobem je musí získávat z programu, který má na starosti měření dat a základní výpočty veličin. Tento program může běžet na stejném fyzickém zařízení nebo se k němu bude připojovat vzdáleně. Jedním z bodů zadání bakalářské práce je proto navrhnout komunikační rozhraní mezi grafickým klientem a virtuálním měřícím přístrojem.

Výsledkem práce Viktora Bubly [Bubla2009] je **Virtuální měřící přístroj (VMP)**, který simuluje chování reálného PMD. Virtuální měřící přístroj vznikl kvůli zjednodušení vývoje softwaru pro reálný PMD. Cílem této bakalářské práce je navrhnout grafické rozhraní, které bude komunikovat s tímto modulem. Virtuální přístroj umí pouze ukládat XML soubory z daného umístění a zasílat je pomocí TCP/IP. Přidání dalších rozhraní, vzhledem k modulárnosti systému, není problém.

Bohužel se VMP nepovedlo jednoduše rozběhnout, protože má části kódu společné s firmwarem reálných PMD a sdílený kód se od doby posledních úprav VMP změnil do té míry, že je nekompatibilní. Jeho úprava by byla časově náročná, proto se rozhodlo, že se bude komunikovat s reálným měřícím přístrojem, konkrétně SMV44 firmy KMB [KMB].

Power Monitoring Device (PMD) je elektronické zařízení, sloužící k monitorování kvality elektrické energie. Bývá vestavěné v rozvodných skříních na přívodech do celých provozů nebo konkrétních strojů. Zařízení provádí výpočty elektrických veličin, zobrazuje je na displeji, archivuje a umožňuje vzdálené připojení pro online monitorování nebo stažení naměřených dat. [Bubla2009]

4.1 XML soubory

Základním způsobem komunikace je ukládaní a čtení Extensible Markup Language (XML) souborů. Tento způsob je nejjednodušší, protože to umí už samotná knihovna *libxmlrw*. Jeho použití je omezené pouze na případ, kdy VMP a grafický klient běží na stejném přístroji a jsou nakonfigurovány tak, aby ukládaly a načítaly soubory ze stejných umístění.

Struktura souborů je stejná, jakou používá software společnosti KMB pro Windows. Ať už je použita jakákoliv metoda komunikace, tyto XML soubory se používají pro ukládání a nahrávání konfigurace PMD.

4.2 Vzdálená komunikace

Největší nevýhodou prvního řešení je nemožnost vzdálené komunikace. Metody, popsané dále, umožňují jak komunikaci v rámci jednoho fyzického zařízení, tak i dvou oddělených zařízení.

Daní za vzdálenou komunikaci je nutnost běhu nějakého serveru na straně PMD a klienta na straně grafické aplikace. Výkonové nároky a složitost implementace serveru i klienta závisí na použitém způsobu komunikace.

FTP

První, spíše teoretickou, možností komunikace je použití File Transfer Protocol (FTP), což je protokol běžně používaný pro posílání souborů po TCP/IP.

Na straně serveru se nemusí nic měnit, jen se jako služba spustí FTP server, který bude přes TCP/IP nabízet ke stažení XML soubory. Server se nastaví tak, aby umožňoval přístup ke složce, kam si VMP ukládá XML soubory, jinak se v samotné aplikaci nic měnit nemusí.

FTP serverů je pro GNU/Linux dostatečný výběr, mezi nejznámější patří *ProFTPD*, *vsftpd*, *Bftpd* nebo *Pure-FTPd*. Pro naše potřeby postačí co nejjednodušší s malou instalací a režií. Jedinou funkcí, o které by se dalo uvažovat, je šifrovaný přenos dat.

V grafické aplikaci je možné také použít na pozadí některý z konzolových FTP klientů, kterých je pro GNU/Linux opět na výběr několik. Pro toto použití je dostatečný *wget*, který je dostupný snad v každé distribuci, a umí i zmíněné šifrování. Nebo je možné FTP klienta doprogramovat přímo do grafické aplikace. Protokol není složitý, proto za použití vhodné knihovny by jeho implementace nebyla problém.

Výhody :

- jednoduchá implementace na serveru i klientovi.

Nevýhody :

- nemožnost přímé komunikace – pouze soubory,
- použití dalších programů nebo jejich volání z aplikace,
- pouze pro TCP/IP.

Web server

Další možností komunikace je použití jednoduchého Hypertext Transfer Protocol (HTTP) serveru a odpovídajícího klienta.

Qt obsahuje třídy pro základní síťovou komunikaci a při použití *QTcpSocket* je vytvoření HTTP klienta jednoduché. Implementace jednoduchého HTTP serveru na straně PMD by za použití vhodné knihovny také nebyl velký problém.

Jako komunikační prostředek by sloužily opět XML soubory, které by se zapisovaly a četly pomocí knihovny *libxmlrw*. Na rozdíl od FTP je pomocí HTTP možné posílat i přímo data metodami *GET* a *POST*, které jsou definované ve specifikaci HTTP.

Výhody :

- jednoduchá implementace na serveru i klientovi.

Nevýhody :

- pouze pro TCP/IP.

Protokol Modbus

Na PMD firmy KMB je již implementovaná podpora pro komunikaci pomocí protokolu Modbus a je nutné naprogramovat pouze klientskou část.

Modbus je otevřený komunikační protokol, který umožňuje komunikaci typu klient – server pro různá zařízení především v průmyslu. Protokol komunikuje až v aplikační vrstvě, proto může být implementován na různých typech sběrnic nebo sítí. Modbus byl vytvořen v roce 1979 firmou MODICON a od té doby se stal průmyslovým standardem.

Komunikace typu požadavek/odpověď probíhá pomocí funkčních kódů, které jsou popsány ve specifikaci. Část kódů je přesně definovaná a měla by být implementovaná ve všech zařízeních, komunikujících tímto protokolem. Část kódů je volná pro uživatelsky definované funkce. [Bedrník2009]

Nejběžnější implementace komunikují po sériové lince RS-232 nebo RS-485 a pomocí protokolu TCP/IP po ethernetu. PMD firmy KMB podle konkrétní konfigurace mají konektor RS-232, RS-485, ethernet nebo USB, po kterém se ovšem emuluje sériová komunikace jako při použití RS-232. Jedním protokolem je tedy možné komunikovat po všech dostupných rozhraních.

Výhody :

- již implementovaný ve stávajících zařízeních.

Nevýhody :

- PMD nepodporují přístup ke všem naměřeným hodnotám a konfiguračním proměnným,
- zařízení zatím nepodporuje komunikaci po sériové lince.

Ostatní průmyslové komunikační protokoly současná generace přístrojů nepodporuje, proto se jimy ani tato práce nebude dále zabývat.

Proprietární protokol KMB

Firma KMB vytvořila pro komunikaci mezi svými PMD a obslužným softwarem pro windows vlastní protokol pojmenovaný KMB Long. Komunikace je možná po ethernetu, nebo sériové lince. Jeho největší nevýhodou pro použití v tomto projektu je dosavadní neexistence knihovny pro GNU/Linux.

Výhody :

- již implementovaný ve stávajících zařízeních,

– komunikace po ethernetu i sériové lince.

Nevýhody :

– neexistence knihovny pro GNU/Linux.

5 Realizace

Jak již bylo napsáno v teoretické části, pro realizaci byl vybrán programovací jazyk C++ a grafická knihovna Qt pro embedded zařízení bez X Window systému. Aplikace byla testovaná ve virtuálním framebufferu s nastaveným rozlišením 640×480, který je součástí Qt IDE. Knihovna pro toto testování musela být speciálně zkompilovaná, což je popsané v kapitlole 5.6. S těmito prostředky byl naprogramován grafický klient pro zobrazování aktuálních dat a nastavování parametrů PMD. Jako druhá aplikace byl vytvořen textový klient pro prohlížení aktuálních dat a jednoduché vzdálené zobrazení a editaci konfiguračních struktur v PMD.

5.1 Překlad aplikace

Jelikož je aplikace určena pro možné budoucí nasazení na reálných přístrojích určených pro trh, byla jako primární jazyk v GUI i zdrojových kódech zvolena angličtina. Rozhraní programu je v nastavení možné přepnout do češtiny.

Sama knihovna Qt poskytuje komfortní rozhraní pro překlad programů. Při vytváření uživatelského rozhraní v Qt Creatoru jsou všechny texty ukládány do funkcí, které umožňují jejich jednoduchý překlad. Všechny texty, které se píší do kódu ručně a mají se zobrazovat, proto musíme uzavřít do funkcí *tr()*, *trUTF8()* nebo některé z dalších, složitějších konstrukcí. Pro každý jazyk je nutné do konfiguračního souboru projektu **.pro* vložit následující řádek pro identifikaci souboru s překladem.

Zdrojový kód 1: Ukázka přidání češtiny do konfiguračního souboru

```
TRANSLATIONS = calendar_cs_CZ.ts
```

Konzolový program *lupdate*, s projektovým souborem **.pro* jako parametr, vygeneruje a později aktualizuje sobory s překladem, které jsme přidali do konfigurace v **.pro*.

Pro samotný překlad se používá grafický program Qt Linquis, ve kterém se k originálním textů dopisují jejich přeložené alternativy.

Nakonec konzolový program *lupdate*, opět s **.pro* souborem jako parametrem, vytvoří z **.ts* binární sobor s překladem, který už se použije v aplikaci. [QTserial]

V kódu programu se potom o změnu jazyka stará třída *QTranslator* použitá v *main()* a nastavení se provádí v patřičné záložce konfigurace.

Dále v textu bude popisováno anglické rozhraní.

5.2 Ukládání nastavení

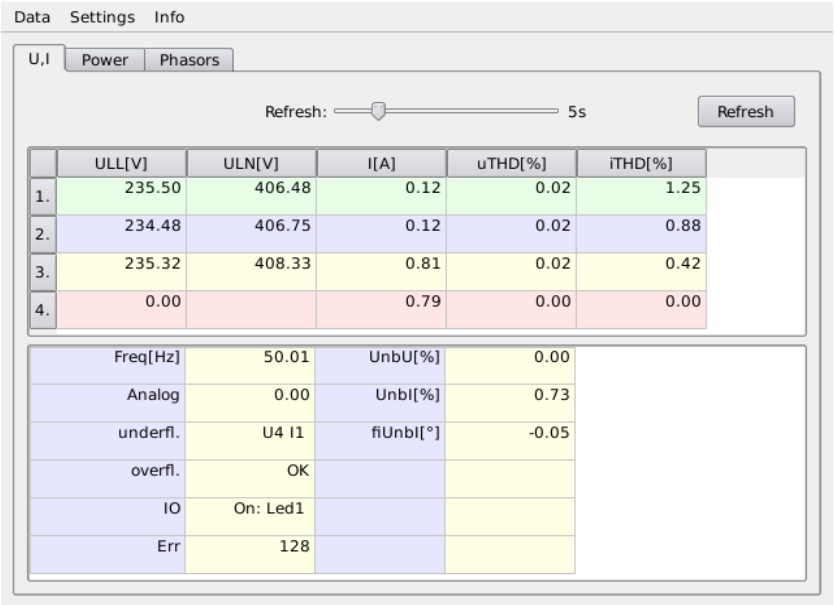
V operačním systému GNU/Linux se standardně konfigurační údaje ukládají do souborů zpravidla začínajících tečkou, například *.configure*. Soubory s nastavením stěžejních programů se ukládají do adresáře */etc/*. Konfigurace uživatelských programů se zpravidla ukládá v domovském adresáři do složek, pojmenovaných podle daného programu a začínajících tečkou, nebo hromadně v adresáři */.config*. Pro tuto aplikaci byl zvolen konfigurační soubor */.config/viewer.conf*.

Do konfiguračního souboru se ukládají pouze nastavení samotné aplikace – jazyk, metoda komunikace a nastavení jednotlivých druhů komunikace. Konfigurační soubory PMD se ukládají pomocí knihovny libxmlrw do XML souborů.

5.3 Grafický klient

Klient je vytvořen pro přehledné zobrazování průběžně měřených dat a úpravu konfiguračních struktur v PMD. Aby byla zachována konzistence ovládání, jsou jednotlivé obrazovky rozvrženy podobně jako v programu pro vzdálený přístup k PMD pro Windows od firmy KMB.

Grafický klient má tři základní části, mezi kterými se přepíná pomocí horního menu. Vždy je aktivní pouze jedna ze tří částí, po zavření se uvolní z paměti a přestanou komunikovat s PMD.



Obrázek 6: Zobrazení aktuálních dat

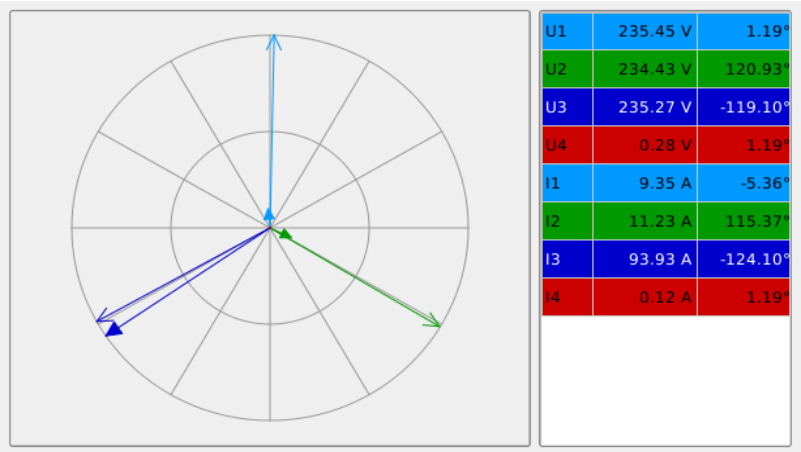
První část slouží k zobrazování aktuálních dat. Jak je vidět na obrázku 6 skládá se ze tří záložek. První zobrazuje proudy, napětí a jejich harmonické zkreslení, frekvenci a další měřené veličiny. Druhá zobrazuje výkony a třetí fázory.

V horní části je vidět posuvník, kterým se nastavuje doba, za jakou se načtou nová data. Programově to je řešeno vlastním widgetem, který obsahuje posuvník, dva popisky, tlačítko pro okamžité obnovení a časovač z frameworku Qt. Tento widget se potom vkládá do všech tří záložek, díky čemuž se mohou načítat nezávisle na sobě.

Nejnižší poloha je označená jako průběžné načítání. Pokud komunikace probíhá pomocí protokolu Modbus, časovač je nastaven na hodnotu 0, což znamená, že spustí přiřazenou akci jakmile jsou volné systémové prostředky. V praxi je tato jeho vlastnost velice užitečná, protože se data načítají nejvyšší možnou rychlostí, ale přitom GUI stále plynule reaguje na uživatelské akce.

Další polohy jsou již napevno nastavené časy načtení ve vteřinách – od jedné do sto dvaceti. Poslední poloha vypíná časovač a obnovení je možné pouze ručně pomocí tlačítka.

Na obrázku 7 je vidět část poslední záložky se zobrazenými fázory. O vykreslení diagramu se stará opět vlastní widget s upraveným *paintEvent*. Do jednoho diagramu se kreslí proudy i napětí. Největší proud a napětí jsou brány jako maximum a jejich délka je spočítána z rozměrů widgetu – mohou se tedy přizpůsobovat změně velikosti okna. Délky ostatních šipek jsou v poměru zmenšeny.



Obrázek 7: Zobrazení fázorů

Druhá volba menu obsahuje záložky s nastaveními – čtyři pro vzdálenou konfiguraci PMD a jednu pro nastavení samotné aplikace. Záložky se vzdáleným nastavením vypadají podobně – v horní části jsou standardní widgety, kterými se konfigurují volby a dole jsou

čtyři tlačítka. Pomocí *read* a *write* se načte a zapíše konfigurace do PMD pomocí zvolené metody komunikace. Po stisku tlačítek *load* a *save* se vyvolá dialog pro uložení nebo otevření souboru a je možné nahrát nebo uložit konkrétní XML soubor.

Na poslední záložce se nastavuje jazyk aplikace, použitá komunikační metoda a nastavení vybrané metody. U konfigurace protokolu Modbus je možnost ručně se připojit k vzdálenému přístroji, ale dojde k tomu i automaticky, při prvním pokusu o získání dat.

V poslední části se zobrazují pouze informace o programu.

V příloze B je uveden jednoduchý návod a screenshosty klienta.

5.3.1 Struktura programu

Program je tvořen s použitím objektového přístupu. Každá ze tří částí programu – Data, Settings a Info, je reprezentována jednou třídou. Jednotlivé záložky v sekci Data a Settings jsou také reprezentované vlastními třídami. Část záložek byla tvořena pomocí grafického nástroje pro tvorbu GUI a část byla tvořena ručně, aby se daly porovnat přednosti obou přístupů.

Při ruční tvorbě uživatelského rozhraní má programátor vše plně pod kontrolou, ale je to velice zdlouhavé a pracné, především při vytváření složitějších formulářů. Jediný případ, kdy to bylo jednodušší než použití grafického nástroje, je vytváření tabulek.

Grafický návrhář GUI je propracovaný a při jeho použití se neobjevily žádné problémy. Jediným drobným zádrhelem byla tvorba tabulek. Do každé buňky, která má později umožňovat měnit svůj obsah, se musí vložit nějaká hodnota, jinak pro ni návrhář nevytvoří proměnnou a nejde s ní vůbec pracovat.

Pro použití v jednotlivých záložkách bylo vytvořeno několik vlastních widgetů pro specifické účely.

Vlastní widget pro vykreslování fázorů je již popsán výše. Ovládání doby obnovení dat v sekci data je také řešené vlastním widgetem, který obsahuje časovač a další ovládací widgety.

Konfigurace jednotlivých způsobů komunikace je, pro možnost jejich jednoduché výměny při změně vybrané metody, řešena také vlastním widgetem pro každý způsob komunikace. Screenshoty nastavování komunikací jsou v příloze B.

5.4 Textový klient

Pro účely jednoduchého testování vznikla konzolová aplikace, která se snaží podobnou funkčnost jako grafický klient přenést do textového režimu. Má velkou část kódu společnou s

grafickým klientem, čímž je také demonstrováno oddělení prezentačních a funkčních částí kódu.

Klient obsahuje jednoduchý příkazový řádek, který má tři příkazy – *help*, *show* a *set*. Prvním příkazem se vypíše nápověda s dostupnými příkazy a jejich parametry. Stejného efektu se docílí i potvrzením prázdného řádku. Rozhraní programu je pouze v anglickém jazyce, ale vše je připraveno pro případnou lokalizaci do češtiny a dalších jazyků.

Příkaz *show* slouží k zobrazení aktuálních dat nebo konfigurační struktury. Jediný povinný parametr je název zobrazovaných dat nebo struktury. Podporováno je zobrazení aktuálních dat, které je pro přehlednost rozděleno na část s výkony, která je zde pojmenovaná *actdataPWR* a na zbytek, kde jsou především proudy a napětí, pojmenovaný *actdataUI*. Data se zobrazují pod sebou ve dvojici – název proměnné a její hodnota.

Konfigurační struktury se, na rozdíl od grafického klienta, nepřepočítávají do uživatelsky přívětivé podoby, ale zobrazují se přímo jako proměnné ze struktur. Pro běžné uživatele to je nepoužitelné, ale pro vývojáře to užitečné je, protože vidí přímo surová data. Výjimku tvoří ip adresy, které se převádí z *u32* do čitelné podoby.

Při zobrazení dat se na řádek nejdříve vypíše číslo, udávající pořadí proměnné, potom její název a nakonec hodnota. Číslo jednoznačně identifikuje každou proměnnou a usnadňuje jejich pozdější úpravu. Podporované struktury jsou *smpinstallconfig*, *smpconfig*, *smpelectricitymeterconfig* a *smparcconfig*.

Příkaz *set* slouží k přímému nastavování proměnných ze struktur. Podporovány jsou všechny struktury jako při čtení. Povinné parametry jsou tři, první je název struktury, druhý pořadí proměnné nebo její název a třetí je nová hodnota. Program nekontroluje správnost a smysluplnost zadaných hodnot, je proto nutné dbát zvýšené opatrnosti, protože je možné připojené PMD přenastavit takovým způsobem, že případné chyby nepůjdou jinými nástroji vrátit zpět.

Výběr použité komunikační metody a nastavení jednotlivých metod se načítá z konfiguračního souboru, který se musí editovat ručně v textovém editoru, jak to je pro konzolové programy běžné.

Pokud je zadán neplatný příkaz, nebo neplatný parametr, je zobrazena chybová hláška.

V ukázce 2 je vypsán příkaz help, delší ukázka práce s klientem je v příloze C.

Zdrojový kód 2: Ukázka z textového klienta

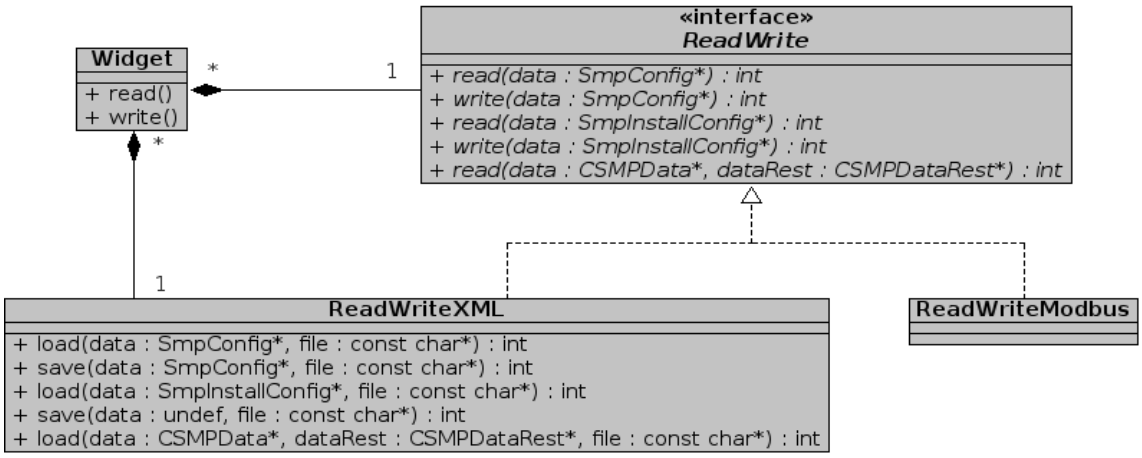
```
for help type 'help' and press return
> help
help                    Prints this help
show [data]             Display data
     actdataUI          U and I from ActData
     actdataPWR         Powers from ActData
     smpinstallconfig   SmpInstallConfig
     smpconfig          SmpConfig

set [structure] [position] [value]
        Set value to variable on position in structure
set [structure] [variable] [value]
        Set value to variable in structure
>
```

5.5 Komunikace

Nakonec bylo rozhodnuto, že klient bude umožňovat komunikaci pomocí XML souborů a protokolu Modbus. Každá záložka načítá data nezávisle na ostatních, ale protokol Modbus má jedno globální spojení pro celou aplikaci, přes které komunikují všechny záložky. Spojení se vytváří automaticky při prvním požadavku a po chybě se opět automaticky spouští, ale je možné ho ručně vytvořit nebo zrušit v nastavení aplikace.

Komunikace je naprogramovaná univerzálně pro textového i grafického klienta a díky své objektové struktuře umožňuje velice jednoduché přidávání dalších komunikačních metod bez zásahu do zbytku programu.

Protože PMD v každé z podporovaných metod komunikace nabízí data v jiném stádiu přepočítání je nutné hned po přijetí data přepočítat do podoby, ve které se budou zobrazovat v jednom z klientů.



Obrázek 8: UML diagram tříd ReadWrite

5.5.1 Abstraktní třída ReadWrite

Aby bylo možné jednoduše přidávat další způsoby komunikace byla vytvořena abstraktní třída *ReadWrite*, ve které se definuje rozhraní, které musí implementovat všechny třídy reprezentující konkrétní komunikační metody. S instancemi těchto konkrétních tříd se potom pracuje jako s instancí třídy *ReadWrite*, díky čemuž je kód nezávislý na použité komunikační metodě.

Ve třídě *ReadWrite* se jsou deklarované dvě přetížené virtuální metody – *read* a *write*. Obě mají jeden parametr – strukturu, do které ukládají nebo ze které čtou data. Podle typu struktury se rozhodne, jaká konkrétní metoda bude použita. Jak je vidět na ukázce 3 z deklarace virtuální třídy *ReadWrite*, je implementováno čtení a zápis tří konfiguračních struktur a čtení jedné datové struktury.

Zdrojový kód 3: Definice přetížených virtuálních funkcí read a write

```
virtual int read(SmpConfig *data)=0;
virtual int write(SmpConfig *data)=0;
virtual int read(SmpInstallConfig *data)=0;
virtual int write(SmpInstallConfig *data)=0;
virtual int read(SmpElectricityMeterConfig *data)=0;
virtual int write(SmpElectricityMeterConfig *data)=0;
virtual int read(CSMPDataCalc *pCSMPDataCalc)=0;
```

Metody mají jako návratovou hodnotu integer indikující chybu. Pokud metoda vrátí 0, proběhlo čtení nebo zápis v pořádku, jakákoliv jiná návratová hodnota indikuje chybový stav. Aplikace podle typu klienta a zpracovávané struktury na to aplikace patřičně zareaguje, ale nikdy to nezastaví běh klienta.

Z virtuální metody *ReadWrite* se potom odvozují třídy implementující konkrétní komunikační metody a přepočet dat do konečné podoby.

Třída ReadWriteFactory

O tom, který z potomků třídy *ReadWrite* se použije, se rozhoduje v metodě *chooseRWType* třídy *ReadWriteFactory*. Tato metoda, na základě nastavení uloženého v konfiguraci, určuje způsob komunikace a vrací ukazatel na instanci třídy *ReadWrite*, za kterým je skryta instance konkrétní třídy implementující žádanou komunikaci. V konstruktoru každé třídy, ve které budeme načítat nebo ukládat data, vytvoříme instanci třídy *ReadWriteFactory* a její metodou *chooseRWType* získáme instanci třídy *ReadWrite*, pomocí které čteme a zapisujeme data. Ve vlastní třídě pro zobrazování dat tedy není žádný kód závislý na vybrané komunikační metodě.

5.5.2 XML soubory

Jak už bylo řečeno v kapitole 4.1, nejjednodušším způsobem komunikace je načítání a ukládání souborů do sdíleného adresáře. Tento způsob komunikace implementuje třída *ReadWriteXML*.

Oproti třídě *ReadWrite* obsahuje navíc přetížené metody *load* a *save*, které mají stejně jako *read* a *write* jako parametr jednotlivé datové struktury, ale navíc mají ještě parametr typu *const char**, kterým je cesta k souboru. Tyto metody se využívají při ručním ukládání a nahrávání XML souborů z konkrétního umístění. Metody *read* a *write* je také vnitřně používají, ale jako umístění souboru používají nastavení načtené z konfigurace.

Základem fungování všech metod je volání knihovny *libxmlrw*, která vznikal během loňského projektu [Bedrník2009]. Pomocí standardní knihovny *libxml2* načítá a ukládá do souborů data z konfiguračních a datových struktur používaných v PMD. Během bakalářské práce probíhaly na této knihovně úpravy, spočívající především v přidávání a úpravě podporovaných souborů a datových typů, nebo v opravě drobných nedostatků. Vnitřní filozofie ani struktura nebo API se neměnilo.

Knihovna je z důvodů jejího použití i v jiných projektech naprogramovaná v C. Pro potřeby klientů se kvůli předejitím zbytečným problémům překládá jako C++.

Používá se pomocí jednoduchého API. Každý XML soubor, který umí načítat a ukládat, má v knihovně funkci, pojmenovanou podle typu souboru, který načítá. Jako parametry potom dostává adresu souboru jako *const char** a ukazatel na strukturu. Například 4 je ukázaná jedna z přetížených metod *load* třídy *ReadWriteXML* načítá ze souboru *file* data do struktury SmpInstallConfig.

Zdrojový kód 4: Ukázka použití funkce z knihovny libxmlrw ve třídě ReadWriteXML

```
int ReadWriteXML::load(SmpInstallConfig *pSmpInstallConfig,
 const char *file)
{
    if(smpInstallConfigReader(file,pSmpInstallConfig))
    {
        return -1;
    }
    return 0;
}
```

V současné době umí knihovna načítat 9 a zapisovat 8 druhů souborů. Do budoucna se do stávající knihovny může přidávat podpora pro další soubory, ale kód se bude stávat nepřehledným. Možným řešením je přepsat knihovnu od začátku. Pokud by se využívala i jinde, tak nad stávající knihovnou libxml2. Pokud by ovšem byla používána jen v tomto grafickém klientovi, bylo by ideální využít podporu pro XML přímo v knihovně Qt.

5.5.3 Protokol Modbus

Pro komunikaci pomocí protokolu Modbus slouží třída ReadWriteModbus. Před vlastním přenosem dat je nutné nejdříve zabezpečit spojení – to se děje mimo tuto třídu a ta díky tomu může být univerzálně použitelná pro TCP i sériovou komunikaci. Protokol Modbus chápe pojem Master a Slave obráceně než je běžné, proto se grafická aplikace označuje jako Master a PMD, se kterým se komunikuje a ze kterého se stahují data, je Slave.

Při požadavku na stažení dat se nejprve zjistí, jestli je aplikace spojená s nějakým Slave

zařízením. Pokud není, automaticky se pokusí připojit s údaji uloženými v konfiguraci. V případě chyby spojení nebo získávání dat se tabulka podbarví červeně.

Pro spojení slouží funkce *connectMODBUS* a analogicky *disconnectMODBUS*, které v globální proměnné *paramModb* uchovávají údaje o aktuálním spojení, které potom může být využíváno z celé aplikace. Spojení se může přerušit ručně v nastavení nebo automaticky v destruktoru aplikace.

Pro komunikaci je použita knihovna *libmodbus* [Libmodbus], která umožňuje oba typy spojení. Obsahuje dvě funkce pro inicializaci spojení – jednu pro TCP a druhou pro sériovou linku. Rozdíl mezi nimi je jednak v použitém komunikačním médiu a jednak ve vysílacím režimu. Po TCP se komunikuje v Modbus ASCII, ale mohl by i v Modbus RTU. Po sériové lince se komunikuje pouze pomocí Modbus RTU.

V Modbus RTU je každý byte zprávy složen ze dvou čtyř bitových hexadecimálních znaků, mezera mezi znaky nesmí být delší než jeden a půl znaku. Začátek a konec zprávy je definován jako odmlka delší než I znaku.

V Modbus ASCII je každý byte posílán jako dva ASCII znaky. Oproti RTU je pomalejší, ale umožňuje delší mezery mezi vysíláním, až 1 sekunda.

Komunikace po sériové lince je implementována, ale nemohla být vyzkoušena, protože ji zatím PMD nepodporuje. Zařízení, které bylo k dispozici pro testování nemá port RS-232, ale Universal Serial Bus (USB), po kterém ale probíhá komunikace simulující komunikaci po RS-232. Pro připojení takovéhoto zařízení ke GNU/Linuxu je nutné zavést modul *usbserial* s patřičnými vendor a product id. Zavedení modulu předvedeno v ukázce 5. Po zavedení modulu se vytvoří zařízení */dev/ttyUSB0*, které se zadá do nastavení Modbus komunikace.

Zdrojový kód 5: Zavedení modulu

```
modprobe usbserial vendor=0x0403 product=0xacb2
```

5.6 Kompilace

Protože zkompilování obou klientů není úplně triviální, je mu věnovaná krátká podkapitola.

Knihovny

Mezi zdrojovými kódy obou klientů jsou i dvě knihovny – *libxmlrw* a *libmodbus*. Před začátkem kompilování klientů je nutné zkompilovat a nainstalovat obě knihovny. *Libmodbus*

je nijak neupravovaná distribuce [Libmodbus], která se kompiluje a instaluje standardní posloupností příkazů *./configure*, *make* a *make install*, v případě nejasností obsahuje distribuce návod.

Knihovna libmodbus obsahuje vlastní makefile, který výsledný archiv *libxmlrw.a* vytvoří v pracovním adresáři grafického klienta, je tedy nutné dodržet adresářovou strukturu. Zdrojové kódy jsou uloženy jako projekt vývojového prostředí NetBeans, v případě potřeby je v něm možné upravit parametry kompilátoru a archivátoru – jsou použity *g++* a *ar*.

Klienti

Oba klienti jsou uloženi jako samostatné Qt projekty. Vývoj probíhal v Qt Creatoru, ale zkompilovat jdou i pouze s příkazy *qmake -r* a *make -w*. Důležitý je ale výběr správného qmake. Jak již bylo řečeno, vývoj probíhal ve virtuálním framebufferu, čemuž byla přizpůsobena i celá knihovna, speciálně zkompilovaná pro toto použití. Při použití standardní vývojové knihovny Qt jde aplikace zkompilovat také, ale spustí se maximalizovaná a mohou se objevit další drobné nedostatky.

Konzolového klienta zkompilujeme obdobně jako grafického, jen je opět nutné dodržet uspořádání adresářů, protože používá mnoho kódů z adresáře grafického klienta.

5.6.1 Knihovna Qt

Pro používání virtuálního framebufferu je nutné zkompilovat Qt knihovnu s patřičnými parametry. Zdrojové kódy získáme z internetových stránek knihovny [QT]. Nejprve spustíme konfigurační utilitu *./configure*. Minimální parametry pro *make* jsou následující.

- -qt-gfx-qvfb, grafický ovladač QVFb pro virtual framebuffer,
- -qt-kbd-qvfb, vstup klávesnice z QVFb,
- -qt-mouse-qvfb, vstup myši z QVFb,
- -prefix /usr/local/Trolltech/Qt-qvfb-version, odlišení instalace QVFb knihovny od ostatních.

[LernQt]

Příkazem *make* s těmito parametry knihovnu zkompilujeme a *make install* nainstalujeme do umístění určeného prefixem. Pro správu více verzí je vhodnější použít Qt Creator, který pro to má komfortní uživatelské rozhraní.

6 Závěr

Výsledkem práce je grafický klient s parametry požadovanými v zadání – je schopný běhu na embedded platformě, využívá knihovnu Qt, zvládá komunikaci několika metodami a umí zobrazovat data a měnit nastavení PMD. Navíc vznikl, pro účely budoucího testování, ještě konzolový klient, který umožňuje zobrazení měřených dat v textové podobě a především přímé čtení a zápis konfiguračních proměnným na připojeném PMD. Screenshoty a ukázka použití klientů jsou předvedeny v přílohách B a C.

V zadání bylo požadováno testování komunikace s virtuálním měřícím přístrojem, ten se během práce ukázal jako zastaralý a jeho opětovné uvedení do provozu se současnými změnami ve sdíleném kódu by bylo časově náročné, proto se komunikace testovala přímo na reálném PMD. Pro ověření kompatibility je to dokonce lepší než virtuální komunikace.

Klienti umí komunikovat s PMD přímou výměnou XML souborů nebo pomocí protokolu Modbus po TCP. Komunikace po sériové lince je ze strany klientů implementována, ale PMD ji zatím nepodporuje. Díky použitému objektovému způsobu programování je budoucí rozšiřování klientů o podporované komunikační metody nebo zobrazovaná data velice jednoduché.

Řešení obou klientů je postavené na knihovně Qt a snaží se využít všech jejích vlastností, které usnadňují programování. Nevýhodou použití takto komplexní knihovny je poměrně velký výsledný program. Při použití jednodušší knihovny by byl výsledný program menší a méně náročný na systémové prostředky, ale vývoj aplikace by byl složitější a časově náročnější. Protože se předpokládá budoucí rozšiřování aplikace, budou se rozdíly v případném použití menší knihovny zmenšovat a naopak budou dále růst výhody tohoto komplexního řešení.

Přínos této práce je především v nalezení vhodného způsobu tvorby grafického klienta, jeho komunikace s PMD a předvedení tohoto způsobu v praxi. Vytvoření klienti představují stabilní základ pro budoucí rozšiřování jejich funkcí. Vše bylo vyvíjeno především s požadavkem na budoucí jednoduchou rozšiřitelnost.

Dalším pokračováním práce bude vytvoření kompletního klienta, kterým bude možné plně ovládat PMD a zobrazovat všechna dostupná data. Díky použití komplexní knihovny je možné do klienta přesunout i některé funkce, které jsou pro současné PMD dostupné pouze z obslužné aplikace pro Windows.

Seznam použité literatury

[BBoard] *Beagleboard.org* [online]. 2008, 2010 [cit. 2010-05-18]. Dostupné z WWW: <http://beagleboard.org/>.

[Bedrník2009] BEDRNÍK, Tomáš. *Využití embeded linuxu v průmyslových řídicích aplikacích*. Liberec, 2009. 11 s. Ročníkový projekt. Technická univerzita v Liberci, Fakutla mecahtroniky, informatiky a mezioborových studií.

[Bubla2009] BUBLA, Viktor. *Virtuální měřící a monitorovací přístroj v prostředí GNU/-Linux*. Liberec, 2009. 47 s. Bakalářská práce. Technická univerzita v Liberci, Fakutla mecahtroniky, informatiky a mezioborových studií.

[Freescale] *Freescale* [online]. c2004 [cit. 2010-05-21]. I.MX31 Product Development Kit. Dostupné z WWW: <http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX31PDK>.

[GNU] *GNU Operating System* [online]. c1996, Mar 9 2010 [cit. 2010-05-17]. The Free Software Definition. Dostupné z WWW: <http://www.gnu.org/philosophy/free-sw.html>.

[GTK] *The GTK+ Project* [online]. c2007, April 1, 2010 [cit. 2010-04-19]. Dostupné z WWW: <http://www.gtk.org/>.

[KMB] *KMB systems* [online]. 2010 [cit. 2010-05-10]. Dostupné z WWW: <http://www.kmb.cz/>.

[LernQt] *Lern Qt : Teaching Qt® to the world!* [online]. 2009-06 [cit. 2010-05-16]. An Embedded Linux Development Setup. Dostupné z WWW: <http://thelins.se/learnqt/2009/06/an-embedded-linux-development-setup/>.

[Libmodbus] *A groovy libmodbus!* [online]. 2007, 2010-03-24 [cit. 2010-05-15]. Dostupné z WWW: <https://launchpad.net/libmodbus>.

[LynuxWorks] *LynuxWorks* [online]. 2002 [cit. 2010-05-21]. BSQUARE Intel XScale PXA255 DevkitIDP. Dostupné z WWW: <http://www.lynuxworks.com/board-support/bsquare/intel-pxa255-devkitidp.php>.

[OpenMotif] *The Open Group : Making Standards Work* [online]. 19-Mar-2003 [cit. 2010-05-10]. Open Motif Data Sheet. Dostupné z WWW: <http://www.opengroup.org/openmotif/datasheet.html>.

[OSI] ARNOŠT, Pavel. *Open Source Initiative* [online]. 2004 [cit. 2010-05-18]. The Open Source Definition. Dostupné z WWW: <http://www.opensource.org/docs/osd>.

[QT] *Qt : Cross-platform application and UI framework* [online]. c2008, Apr 09, 2010 [cit. 2010-04-19]. Dostupné z WWW: <http://qt.nokia.com/>.

[QTserial] WATZKE, David; VANĚK, Petr. *ABC Linuxu* [online]. 3.3.2009, 26.1.2010 [cit. 2010-05-21]. Seriál: Qt 4 - psaní grafických programů. Dostupné z WWW: <http://www.abclinuxu.cz/serialy/qt-4-psani-grafickych-programu>.

[ROOT] *ROOT* [online]. 22. 8. 2001 [cit. 2010-05-18]. Co je to "Open Source software". Dostupné z WWW: <http://www.root.cz/clanky/co-je-to-open-source-software/>.

Příloha A - Licence

Licence jsou ve světě GNU/Linuxu poněkud komplikovaný problém. Hlavní myšlenky, na kterých stojí komunita kolem GNU/Linux, jsou Open source a Free software (překládaný jako Svoboný software). Bývají často zaměňovány, ale neznamenají přesně to samé.

Za svobodným softwarem stojí Free software foundation a dává svému uživateli čtyři základní práva:

- právo používat program za jakýmkoliv účelem,
- právo studovat, jak program funguje a upravovat ho pro svoje potřeby,
- právo šířit jeho kopie,
- právo šířit vaše upravené kopie.

[GNU]

Za Open source stojí Open Source Initiative, která definuje Open source v deseti bodech, která jsou zde ve zjednodušené podobě uvedeny:

- licence nesmí omezovat prodej nebo jinou distribuci programu ani jako součást balíku programů,
- produkt musí obsahovat zdrojový kód a musí umožňovat jeho distribuci,
- licence musí umožňovat vytváření odvozených prací a jejich šíření pod stejnou licencí,
- licence může omezovat distribuci zdrojového kódu pouze, pokud umožnuje distribuci tzv. *Patch files* spolu se zdrojovým kódem, musí povolit šíření programu zkompilovaného ze změněných zdrojových kódů,
- licence nesmí diskriminovat osoby nebo skupiny osob,
- licence nesmí zakazovat použití softwaru v určité oblasti, nesmí například zakázat používat sofware v komerčním prostředí,
- práva přiložená k programu musí platit pro všechny, kterým je program distribuován,
- práva přiložená k programu nesmí záviset na existenci programu v určitém sofwarovém balíku,

- licence nesmí ovlivňovat ostatní programy,
- licence musí být nezávislá na použité technologii.

[ROOT] [OSI]
Freee Software Foundation (FSF) udržuje seznam Free sofware licencí, který ještě dále rozděluje na kompatibilní a nekompatiblní s GNU General Public License (GPL). Mezi kompatibilní se řadí všechny tři verze GNU GPL, všechny verze GNU Lesser General Public License (LGPL), Apache License 2.0, Modified BSD, FreeBSD, X11, XFree86 license a mnoho dalších. Mezi nekompatibilní řadí Apache license 1.0 a 1.1, Original BSD license, LaTeX Project Public License, Jabber Open Source License, Eclipse Public license, Mozilla Public License a další.

Open Source Iniciative (OSI) schvaluje Open source licence, mezi které řadí všechny verze GNU GPL a LGPL, Apache license 2.0, Eclipse Public license, Mozilla Public License nové a zjednodušené BSD licence a další.

Z výše uvedeného je zřejmé, že existuje opravdu velké množství licencí a není smyslem této práce je všechny představit, proto jsou dále uvedeny pouze dvě nejpoužívanější.

GNU General Public License

Nejznámější open source a free software licencí je bezesporu GNU GPL, která je použita v mnoha Linuxových programech, především na samotném jádře systému. Licence existuje ve třech verzích, nejrozšířenější je druhá. Třetí je poměrně nová a teprve se začíná rozšiřovat mezi tvůrce programů.

GNU GPL je poměrně složitá, ale ve zkratce by se dalo říci, že zajišťuje všechny čtyři práva, která definují svobodný software. Pro vývojáře je asi nejdůležitější fakt, že když použijí ve svém programu část GNU GPL musí celý program licencovat jako GNU GPL a distribuovat včetně zdrojových kódů. To platí i pro použití knihoven, proto se pro knihovny příliš nepoužívá.

Jelikož je jádro systému distribuováno pod touto licencí, platí pro něj toto pravidlo také. Každá upravená verze Linuxu musí být šířena pod GNU GPL a dostupná včetně zdrojových kódů. Na to v minulosti doplatili někteří výrobci přehrávačů a dalších embedded zařízení založených na GNU/Linux. Tlakem ze strany komunity nebo dokonce soudy byli nuceni zveřejnit zdrojové kódy svých aplikací. V praxi se objevily způsoby, jak tato omezení obcházet, proto vznikla třetí verze licence, která je ještě složitější. Pro přelicencování jádra

pod třetí verzi GNU GPL by byl nutný souhlas všech autorů, kteří se podíleli na jeho vývoji, což je nereálné, proto pravděpodobně bude už navždy dostupné pod druhou verzí licence.

GNU Lesser General Public License

Původní GNU GPL omezovala použití v knihovnách, proto vznikla GNU LGPL, která byla vytvořena právě pro knihovny. Pokud je knihovna, licencovaná pod GNU LGPL, použita v programu, tento program může být distribuován pod jakoukoliv licencí. Omezení se týkají pouze případných úprav samotné knihovny, které musí být zveřejněny pod GNU LGPL a včetně zdrojových kódů. Tato licence se stala nepoužívanější licencí pro knihovny a je použitelná i pro knihovny v této práci.

Jako licence pro knihovny použité v tomto projektu je akceptovatelná jakákoliv licence, která umožní bez poplatků distribuovat výslednou aplikaci pod komerční licencí bez nutnosti zveřejňovat zdrojové kódy.

Knihovna Qt je distribuována pod více licencemi, takže je možné si vybrat tu, která bude pro konkrétní použití vyhovovat. Dostupná je pod GNU GPL 3.0, GNU LGPL 2.1 nebo pod uzavřenou, placenou licencí. Pro použití v tomto projektu vyhovuje licence GNU LGPL.

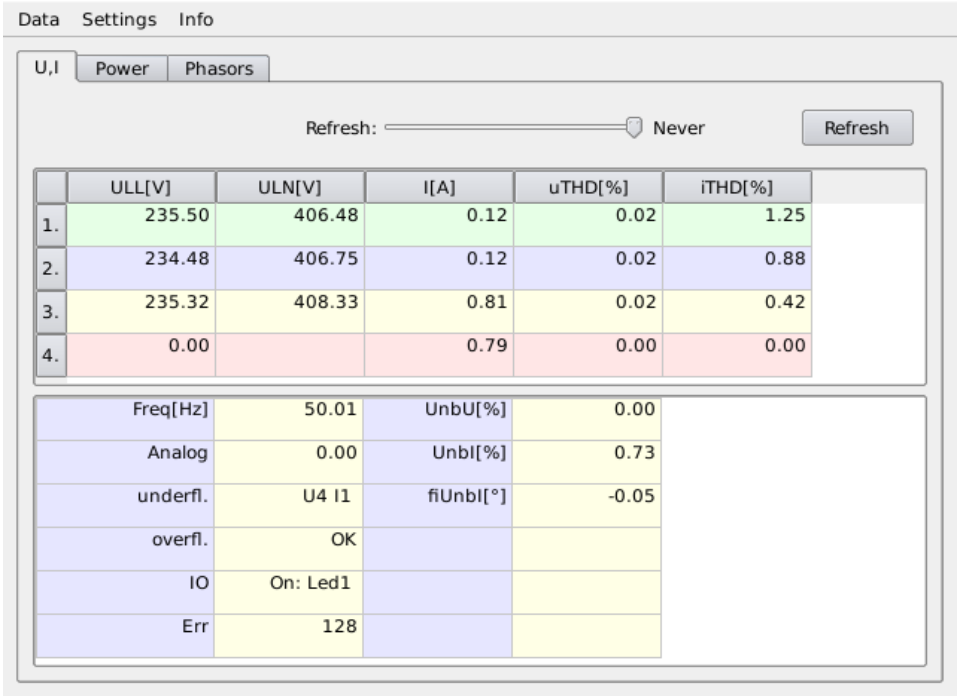
Knihovna *libmodbus* je dostupná také pod licencí GNU LGPL.

Příloha B - Návod a screenshoty grafického klienta

Zde je uveden jednoduchý návod pro obsluhu grafického klienta s odpovídajícími screenshoty.

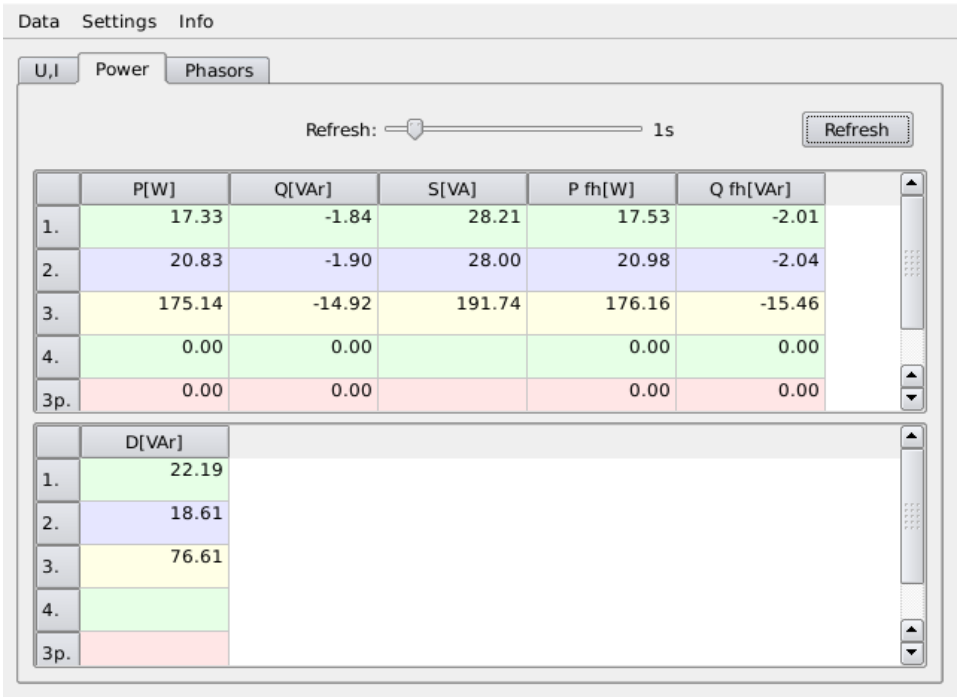
Data

Po startu se klient automaticky připojí pomocí vybraného protokolu a zobrazí se obrazovka se základními měřenými veličinami, jak je vidět na obrázku 1. V horní části je vidět posuvník, který ovládá rychlost načítání dat, je tam také tlačítko pro manuální načtení. Na dalších screenshotech jsou vidět jiná nastavení načítání.



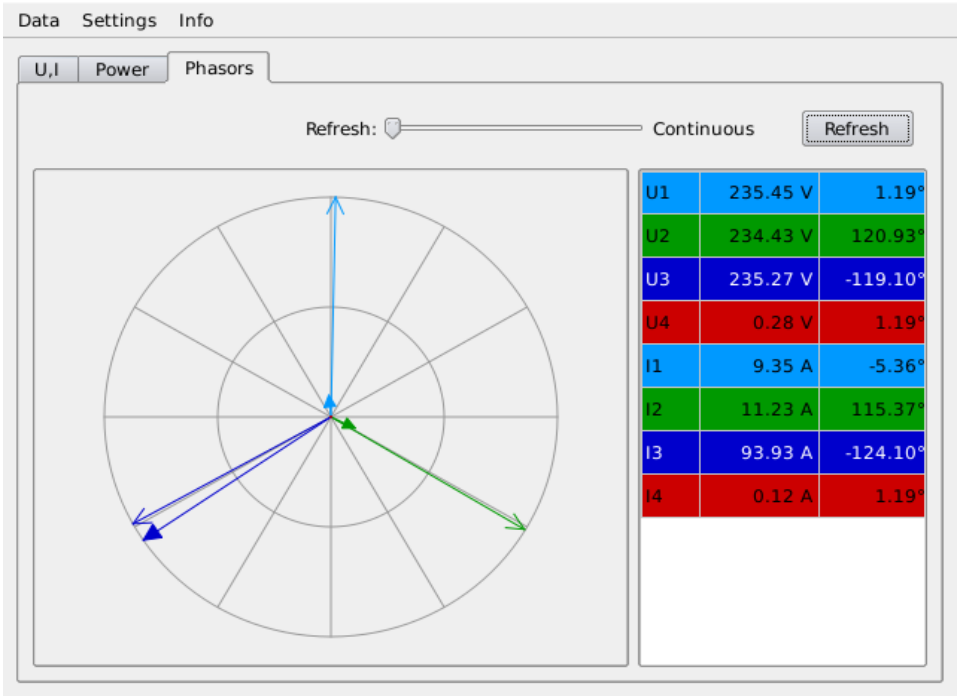
Obrázek 1: Obrazovka se základními daty

Na obrázku číslo 2 je druhá datová záložka, která obsahuje všechny výkony a výkony prvních harmonických.



Obrázek 2: Obrazovka s výkony

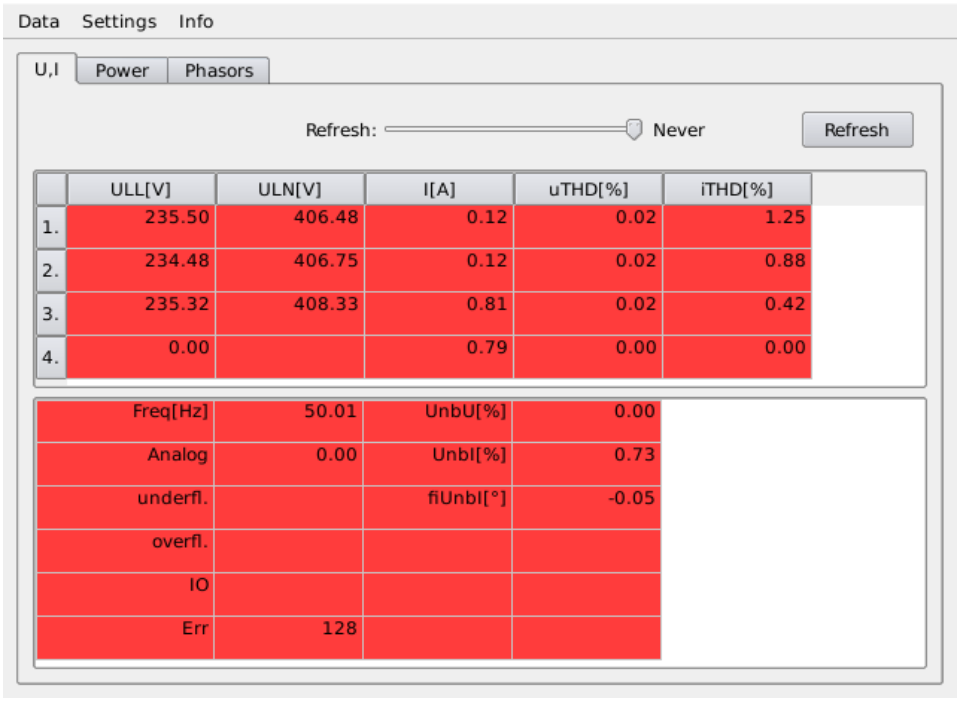
Na poslední záložce se zobrazují fázory – obrázek číslo 3



Obrázek 3: Obrazovka s fázory

Při chybě během čtení dat se všechny pole podbarví červeně, jak to je ukázáno na ob-

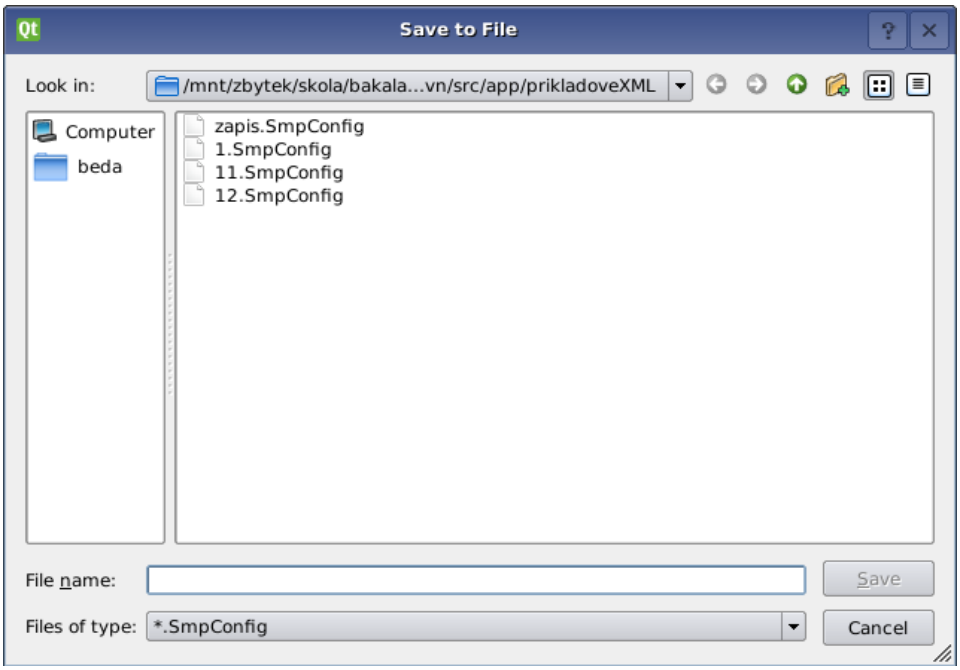
rázku 4.



Obrázek 4: Chyba čtení

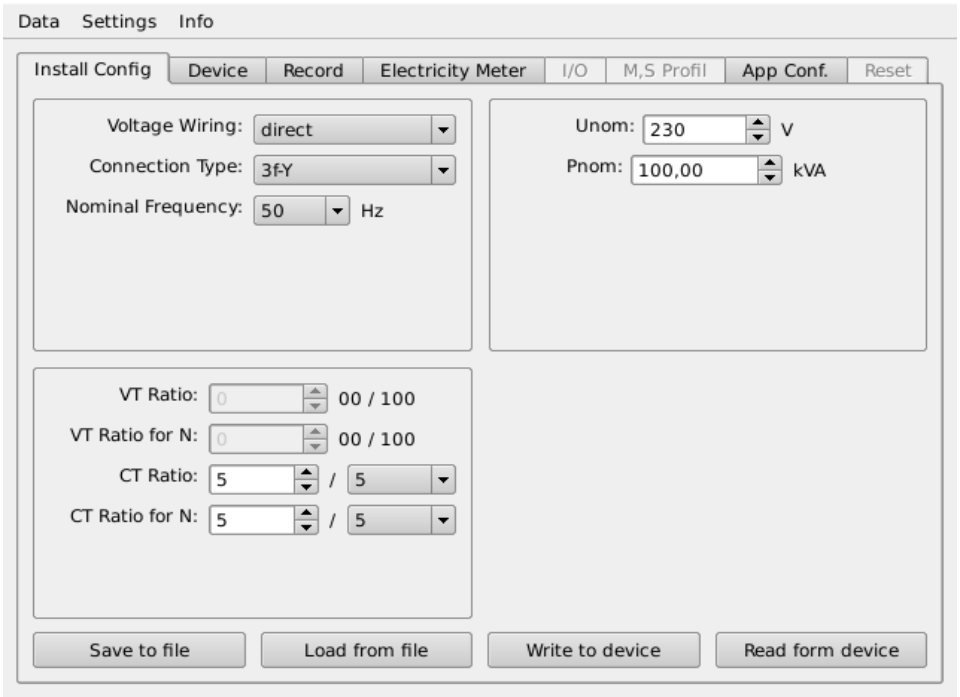
Konfigurace

Jednotlivé obrazovky s nastaveními mají ve spodní části čtyři tlačítka. První dvě, *Save* a *Load*, otevírají standardní dialog pro uložení nebo otevření souboru a vybranou konfiguraci uloží nebo nahrají z vybraného souboru, jak je vidět na obrázku 5. Druhá dvě tlačítka načítají a ukládají nastavení z přístroje pomocí vybrané komunikační metody.



Obrázek 5: Save dialog

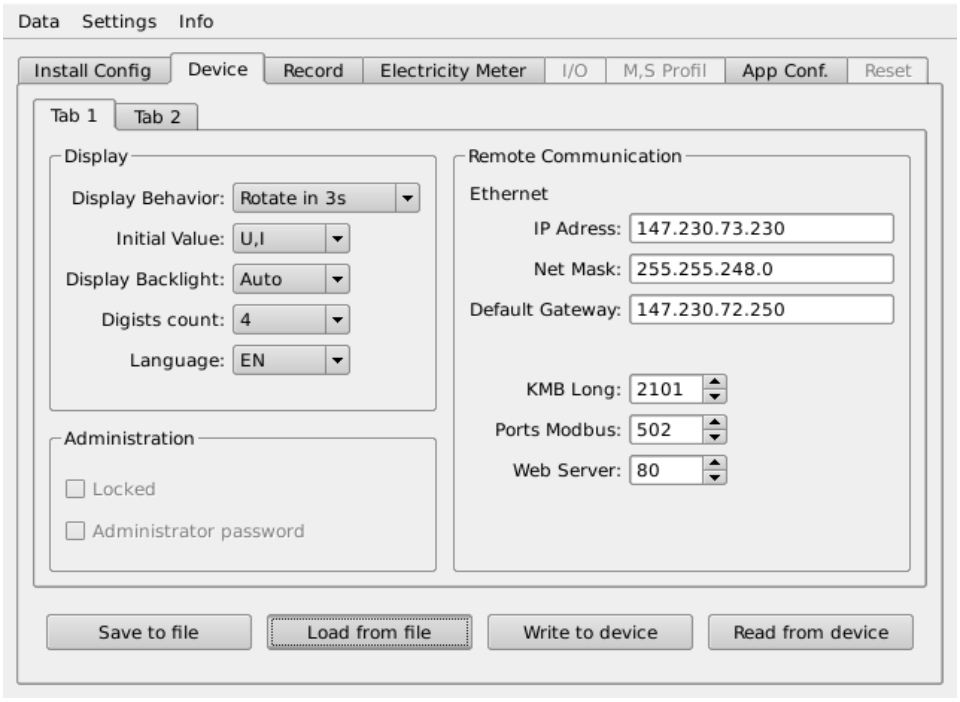
První záložka na obrázku 6 nazvaná Install Config zobrazuje instalační nastavení, které se využívá při přepočtu dat.



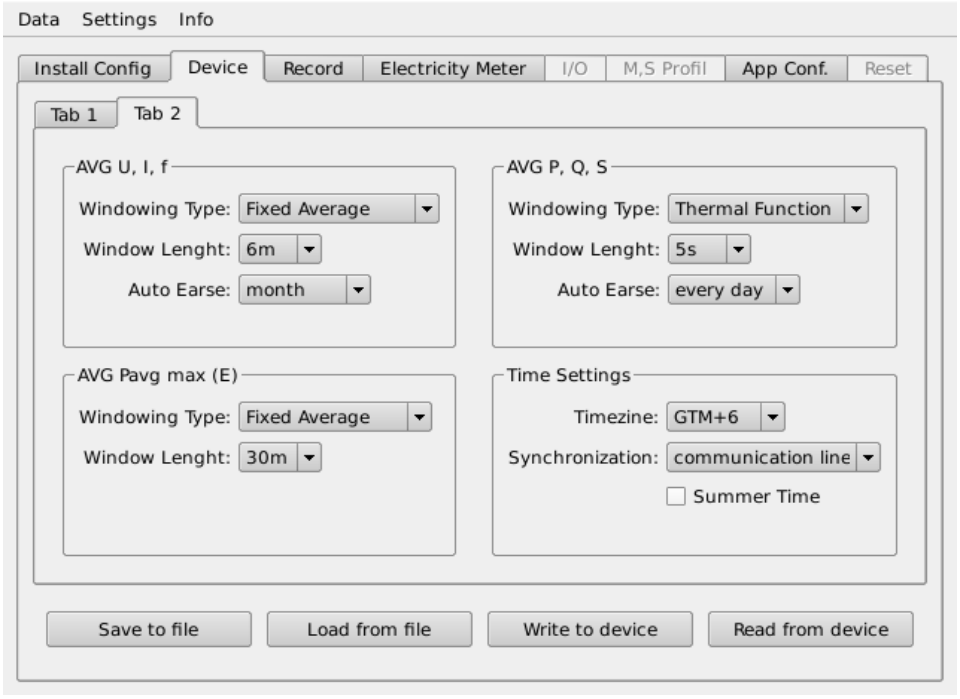
Obrázek 6: Install Config

Druhá záložka obsahuje nastavení zařízení a skládá se ze dvou podzáložek zobrazených

na obrázcích 7 a 8

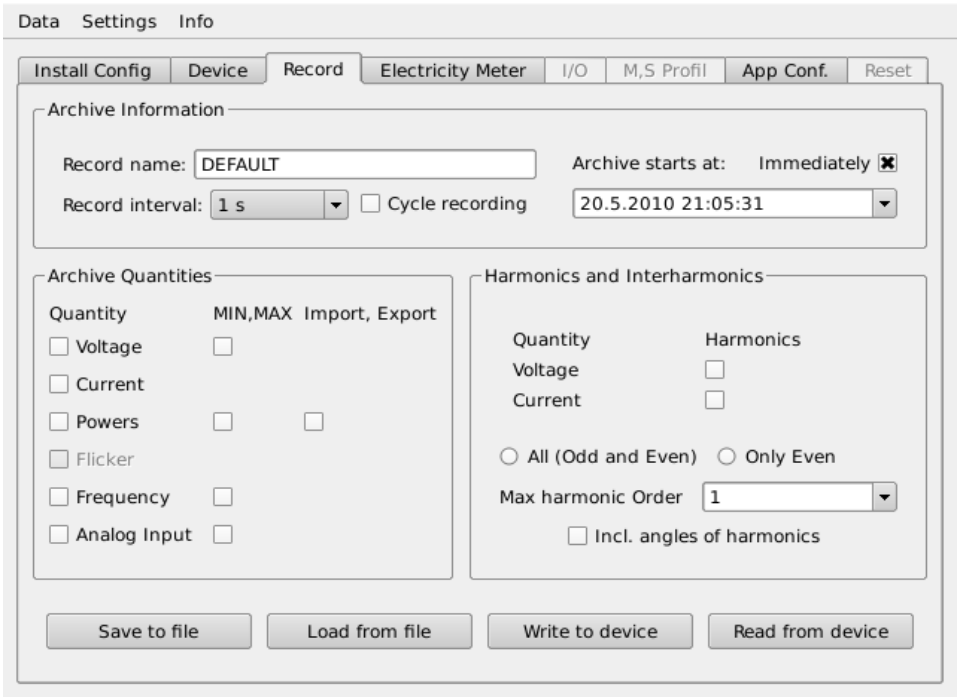


Obrázek 7: Konfigurace zařízení



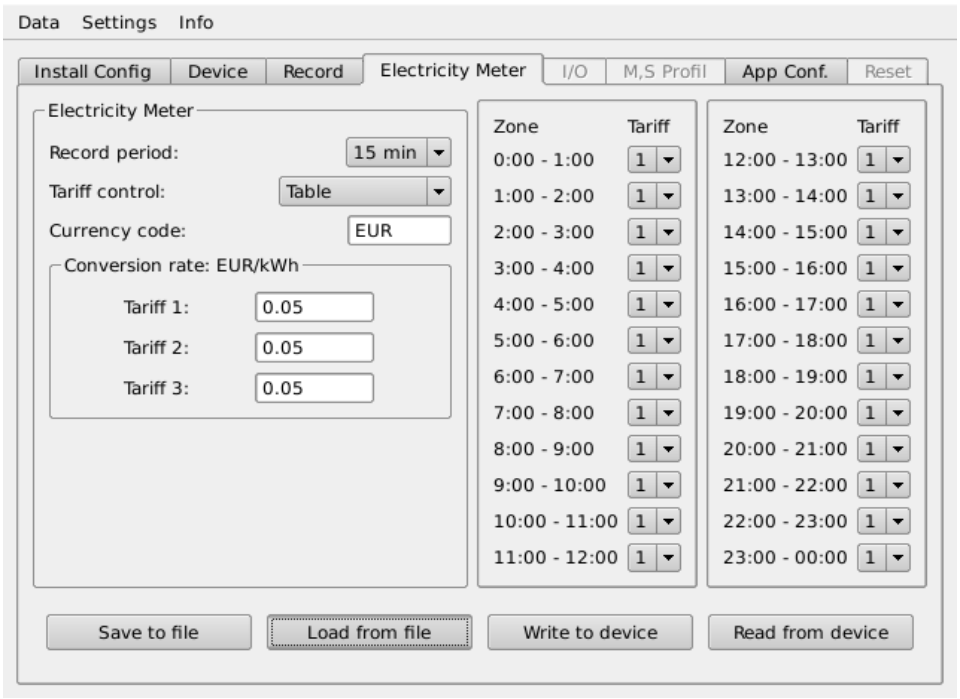
Obrázek 8: Konfigurace zařízení

Na třetí záložce je nastavení záznamů – obrázek 9.



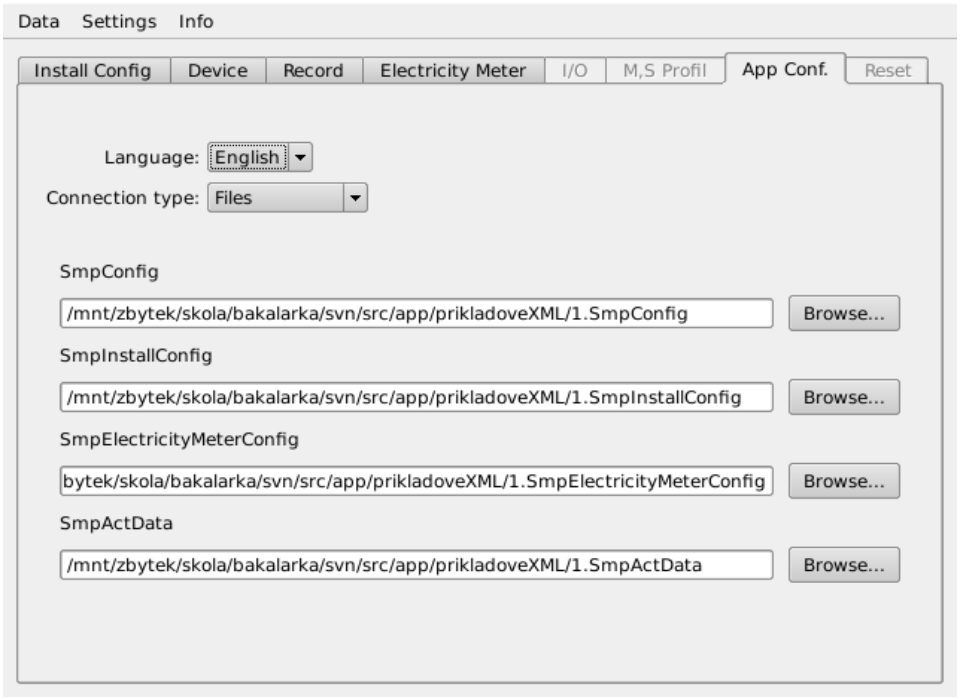
Obrázek 9: Záznam

Poslední záložka, která ovlivňuje nastavení PMD je konfigurace elektroměru na obrázku 10.

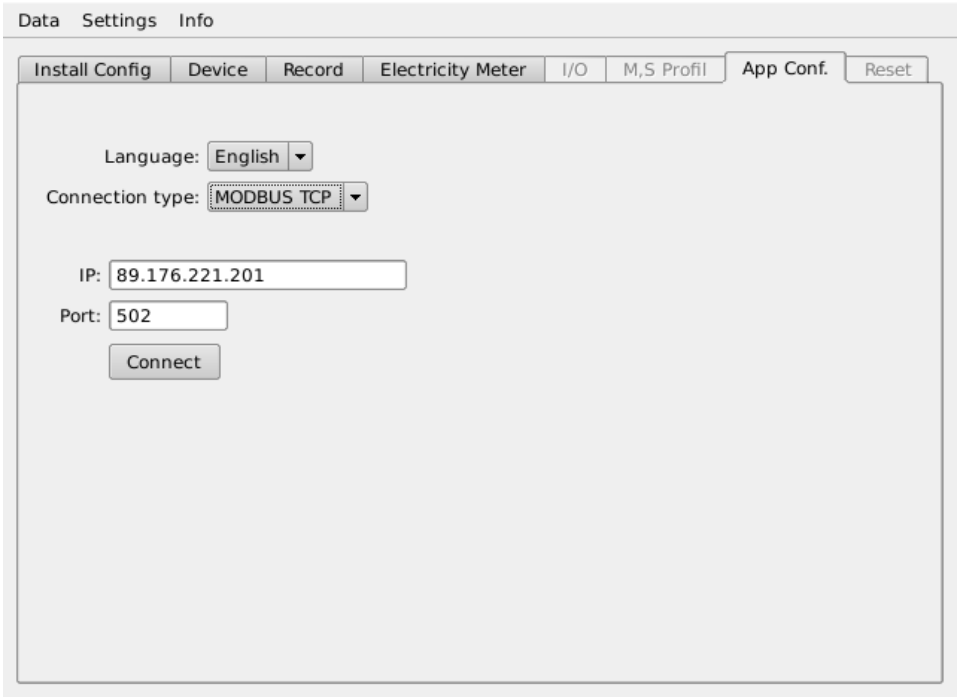


Obrázek 10: Elektroměr

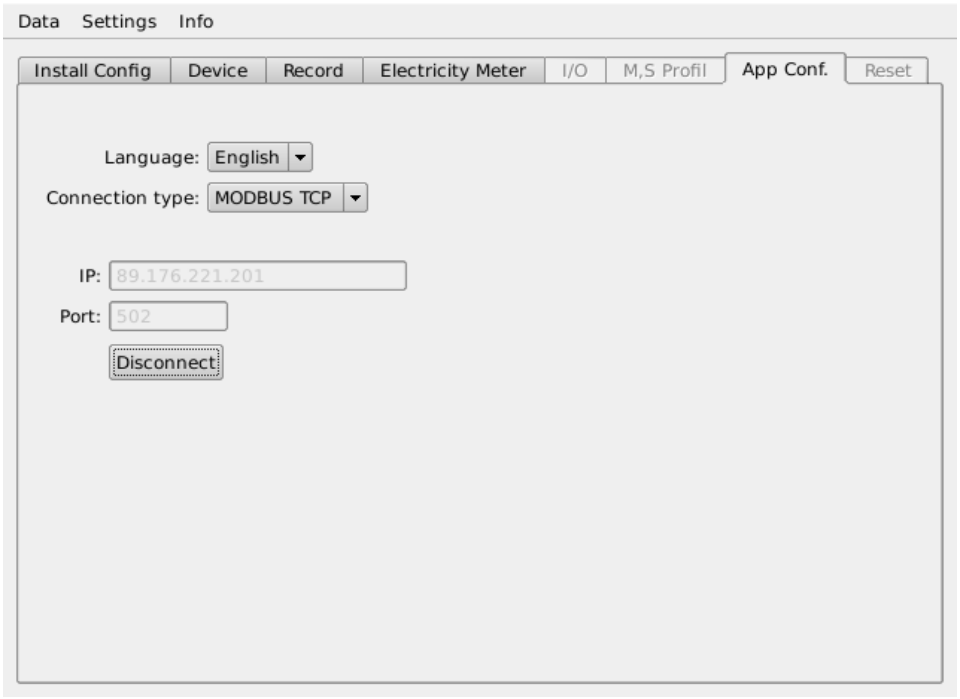
Poslední záložka v nastavení už neovlivňuje PMD ale pouze klienta. Nastavuje se zde jazyk aplikace a komunikace. Je zde také možné ručně připojit PMD pomocí protokolu Modbus.



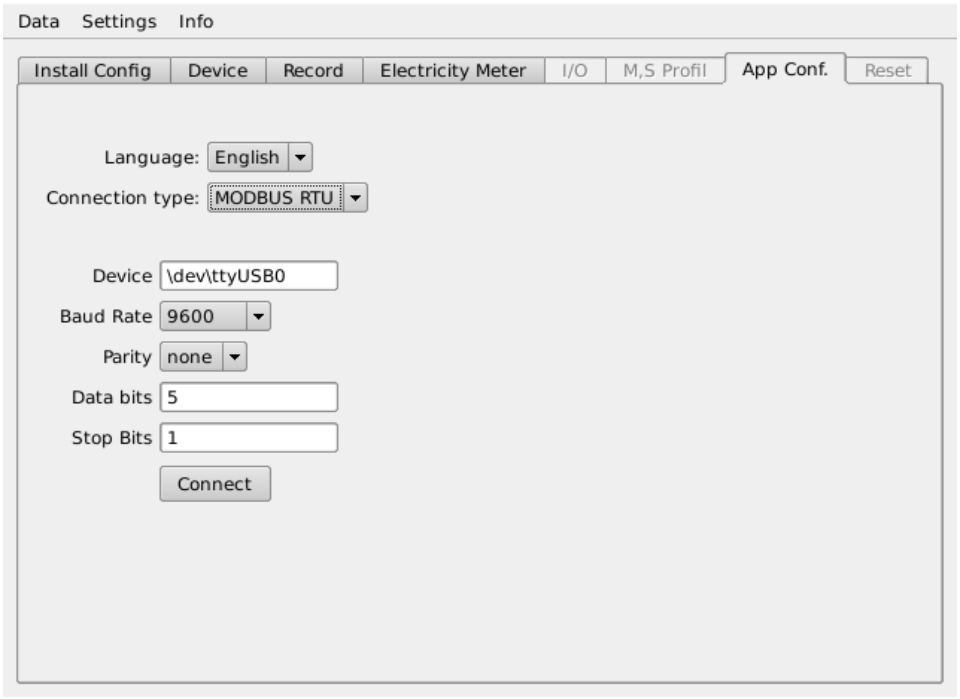
Obrázek 11: Nastavení připojení pomocí přímé výměny souborů



Obrázek 12: Komunikace pomocí Modbus protokolu po ethernetu

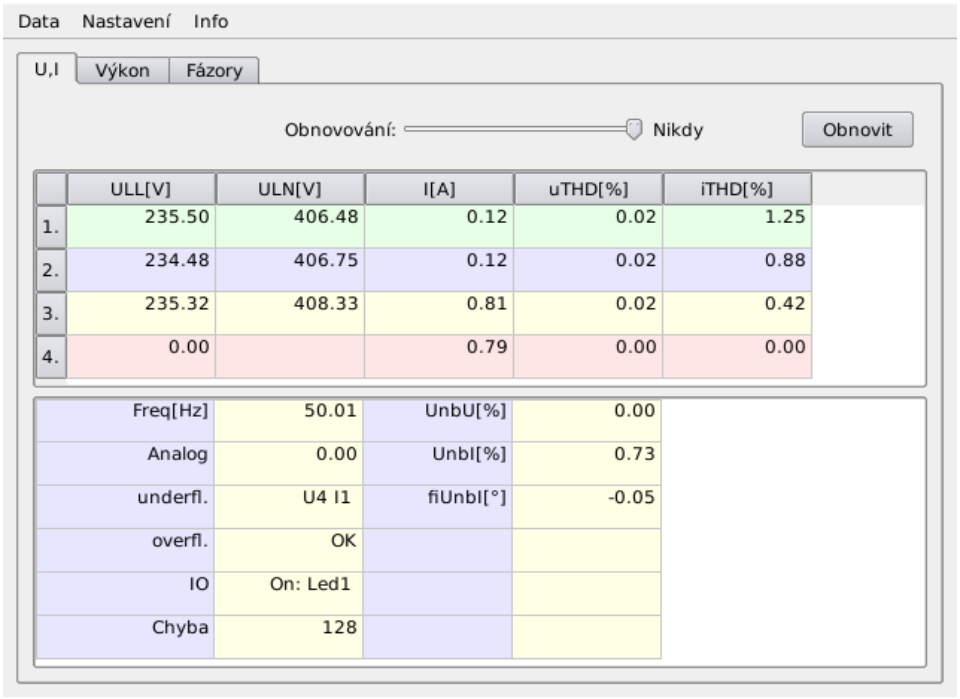


Obrázek 13: Ukázka připojeného zařízení

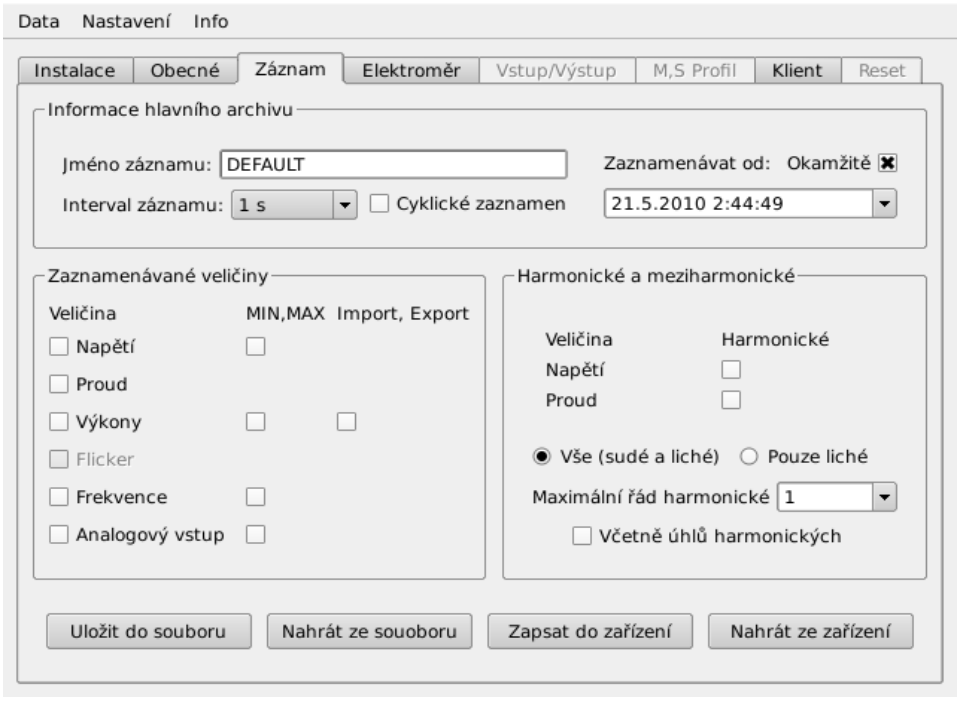


Obrázek 14: Komunikace pomocí protokolu Modbus po sériové lince

Na posledních dvou obrázcích 15 a 16 je ukázán český překlad klienta.



Obrázek 15: Ukázka připojeného zařízení



Obrázek 16: Komunikace pomocí protokolu Modbus po sériové lince

Příloha C - Návod a ukázky z textového klienta

Zde je uvedena sekvenci příkazů a odpovědí konzolového klienta při jeho ukázkovém použití.

Zdrojový kód 1: Zavedení modulu

```
for help type 'help' and press return
> help
help                    Prints this help
show [data]             Display data
     actdataUI          U and I from ActData
     actdataPWR         Powers from ActData
     smpinstallconfig   SmpInstallConfig
     smpconfig          SmpConfig

set [structure] [position] [value]
        Set value to variable on position in structure
set [structure] [variable] [value]
        Set value to variable in structure
> show actdataUI
ULN1[V]        235.50
ULN2[V]        234.48
ULN3[V]        235.32
ULN4[V]          0.00



ULL1[V]        406.48
ULL2[V]        406.75
ULL3[V]        408.33

IL1[V]           0.12
IL2[V]           0.12
IL3[V]           0.81
IL4[V]           0.79
```

```
uTHD1[%]          0.02
uTHD2[%]          0.02
uTHD3[%]          0.02
uTHD4[%]          0.00

iTHD1[%]          1.25
iTHD2[%]          0.88
iTHD3[%]          0.42
iTHD4[%]          0.00

Freq[Hz]         50.01
underfl.        U4 I1
overfl.            OK
IO           On: Led1
Error             128
UnbU[%]           0.00
UnbI[%]           0.73
fiUnbI[angle]       -0.05
> show smpinstallconfig
0 defFreq(u16)                                    50
1 NomU(float)                                 230.00
2 NomPwr(float)                            100000.00
3 MeasureMethod(u8)                               2
4 MTN(u16)                                    65535
5 MTNN(u16)                                   65535
6 MTP(u16)                                    32773
7 MTPN(u16)                                   32773
> set smpinstallconfig 4 1234
Value for smpinstallconfig MTN set to 1234
>
```

Příloha D - Obsah CD

Přiložené CD obsahuje zdrojové soubory této práce ve formátu LaTeXvčetně původních obrázků a schémat. Přiložen je i soubor makefile pro zkompilování souborů.

V samostatné složce jsou umístěny screenshoty grafického klienta.

Zdrojové soubory klientů na přiloženém CD nejsou, protože podléhají firemnímu tajemství společnosti KMB.